Project name-Blockchain Technology For Agriculture Docs Chain

Nm team id- NM2023TMID06454

Date-25-10-2023

Project Report

Project Report

1.Introduction

Agriculture is a big part of the economy of any country because it helps feed the entire population. It connects and communicates with all of the related industries. If the agriculture base is strong, it is generally regarded as a socially and politically stable society. Many modern farms make use of cutting-edge technology and scientific and technological ideas [1].

The following are some of the reasons for food supply chain problems and processing environment challenges.

The maximization of the profits relies on some farmers' vegetables and fruits with chemicals. Chemical fertilizers, insecticides, and other compounds are used in several plants and fruits.

As a result, pesticide residues in vegetables and fruits become excessive. It is a significant health risk.

Food gets contaminated with heavy metals. The irrigation water source of crops is polluted by the excessive intrusion of heavy metal elements such as lead, tin, mercury, and zinc, which are dangerous to human health.

Food additives are used excessively in food processing. Some nefarious enterprises use excessive food additives, antibiotics, hormones, and harmful substances [2].

The following are some of the most common blockchain applications [1]:

(i) Agribusiness insurance.
(ii) Smart farming.
(iii) Traceability.
(iv) Land registration.
(v) Food supply chain.
(vi) Security and safety farms.

Agricultural product e-commerce.

1.1 Project overview

Blockchain have been an interesting research area for a long time and the benefits it provides have been used by a number of various industries. Similarly, the healthcare sector stands to benefit immensely from the blockchain technology due to security, privacy, confidentiality and decentralization. Nevertheless, the Electronic Health Record (EHR) systems face problems regarding data security, integrity and management. In this paper, we discuss how the blockchain technology can be used to transform the EHR systems and could be a solution of these issues. We present a framework that could be used for the implementation of blockchain technology in healthcare sector for EHR.

The aim of our proposed framework is firstly to implement blockchain technology for EHR and secondly to provide secure storage of electronic records by defining granular access rules for the users of the proposed framework. Moreover, this framework also discusses the scalability problem faced by the blockchain technology in general via use of off-chain storage of the records. This framework provides the EHR system with the benefits of having a scalable, secure and integral blockchain-based solution.

1.2 purpose

As a formal definition, the blockchain is a distributed ledger to share transactions or sensitive data across untrusted multiple stockholders in a decentralized network. The data are recorded in a sequential chain of hash-linked blocks that facilitate the data distribution to be more manageable than other traditional data storage formats. The blocks are verified and uploaded into the chain-like system by selected nodes via an agreed consensus protocol. This consensus mechanism allows all the parties to engage in the monitoring process when adding data flow. In addition, the duplicates of these data are stored in all involved nodes to ensure no tampering.

To make agricultural applications more efficient and reliable, we can divide blockchain applications into four categories. The first is the provenance of traceability and food authenticity. The second category is smart agricultural data management. The third category is trading finance in supply chain management. The last is the category of other information management systems [3].

In agriculture, collecting data is frequently prohibitively expensive. The blockchain provides a dependable source of truth about the state of crops, inventories, and contracts. Food provenance is tracked using blockchain technology, which aids in the creation of trustworthy food supply chains and develops trust between producers and consumers. It also enables timely payments among stakeholders generated by data changes when used conjointly with smart contracts [4].

Many characteristics of the blockchain make it unique and promising for future industrial applications. For example, blockchain is decentralized, transparent, immutable, irreversible, autonomous, open-source, ownership, provenance (authenticity and origin), and task automation.

Contract automation (smart contracting) eliminates the need for a traditional contract while improving security and lowering transaction costs. Smart contracts are designed with rules and actions that applied to all parties participating in the transaction [5].

E-agriculture, or smart farming, refers to building innovative methods to use modern information and communication technologies (ICTs), such as the Internet of Things (IoT), cloud computing, machine learning, big data, and blockchain, to move towards more feasible agricultural and farming practices. Blockchain technology in agriculture is gaining traction because of its ability to move away from the centralized approach that now governs the farm value chain. The new technologies have produced Agriculture 4.0 or smart farming [6].

Smart contracts help manage the challenges in implementing the revenue sharing algorithm and improve productivity, transparency, security, traceability, and full integration between supply chain levels

2.LITERATURE SURVEY

2.1 Existing problem

between Existing Agricultural Schemes and the Proposed Model. Figure 1 shows the difference between the existing agriculture supply chain (using the centralized database), a standard blockchain-based agricultural supply chain, and the proposed blockchain-based agrarian architecture.

The key reason we chose to work with blockchain and incorporate its features into our architecture was the absence of need for third parties. Additionally, the control over a decentralized ledger stays with the user rather than a centralized authority.

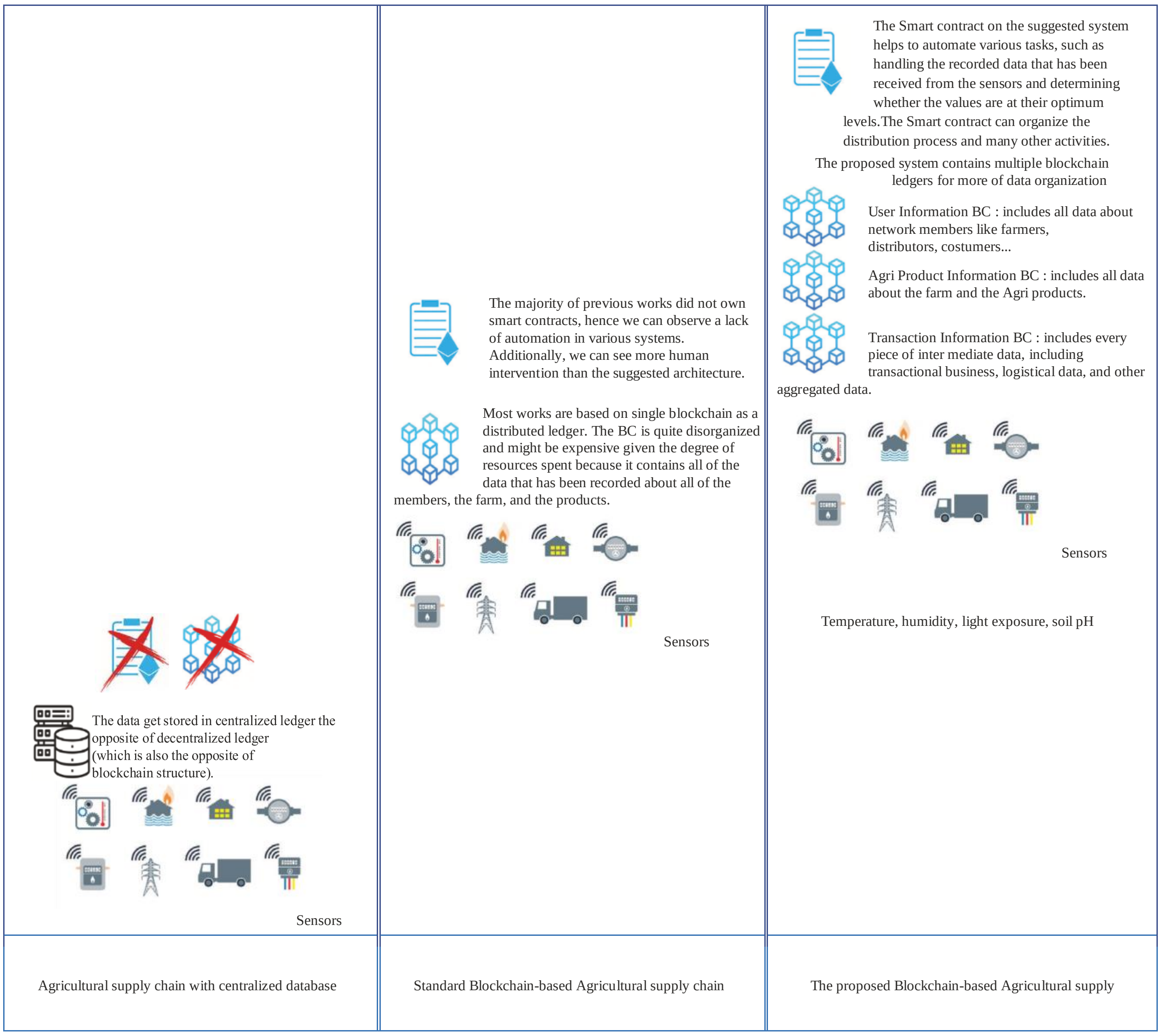
Another benefit of blockchain is that there are no data breaches and hacks. However, the scalability of a centralized system with a single server is limited.

Implementing a blockchain system protects data so it cannot be changed or erased. But the recorded data in the standard blockchain architecture are not well organized, and many previous works did not use smart contracts, which facilitate some operations and tasks without the involvement of network participants.

The suggested model includes several blockchain ledgers that divide the data into user information, agri product information, and transaction information. Additionally, smart contracts are considered a significant factor in this model, which automate many tasks.

Most blockchain systems implement smart contracts provided by the Ethereum platform and its extension platform, Quorum: they compile using Solidity or Serpent into Ethereum virtual machine (EVM) bytecodes. Hyperledger Fabric and Sawtooth, the most active platforms in the Hyperledger family, use Golong, Java, Python, and JavaScript as the major programming languages for smart contract development [3].

1.2. Related Work. The only way to verify and validate transactions in the system is to use IoT devices that are physical consortium members. With the RAFT consensus


The data get stored in centralized ledger the opposite of decentralized ledger (which is also the opposite of blockchain structure).

The majority of previous works did not own smart contracts, hence we can observe a lack of automation in various systems. Additionally, we can see more human intervention than the suggested architecture.

Most works are based on single blockchain as a distributed ledger. The BC is quite disorganized and might be expensive given the degree of resources spent because it contains all of the data that has been recorded about all of the members, the farm, and the products.

The Smart contract on the suggested system helps to automate various tasks, such as handling the recorded data that has been received from the sensors and determining whether the values are at their optimum levels.The Smart contract can organize the distribution process and many other activities.

The proposed system contains multiple blockchain ledgers for more of data organization

User Information BC : includes all data about network members like farmers, distributors, costumers...

Agri Product Information BC : includes all data about the farm and the Agri products.

Transaction Information BC : includes every piece of inter mediate data, including transactional business, logistical data, and other aggregated data.

Sensors
Sensors
Sensors
Temperature, humidity, light exposure, soil pH
Agricultural supply chain with centralized database
Standard Blockchain-based Agricultural supply chain
The proposed Blockchain-based Agricultural supply

Figure 1: Comparison between existing agricultural schemes and the proposed model.

Reference:

[1] G. S. Sajja, K. P. Rane, K. Phasinam, T. Kassanuk, E. Okoronkwo, and P. Prabhue, “Towards applicability of Blockchain in Agriculture sector,” *Materials Today Proceedings*, vol. 5, 2021.

[2] J. Lin, Z. Shen, A. Zhang, and Y. Chai, “Blockchain and IoT Based Food Traceability for Smart Agriculture,” in *Proceedings of the 3rd International Conference on Crowd Science and Engineering*, Singapore, 2018.

[3] W. Lin, X. Huang, V. Wang et al., “Blockchain technology in current agricultural systems: from techniques to applications,” *IEEE Access*, vol. 8, pp. 143920–143937, 2020.

[4] H. Xiong, T. Dalhaus, P. Wang, and J. Huang, “Blockchain technology for agriculture: applications and rationale,” *IEEE Access*, vol. 3, pp. 1–7, 2020.

[5] P. Dutta, T. M. Choi, S. Somani, and R. Butala, “Blockchain technology in supply chain operations: Applications, challenges and research opportunities,” *Transportation Research Part E: Logistics and Transportation Review*, vol. 142, Article ID 102067, 2020.

[6] G. Leduc, S. Kubler, and J. P. Georges, “Innovative Blockchain-based farming marketplace and Smart contract performance evaluation,” *Journal of Cleaner Production*, vol. 306, pp. 1–15, 2021.

[7] P. Bottoni, N. Gessa, G. Massa, R. Pareschi, H. Selim, and E. Arcuri, “Intelligent smart contracts for innovative supply chain management,” *Frontiers in Blockchain*, vol. 3, no. 52, 2020.

[8] K. Dey and U. Shekhawat, “Blockchain for sustainable e-agriculture: literature review,” *Architecture for data management, and implications, Journal of Cleaner Production*, vol. 316, pp. 1–17, 2021.

[9] L. Hang, I. Ullah, and D. Kim, “A secure fish farm platform based on Blockchain for agriculture data integrity,” *Computers and Electronics in Agriculture*, vol. 170, pp. 1–15, 2020.

[10] T. H. Pranto, A. A. Noman, A. Mahmud, and A. K. M. Haque, "Blockchain and Smart contract for IoT enabled smart agriculture," 2021, https://arxiv.org/abs/2104.00632.
[11] X. Huang, D. Ye, and L. Shu, "Securing parked vehicle assisted fog computing with Blockchain and optimal Smart contract design," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 2, pp. 426–441, 2020.
[12] L. Wang, L. Xu, S. Liu et al., "Smart contract-based agricultural food supply chain traceability," *IEEE Access*, vol. 9, pp. 9296–9307, 2021.
[13] K. Salah, N. Nizamuddin, and M. Omar, "Blockchain-based soybean traceability in agricultural supply chain," *IEEE Access*, vol. 7, pp. 73295–73305, 2019.
[14] I. Widi Widayat and M. Ko¨ppen, *Blockchain Simulation Environment on Multi-Image Encryption for Smart Farming Application*, Springer International Publishing, Berlin, Germany, 2021.
C. M. Balaceanu, I. Marcu, and G. Suciu, "Telemetry System for Smart Agriculture," *Business Information Systems Workshops*, Springer, Berlin, germany, 2019.

Problem statement definition

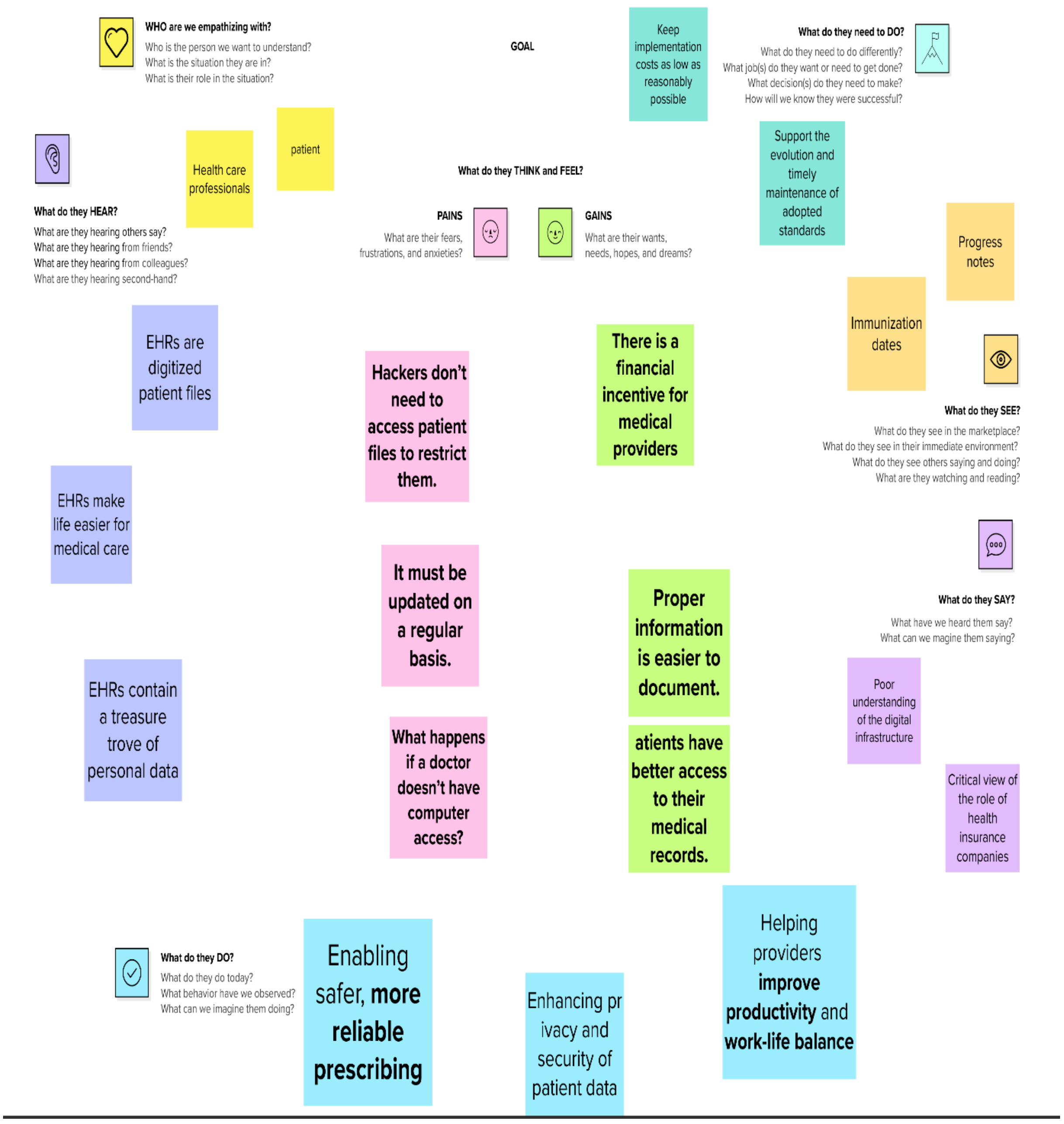
It's no secret that many physicians are unhappy with their electronic health records (EHRs). They say they spend too much time keying in data and too little making eye contact with patients. They say their electronic records are clunky, poorly designed, hard to navigate, and cluttered with useless detail that colleagues have cut and pasted to meet documentation requirements. Meanwhile, the data they really need are buried almost beyond retrieval.

Not all physicians feel this way. Two-thirds of primary care physicians say there are satisfied with their current EHRs, according to a 2018 survey by The Harris Poll. But the critics have a point. Current EHRs are not well-designed to meet the needs of users. And they don't do enough to make clinicians smarter and more efficient. This doesn't mean we would be better off in the paper world of 10 years ago. But it does mean that EHRs need improvement.

3. IDEATION & PROPOSED SOLUTION

Empathy Map Canvas

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user’s behaviours and attitudes.It is a useful tool to helps teams better understand their users.Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user’s perspective along with his or her goals

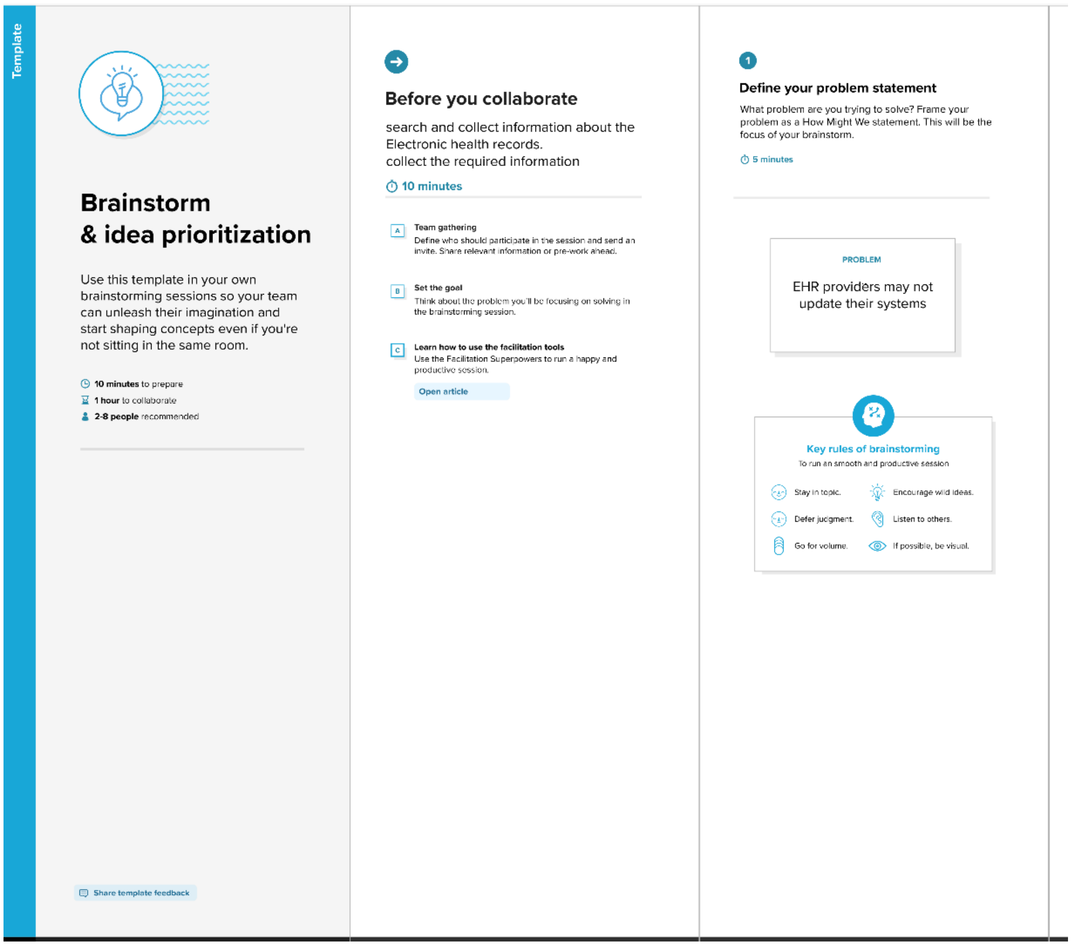

WHO are we empathizing with?
Who is the person we want to understand?
What is the situation they are in?
What is their role in the situation?
GOAL
Keep implementation costs as low as reasonably possible
What do they need to DO?
What do they need to do differently?
What job(s) do they want or need to get done?
What decision(s) do they need to make?
How will we know they were successful?
patient
Health care professionals
What do they THINK and FEEL?
Support the evolution and timely maintenance of adopted standards
What do they HEAR?
What are they hearing others say?
What are they hearing from friends?
What are they hearing from colleagues?
What are they hearing second-hand?
PAINS
What are their fears, frustrations, and anxieties?
GAINS
What are their wants, needs, hopes, and dreams?
Progress notes
Immunization dates
EHRs are digitized patient files
Hackers don't need to access patient files to restrict them.
There is a financial incentive for medical providers
What do they SEE?
What do they see in the marketplace?
What do they see in their immediate environment?
What do they see others saying and doing?
What are they watching and reading?
EHRs make life easier for medical care
It must be updated on a regular basis.
Proper information is easier to document.
What do they SAY?
What have we heard them say?
What can we magine them saying?
Poor understanding of the digital infrastructure
EHRs contain a treasure trove of personal data
What happens if a doctor doesn't have computer access?
atients have better access to their medical records.
Critical view of the role of health insurance companies
What do they DO?
What do they do today?
What behavior have we observed?
What can we imagine them doing?
Enabling safer, more reliable prescribing
Enhancing pr ivacy and security of patient data
Helping providers improve productivity and work-life balance

3.2 Brainstorm & Idea Prioritization Template:

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

Step-1: Team Gathering, Collaboration and Select the Problem Statement



Template

Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

- **10 minutes** to prepare
- **1 hour** to collaborate
- **2-8 people** recommended

Share template feedback

Before you collaborate

search and collect information about the Electronic health records.
collect the required information

10 minutes

A **Team gathering**
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B **Set the goal**
Think about the problem you'll be focusing on solving in the brainstorming session.

C **Learn how to use the facilitation tools**
Use the Facilitation Superpowers to run a happy and productive session.

Open article

1

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM

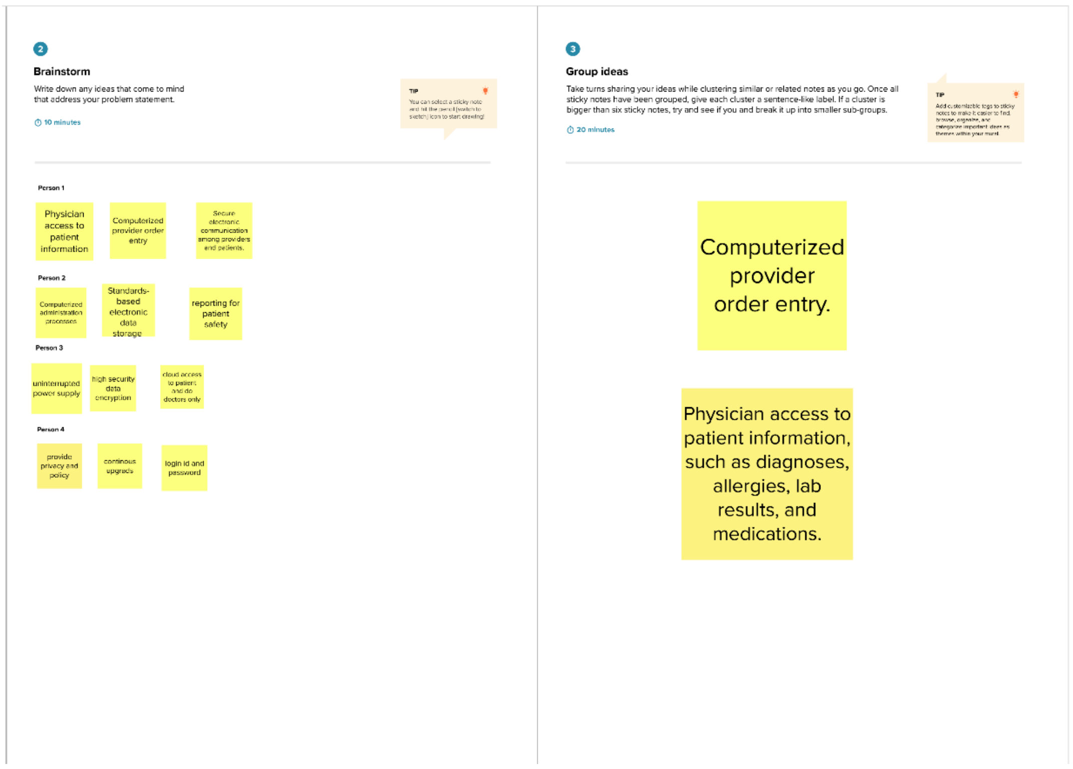
EHR providers may not update their systems

Key rules of brainstorming

To run an smooth and productive session

Stay in topic.	Encourage wild ideas.
Defer judgment.	Listen to others.
Go for volume.	If possible, be visual.

Step-2: Brainstorm, Idea Listing and Grouping


2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP

You can select a sticky note and hit the pencil (switch to sketch) icon to start drawing!

Person 1

Physician access to patient information

Computerized provider order entry

Secure electronic communication among providers and patients.

Person 2

Computerized administration processes

Standards-based electronic data storage

reporting for patient safety

Person 3

uninterrupted power supply

high security data encryption

cloud access to patient and do doctors only

Person 4

provide privacy and policy

continous upgreds

login id and password

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

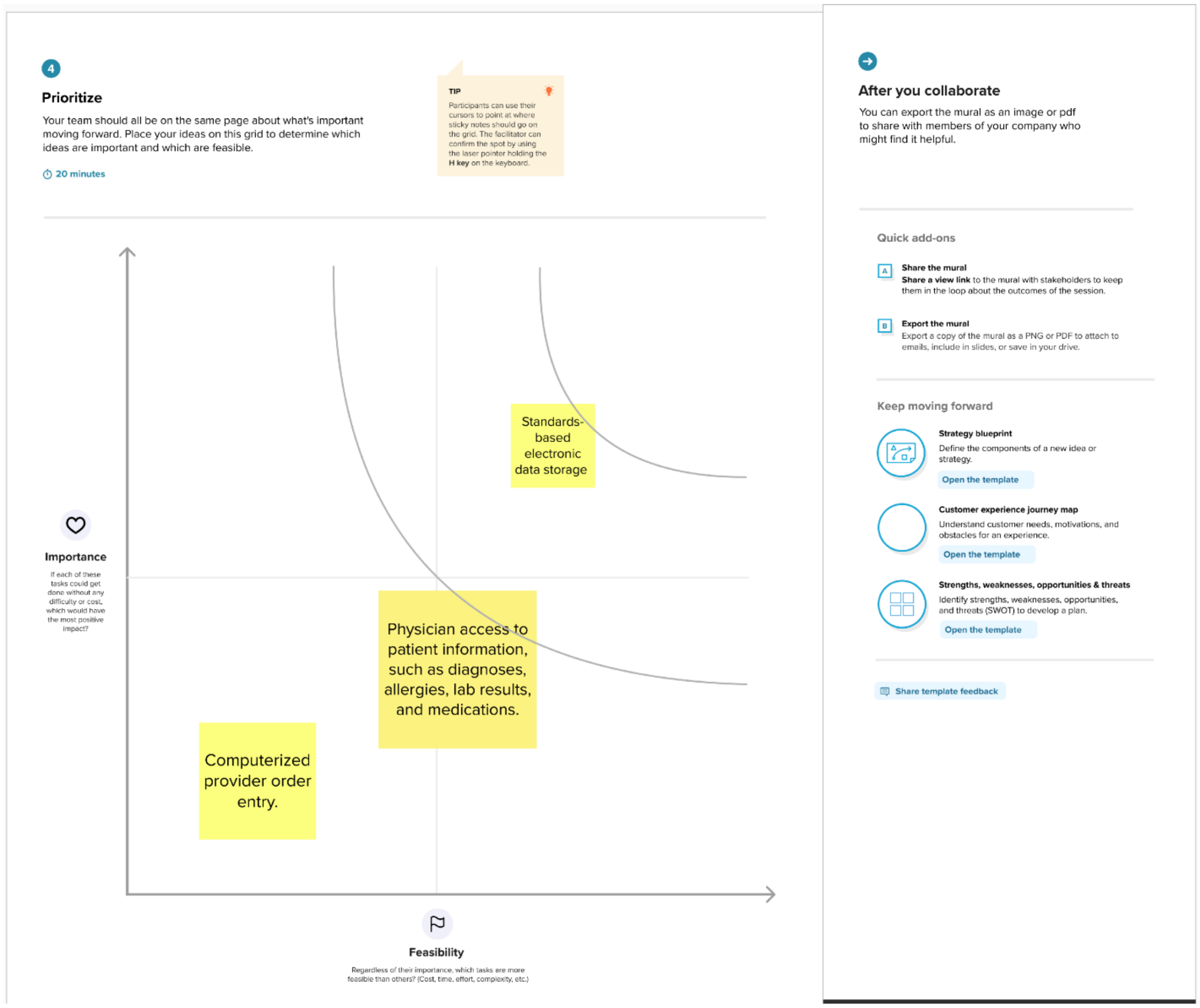
TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

Computerized provider order entry.

Physician access to patient information, such as diagnoses, allergies, lab results, and medications.

Step-3: Idea Prioritization



4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes

TIP

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the **H key** on the keyboard.

Standards-based electronic data storage

Physician access to patient information, such as diagnoses, allergies, lab results, and medications.

Computerized provider order entry.

Importance

If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?

Feasibility

Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

After you collaborate

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

Quick add-ons

A **Share the mural**
Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.

B **Export the mural**
Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive.

Keep moving forward

Strategy blueprint
Define the components of a new idea or strategy.
Open the template

Customer experience journey map
Understand customer needs, motivations, and obstacles for an experience.
Open the template

Strengths, weaknesses, opportunities & threats
Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.
Open the template

Share template feedback

4.REQUIREMENT ANALYSIS

4.1 functional requiremnets

Visual studio code

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages and runtimes (such as C++, C#, Java, Python, PHP, Go, .NET). Begin your journey with VS Code with these introductory videos.

History

Visual Studio Code was first announced on April 29, 2015, by Microsoft at the 2015 Build conference. A preview build was released shortly thereafter.[15]

On November 18, 2015, the source of Visual Studio Code was released under the MIT License and made available on GitHub. Extension support was also announced.[16] On April 14, 2016, Visual Studio Code graduated from the public preview stage and was released to the Web.[17] Microsoft has released most of Visual Studio Code's source code on GitHub under the permissive MIT License,[6][18] while the binary releases by Microsoft are freeware,[8] and include proprietary code.[5] A community distribution, called VSCodium, is maintained, which provides MIT licensed binaries.[10][19][20

Features[edit]

Visual Studio Code is a source-code editor that can be used with a variety of programming languages, including C, C#, C++, Fortran, Go, Java, JavaScript, Node.js, Python, Rust, and Julia.[21][22][23][24][25] It is based on the Electron framework,[26] which is used to develop Node.js web applications that run on the Blink layout engine. Visual Studio Code employs the same editor component (codenamed "Monaco") used in Azure DevOps (formerly called Visual Studio Online and Visual Studio Team Services).[27]

Out of the box, Visual Studio Code includes basic support for most common programming languages. This basic support includes syntax highlighting, bracket matching, code folding, and configurable snippets. Visual Studio Code also ships with IntelliSense for JavaScript, TypeScript, JSON, CSS, and HTML, as well as debugging support for Node.js. Support for additional languages can be provided by freely available extensions on the VS Code Marketplace.[28]

Visual Studio Code Insiders logo

Instead of a project system, it allows users to open one or more directories, which can then be saved in workspaces for future reuse. This allows it to operate as a language-agnostic code editor for any language. It supports many programming languages and a set of features that differs per language. Unwanted files and folders can be excluded from the project tree via the settings. Many Visual Studio Code features are not exposed through menus or the user interface but can be accessed via the command palette.[29]

Visual Studio Code can be extended via extensions,[30] available through a central repository. This includes additions to the editor[31] and language support.[29] A notable feature is the ability to create extensions that add support for new languages, themes, debuggers, time travel debuggers, perform static code analysis, and add code linters using the Language Server Protocol.[32]

Source control is a built-in feature of Visual Studio Code. It has a dedicated tab inside of the menu bar where users can access version control settings and view changes made to the current project. To use the feature, Visual Studio Code must be linked to any supported version control system (Git, Apache Subversion, Perforce, etc.). This allows users to create repositories as well as to make push and pull requests directly from the Visual Studio Code program.

Visual Studio Code includes multiple extensions for FTP, allowing the software to be used as a free alternative for web development. Code can be synced between the editor and the server, without downloading any extra software.

Visual Studio Code allows users to set the code page in which the active document is saved, the newline character, and the programming language of the active document. This allows it to be used on any platform, in any locale, and for any given programming language.[promotion?]

Visual Studio Code collects usage data and sends it to Microsoft, although this can be disabled.[33] Some of the telemetry code is accessible to the public,[34] but according to Visual Studio Code maintainers, some telemetry functionality is also added to the program before it is released with a proprietary license.[35][5]

Node js

Node.js is a cross-platform, open-source server environment that can run on Windows, Linux, Unix, macOS, and more. Node.js is a back-end JavaScript runtime environment, runs on the V8 JavaScript engine, and executes JavaScript code outside a web browser.

Node.js lets developers use JavaScript to write command line tools and for server-side scripting. The ability to run JavaScript code on the server is often used to generate dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web-application development around a single programming language, as opposed to using different languages for the server- versus client-side programming.

Node.js has an event-driven architecture capable of asynchronous I/O. These design choices aim to optimize throughput and scalability in web applications with many input/output operations, as well as for real-time Web applications (e.g., real-time communication programs and browser games).

The Node.js distributed development project was previously governed by the Node.js Foundation, and has now merged with the JS Foundation to form the OpenJS Foundation. OpenJS

Foundation is facilitated by the Linux Foundation's Collaborative Projects program.

Platform architecture**[edit]**

Node.js brings event-driven programming to web servers, enabling development of fast web servers in JavaScript. Developers can create scalable servers without using threading, by using a simplified model of event-driven programming that uses callbacks to signal the completion of a task. Node.js connects the ease of a scripting language (JavaScript) with the power of Unix network programming.

Node.js was built on top of Google's V8 JavaScript engine since it was open-sourced under the BSD license. It is proficient with internet fundamentals such as HTTP, DNS, and TCP. JavaScript was also a well-known language, making Node.js accessible to the web development community.

Console

The **console** is a module provided by Node.js that is akin to the JavaScript console in the browser when you inspect a webpage. The console has methods that are available for us to use for debugging purposes.

- `console.log()`: Frequently used to log some sort of output.
- `console.warn()`: Explicitly delivers a warning to the console.
- `console.error()`: Explicitly delivers an error message to the console. You can log an error as a string or as an object. If logged as a new `Error()`, a traceback will be included as part of the message.
- `console.trace()`: Logs a traceback when an error occurs in your code. Gives line number and column number of the file that the error probably occurred.

File System

The **file system (fs) module** allows us to interact with files in Node.js. There are synchronous and asynchronous methods that can be used to read or write to a file using the fs module. In contrast to using console or the Buffer class, we need to import the fs module into the file that we would like to use in order to get it to work.

The code example below shows how the `readFile`, an asynchronous method, works. Notice that the last argument in the method is a callback function whose first argument is an error. By definition this callback will always pass in the error first before the data.

Metamask

MetaMask is a popular cryptocurrency wallet and browser extension that allows users to manage their Ethereum-based assets and interact with decentralized applications (DApps) on the Ethereum blockchain. It was developed by ConsenSys, a blockchain technology company.

Here are some key features and functions of MetaMask:

1. Wallet: MetaMask serves as a secure digital wallet that can store Ethereum (ETH) and other Ethereum-based tokens (ERC-20 and ERC-721 tokens). Users can send, receive, and manage their cryptocurrency holdings through the wallet.

2. Browser Extension: MetaMask is primarily a browser extension available for various web browsers like Chrome, Firefox, and Brave. It integrates with your browser, making it easy to interact with Ethereum-based DApps directly from your web browser.

3. DApp Interaction: MetaMask allows users to connect to decentralized applications (DApps) on the Ethereum blockchain. Users can access various DeFi (Decentralized Finance) platforms, NFT (Non-Fungible Token) marketplaces, games, and other blockchain-based services through the extension.

4. Private Keys: MetaMask uses a hierarchical deterministic (HD) wallet system and stores private keys locally on your device, enhancing security. Users are responsible for safeguarding their private keys and seed phrases to protect their funds.

5. Cross-Platform Compatibility: MetaMask is available as both a browser extension and a mobile app for iOS and Android devices. This allows users to access their wallet and interact with DApps on multiple platforms.

6. Network Selection: Users can choose which Ethereum network they want to interact with, such as the Ethereum mainnet, testnets (Ropsten, Rinkeby, Goerli, and Kovan), and custom networks.

7. Security Features: MetaMask includes various security features, such as password protection, biometric authentication, and a phishing detection system to help users avoid potential scams.

8. Token Swaps: MetaMask provides a token swap feature that allows users to exchange one cryptocurrency for another directly within the wallet.

9. Gas Fees: Users can customize gas fees for their transactions, allowing them to control the speed and cost of their Ethereum transactions.

MetaMask has gained widespread popularity due to its user-friendly interface and the convenience it offers for interacting with the Ethereum blockchain and various decentralized applications. However, users should always exercise caution and follow best security practices when managing their cryptocurrency assets.

Procedure to install metamask

To install MetaMask, you can follow these steps. Please note that the procedure may vary slightly depending on your web browser (commonly used on Chrome or Firefox). As of my last knowledge update in September 2021, MetaMask was available as a browser extension. Please ensure you are using a secure and up-to-date web browser. Here's a general procedure for installing MetaMask:

1. Open your web browser: Launch your preferred web browser (e.g., Chrome, Firefox, or Brave).

2. Go to the MetaMask website:

- Type "MetaMask" in your browser's search bar or go directly to the MetaMask website: https://metamask.io/.

3. Click on "Get Chrome Extension" (or the equivalent for your browser): You will see an option to install the MetaMask browser extension. Click on this option.

4. Install the extension:

- For Chrome: Click "Add to Chrome" or "Add to Brave" if you're using the Brave browser.
- For Firefox: Click "Add to Firefox."

5. Confirm the installation:

- A pop-up or prompt will appear asking you to confirm the installation. Click "Add Extension" to proceed.

6. Wait for the installation: The extension will be downloaded and added to your browser. Once the installation is complete, you'll see the MetaMask fox icon in your browser's extensions area.

7. Set up your wallet:

- Click the MetaMask fox icon in your browser.

- Click "Get Started" to begin the wallet setup process.

- You will be guided through creating a new wallet or importing an existing one. Follow the instructions provided by MetaMask.

8. Secure your wallet:

- During the wallet setup, you'll be asked to create a password and back up a recovery seed phrase. Make sure to store this recovery phrase in a safe and secure place. This is essential for wallet recovery in case you forget your password or your device is lost.

9. Complete the setup:

- Once your wallet is set up and secured, you can start using MetaMask to interact with Ethereum DApps and manage your cryptocurrency assets.

10. Optional: Customize your settings, including network selection, gas fees, and additional security features within the MetaMask extension.

Remember to keep your wallet password and recovery seed phrase secure and never share it with anyone. Additionally, ensure your browser and MetaMask extension are always up to date for the latest security features and enhancements.

Please note that the installation steps might be slightly different depending on your specific browser version and updates to MetaMask, so always refer to the latest instructions from MetaMask's official website or browser extension store.

4.2 Non functional requirements

Non-functional requirements in the context of a blockchain project refer to aspects of the system that are not directly related to its primary functionality but are crucial for its overall success. These requirements focus on characteristics like performance, security, scalability, and usability. Here are some non-functional requirements commonly associated with blockchain projects:

1. **Security:**

   - **Immutability:** Data on the blockchain should be tamper-proof.

   - **Privacy:** Depending on the use case, some data should be private and accessible only to authorized parties.

   - **Authentication and Authorization:** Secure and robust methods of authenticating users and granting appropriate permissions.

   - **Consensus Algorithm Security:** The chosen consensus algorithm should be resistant to attacks.

2. **Performance:**

   - **Transaction Throughput:** The ability to handle a high volume of transactions per second (TPS) while maintaining low latency.

   - **Scalability:** The system should be able to scale as the user base and transaction volume grow.

   - **Network Latency:** Minimizing network latency is critical for real-time applications.

   - **Resource Efficiency:** Efficient use of computational resources to reduce operating costs.

3. **Scalability:**

   - **Horizontal and Vertical Scaling:** The ability to add more nodes (horizontal scaling) or resources (vertical scaling) to accommodate growth.

   - **Sharding:** If applicable, support for sharding to divide the network into smaller, manageable segments.

   - **Interoperability:** Compatibility with other blockchain networks or systems.

4. **Reliability:**

   - **High Availability:** The system should be available and accessible as close to 100% of the time as possible.

   - **Disaster Recovery:** Measures to recover the system in the event of a failure or a disaster.

5. **Regulatory Compliance:**

   - **Data Retention and Compliance:** Compliance with data retention and privacy regulations applicable to the project.

   - **KYC/AML:** Implementing Know Your Customer (KYC) and Anti-Money Laundering (AML) procedures when required.

6. **Usability:**

   - **User-Friendly Interface:** An intuitive and user-friendly interface for users and administrators.

   - **Documentation:** Comprehensive documentation for users and developers.

   - **Training and Support:** Provision of training and support for users and administrators.

7. **Interoperability:**

   - **Compatibility:** Ability to integrate with other systems, networks, or blockchains.

   - **Smart Contract Interoperability:** The capability to interact with smart contracts on different blockchains or platforms.

8. **Compliance:**

   - **Legal and Regulatory Compliance:** Ensuring that the project complies with relevant laws and regulations in the jurisdictions it operates in.

   - **Licensing:** Complying with open-source licensing requirements if applicable.

9. **Environmental Impact:**

   - **Energy Efficiency:** Reducing the environmental impact by optimizing energy consumption, especially in Proof of Work (PoW) blockchains.

10. **Data Storage and Retention:**

    - **Data Storage Costs:** Managing the long-term storage of data on the blockchain and associated costs.

    - **Data Pruning:** Implementing data pruning strategies to manage blockchain bloat.

11. **Auditability and Transparency:**

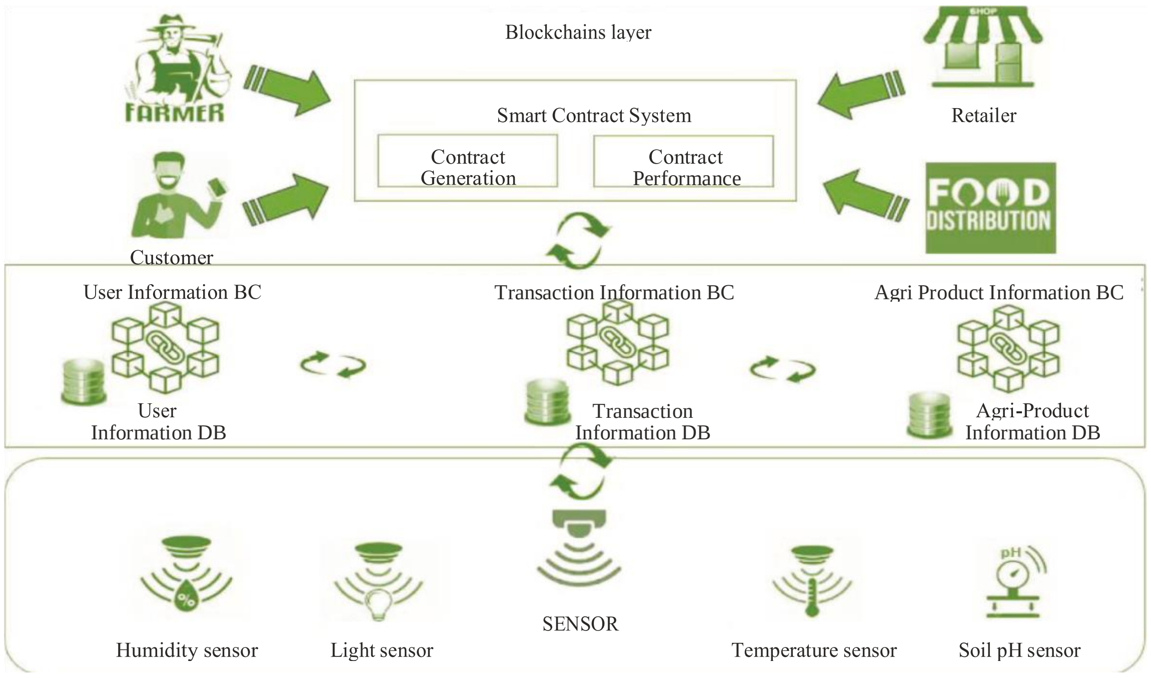
- **Public Ledger:** Ensuring that transactions are transparent and can be audited by authorized parties.

- **Auditing Capabilities:** Supporting the auditing of transactions, contracts, and system performance.

12. **Upgradability:**

- **Hard and Soft Forks:** Preparing for network upgrades through hard forks and soft forks, ensuring minimal disruptions.

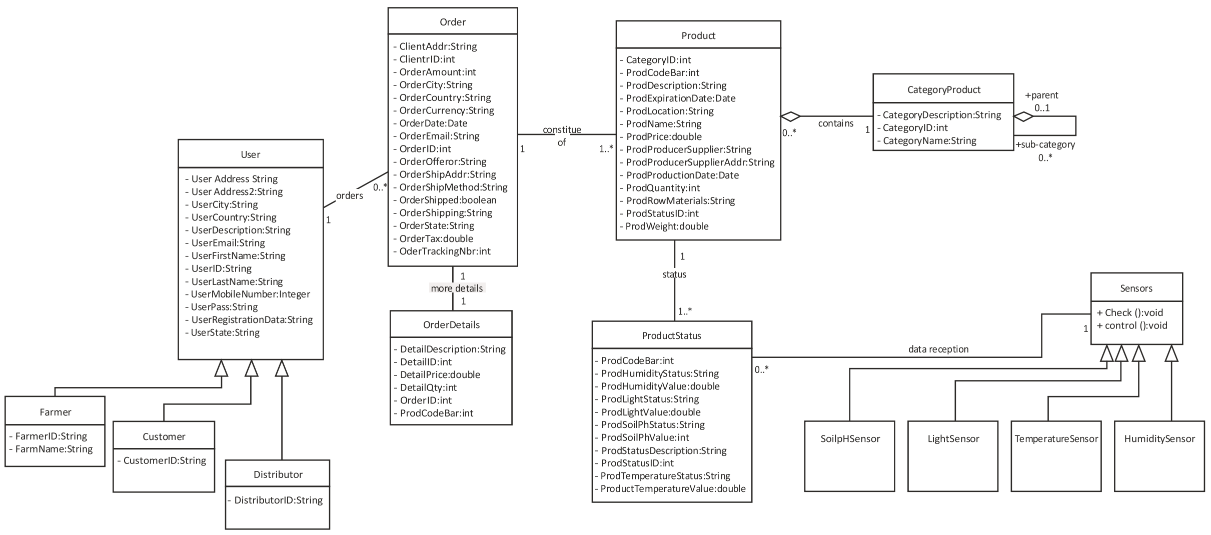
Non-functional requirements are essential to the success of a blockchain project. The specific requirements will vary depending on the project's use case, technology stack, and intended audience. It's crucial to define and



these requirements early in the project's planning and development stages.

5. Project design

5.1 data flow diagram



User stories

User stories are a way to describe the functionality and features of a system from the perspective of its end-users. In the context of electronic health records (EHR) systems, various stakeholders have different needs and use cases. Here are some user stories for different types of users who interact with EHR systems:

1. **As a Patient:**

- As a patient, I want to be able to access my electronic health records online, so I can review my medical history and treatment plans.

- As a patient, I want to receive automated reminders for upcoming appointments and medication refills.

   - As a patient, I want to request prescription refills and appointments online to save time and reduce administrative hassles.

2. **As a Healthcare Provider:**

   - As a healthcare provider, I want an EHR system that allows me to easily access and update patient records to provide the best possible care.

   - As a healthcare provider, I want to receive real-time alerts and notifications for critical patient information, such as allergies and contraindications.

   - As a healthcare provider, I want a user-friendly system for documenting patient encounters, including clinical notes, diagnoses, and treatment plans.

3. **As a Nurse:**

   - As a nurse, I want the ability to scan patient wristbands and medications to ensure medication administration accuracy.

   - As a nurse, I want a system that notifies me of any changes in a patient's condition, so I can respond promptly.

   - As a nurse, I want to easily access a patient's care plan and medication history to provide consistent care.

4. **As a Healthcare Administrator:**

- As a healthcare administrator, I want to track and manage user access and permissions to ensure data security and privacy.

- As a healthcare administrator, I want a system that provides comprehensive reporting and analytics for decision-making and compliance purposes.

- As a healthcare administrator, I want the system to be scalable and support multiple healthcare facilities within our organization.

5. **As a Specialist Consultant:**

- As a specialist consultant, I want to have secure access to relevant patient records when consulting on cases from other facilities.

- As a specialist consultant, I want to provide feedback and recommendations within the EHR system to collaborate with the primary care team.

- As a specialist consultant, I want to view images and test results for in-depth analysis and diagnosis.

6. **As a Pharmacist:**

- As a pharmacist, I want to access patient medication histories to prevent potential drug interactions or duplications.

- As a pharmacist, I want to receive e-prescriptions from healthcare providers for accuracy and efficiency.

- As a pharmacist, I want to communicate with healthcare providers through the EHR system regarding patient medication concerns.

7. **As a Researcher:**

- As a researcher, I want access to de-identified patient data for medical research and studies.

- As a researcher, I want to track patient demographics, conditions, and treatments for epidemiological studies.

- As a researcher, I want to have the ability to export and analyze EHR data for research purposes.

These user stories represent a range of EHR system users and their specific needs. They serve as a basis for developing features and functionality that cater to the diverse requirements of stakeholders in the healthcare ecosystem. Each user story can be further refined and prioritized during the development process.

5.2 solution architecture

Designing a solution architecture for blockchain-based electronic health records (EHR) requires careful consideration of various factors, including data security, privacy, interoperability, and regulatory compliance. Here is a high-level solution architecture for a blockchain EHR system:

1. Blockchain Network:

- **Blockchain Type:** Choose an appropriate blockchain type, such as a permissioned blockchain (e.g., Hyperledger Fabric) or a public blockchain (e.g., Ethereum), depending on the specific use case and regulatory requirements.

- **Consensus Mechanism:** Select a consensus mechanism that aligns with the desired level of decentralization and security. Options include Proof of Work (PoW), Proof of Stake (PoS), Practical Byzantine Fault Tolerance (PBFT), and more.

2. Identity and Access Management:

- **User Authentication:** Implement secure user authentication mechanisms, including multi-factor authentication (MFA) for healthcare providers, patients, and other stakeholders.

- **Identity Verification:** Verify the identity of users, including healthcare providers, patients, and administrators, to ensure data integrity and compliance with regulations.

3. Smart Contracts:

- **Smart Contract Development:** Develop smart contracts to handle EHR-related transactions, such as record creation, access permissions, and updates. Smart contracts should be written securely and audited for vulnerabilities.

- **Access Control:** Use smart contracts to manage access control to EHR data, specifying who can view or update patient records.

4. Data Storage and Encryption:

- **Off-Chain Data Storage:** Store large EHR files (e.g., medical images) off-chain for scalability, using IPFS or similar technologies.

- **On-Chain Metadata:** Store metadata, such as record pointers and access permissions, on the blockchain.

- **Data Encryption:** Implement robust data encryption for both data at rest and data in transit to protect patient information.

5. Interoperability:

- **FHIR Standards:** Ensure interoperability by adhering to Fast Healthcare Interoperability Resources (FHIR) standards, enabling data exchange with other healthcare systems.

- **APIs:** Provide APIs to connect with legacy EHR systems, medical devices, and healthcare applications, allowing data sharing and integration.

6. User Interfaces:

- **Patient Portal:** Develop a user-friendly patient portal for patients to access and manage their EHR data, including appointment scheduling, prescription refills, and medical history review.

- **Provider Dashboard:** Create a comprehensive dashboard for healthcare providers to view patient records, update information, and communicate securely.

7. Regulatory Compliance:

- **HIPAA Compliance:** Ensure compliance with the Health Insurance Portability and Accountability Act (HIPAA) or other applicable data protection regulations, depending on the jurisdiction.

- **Audit Trail:** Implement an immutable audit trail to track all interactions with patient records for compliance and accountability.

8. Privacy and Consent Management:

- **Patient Consent:** Develop a system for patients to control who can access their EHR data and under what conditions.

- **Data De-Identification:** Implement data de-identification techniques to protect patient privacy while still enabling research and analytics.

9. Data Ownership and Portability:

- **Data Ownership:** Clearly define data ownership and control, allowing patients to take their EHR data with them when switching healthcare providers.

10. Scalability and Performance:

- **Load Balancing:** Implement load balancing to ensure the system can handle a high volume of transactions and data storage.

- **Sharding:** Consider sharding the blockchain to distribute data across multiple nodes for improved scalability.

11. Disaster Recovery and Redundancy:

- **Data Backup:** Regularly back up blockchain data to ensure data recovery in case of system failure or disaster.

12. Monitoring and Analytics:

- **Real-Time Monitoring:** Implement real-time monitoring and analytics to detect anomalies, suspicious activities, and performance issues.

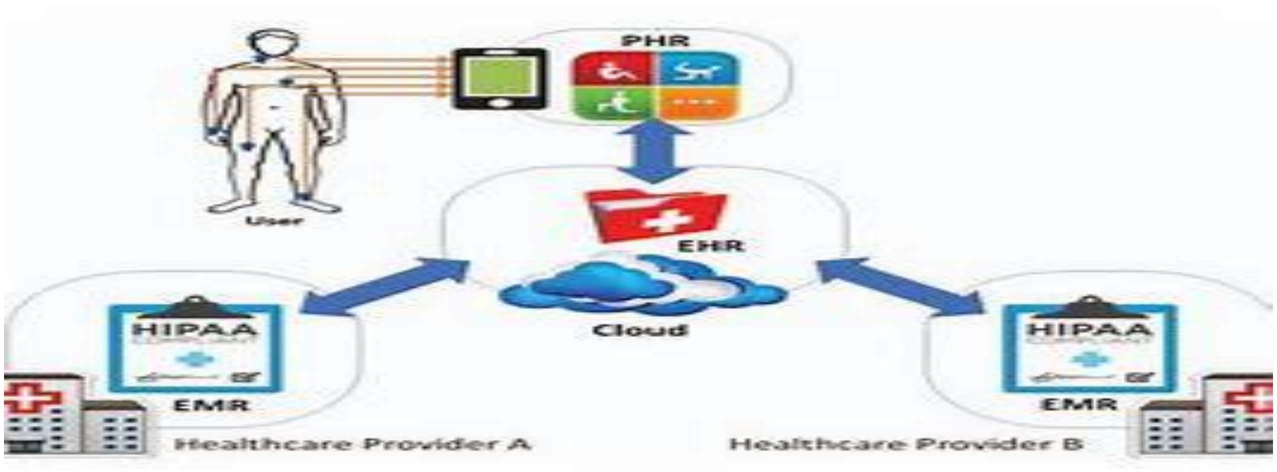
13. Training and Support:

- **User Training:** Provide training for healthcare providers, administrators, and patients to ensure they can effectively use the blockchain EHR system.

14. Integration with External Systems:

- **Integration with Labs and Imaging Systems:** Allow integration with external systems for seamless access to test results and medical images.

This solution architecture is a starting point for designing a blockchain-based EHR system. It should be adapted to meet the specific requirements of the healthcare organization and the regulatory environment in which it operates. Additionally, regular security audits and compliance assessments are essential to maintain the system's integrity and trustworthiness.



6. project planning and solution

6.1 Technical architecture

A technical stack, sometimes referred to as a technology infrastructure or solutions stack, is currently crucial for creating scalable, maintainable applications. A technology stack is a collection of technologies layered on top of each other to develop any application. The technology stack influences the sort of apps you may design, the degree of customization you can make, and the resources you require to build your application.

Proof of concept (PoC) is an essential component of every software project since they reduce possible risks and demonstrate the project's perspective.

Proof of concept should be the foundation of every software development process since it has several benefits for projects. PoC takes care of everything, from reducing failure risks to evaluating the possibility of scalability. To improve the overall research, we have collected all the advantages of this procedure.

Risk minimization is by far a proof of concept's most significant benefit. The development team analyzes potential problems before beginning the procedure.

It is just as crucial to plan your scaling approach as it is to release a top-notch product. In the long term, you need to know how to build your solution technically and commercially.

During a seed round, supporting the idea with concrete evidence of its potential is essential. Nobody pays for abstract ideas.

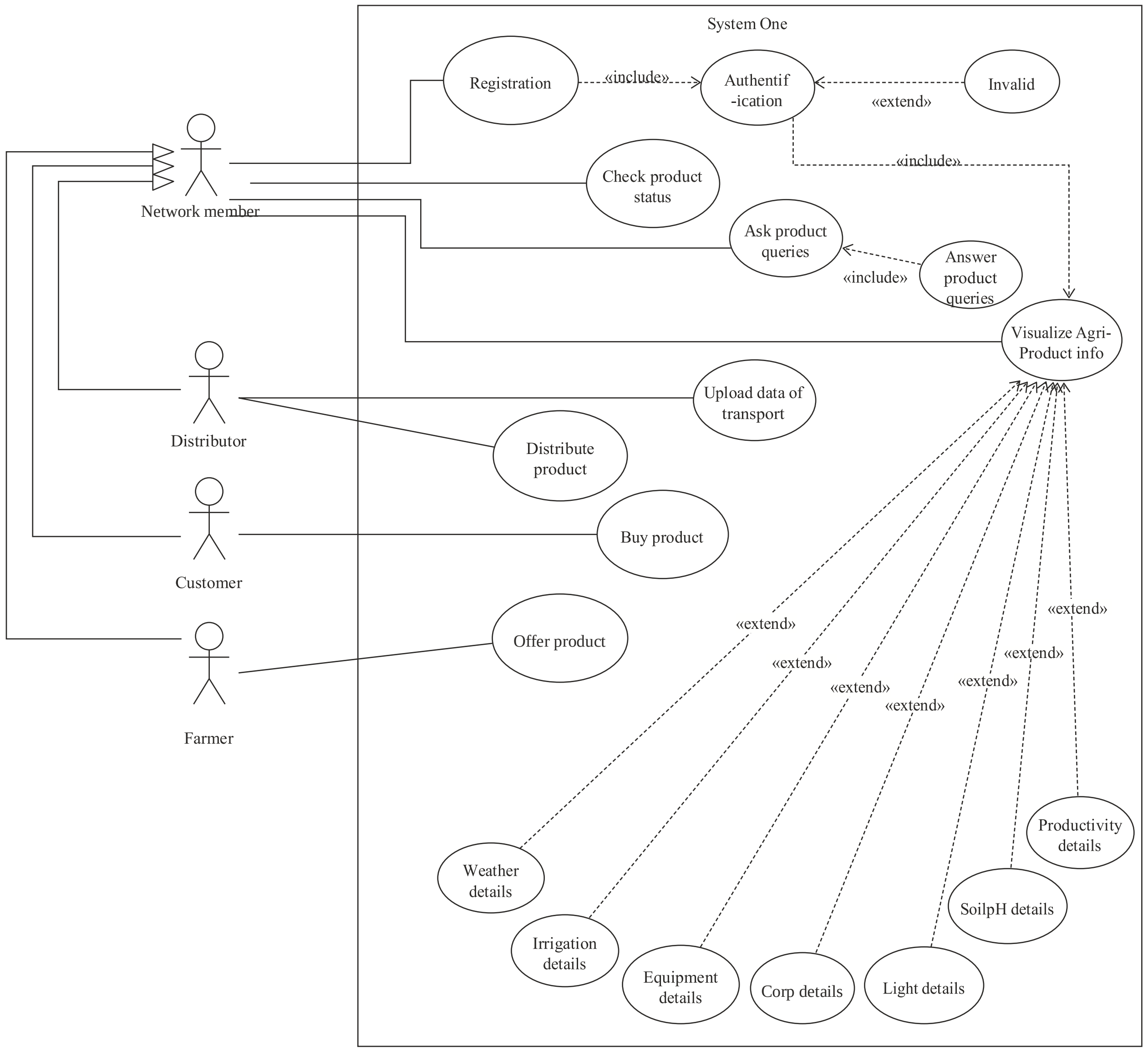


Figure 4: Use case diagram of the proposed model

Building a PoC to test concepts in actual settings and obtain early feedback involves selecting the right technology stack and creating the architecture.

Figure 5 shows the technology stack of the proposed model.

The physical or infrastructure layer assists in gathering environmental data, such as soil pH, temperature, humidity, and light exposure measurements. We also have nodes, which are network members. A node is any device that has permission to access blockchain network.

Blockchain, as a concept, organizes data into user data, agricultural product data, and transaction data. Blockchains employ consensus methods to make sure the nodes reach an agreement. It is a fantastic procedure to increase the network’s efficiency by introducing a new degree of dependability and achieving information security.

Large portions of the model are at the level of smart contracts. The middleman got just eliminated. You will not have to worry about trust issues since there are standards protecting everyone’s legal rights. Since everything gets computerized, higher authorities cannot have any impact, and the procedure is quite open.

The Kaleido interface serves as a functional testing platform. It assists in facilitating access to the decentralized ledger. The users receive great help from this layer. This platform enables access to all data. Peer-to-peer server networks allow users to connect to the blockchain network.

We have selected the blockchain network and its tools/ services. The system software, database, server, and frameworks comprise most of the backend or server-side technology stack. These technologies are appropriate for our planned research. For the automation of various functions, we also have the opportunity of dealing with smart contracts.

The front end of choice for most online apps stays the

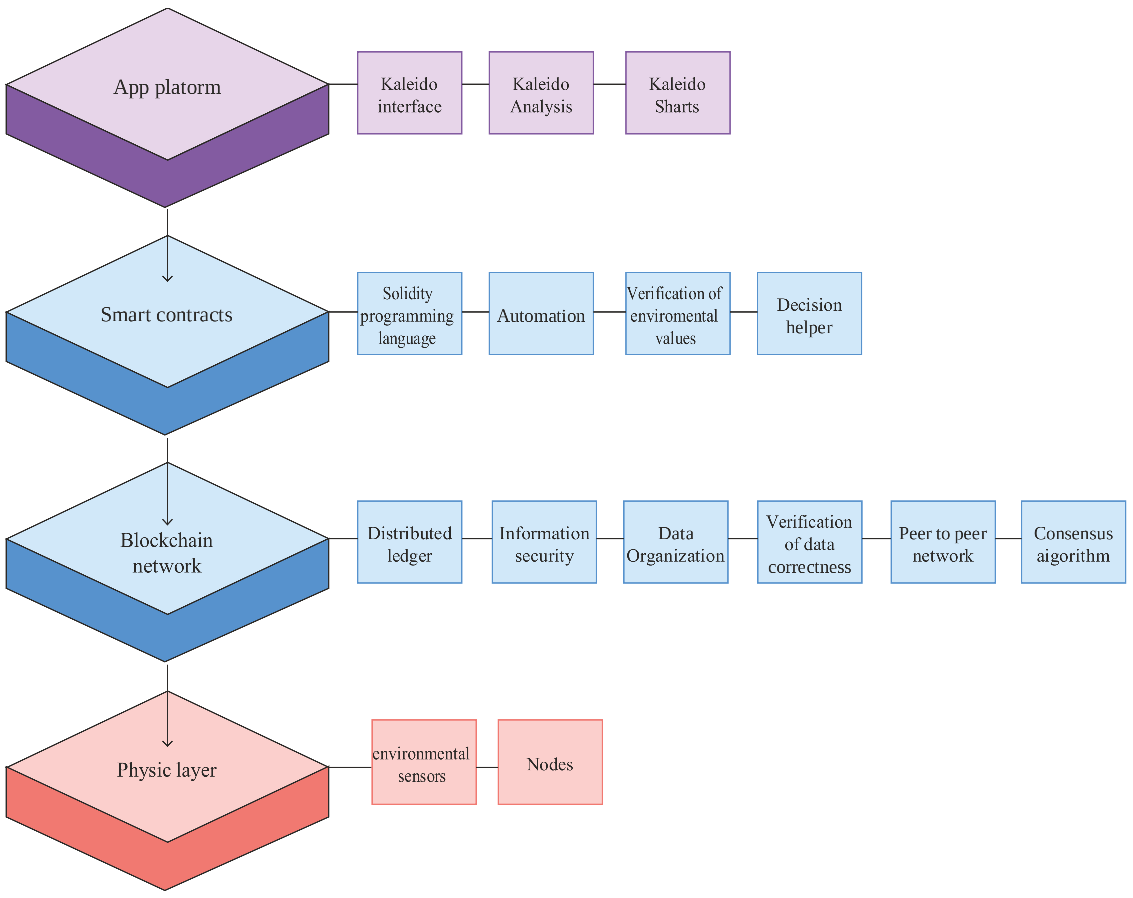


Figure 5: The technical stack of the proposed model.

The research will use a cutting-edge web application built on the foundations of HTML, CSS, JavaScript, and TypeScript to deliver additional functionalities.

Programming languages handle the crucial business logic required by the applications. It is necessary to utilize the Solidity programming language and create source code that will automate various functions, such as identifying the circumstances of crop growth. However, no technology can provide complete 100% security. The blockchain network consistently adheres to best practices to guarantee the highest data and transaction protection level and reduce risks.

Future testing will focus on the new Web 3.0 blockchain technology stack, which differs from earlier iterations. However, transitioning from client-server architecture to a decentralized web will not be radical. The changes are significant and distinct.

Web 3.0 blockchain technology stacks are still developing and maturing step by step. Even though they are safer, they are often a little bit slower. As a result, the transition would initially include building a partially decentralized network before going decentralized.

Testing and Analysis

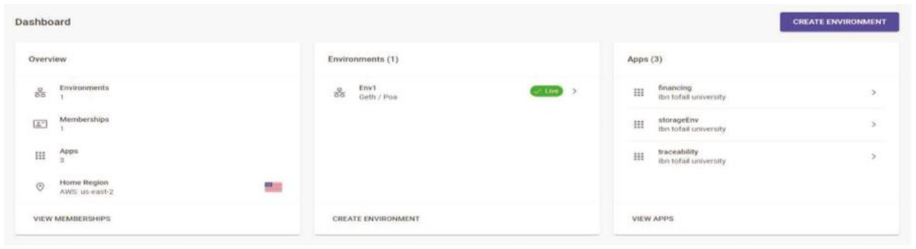
The figures depict the tested blockchain system. The tested architecture contains one environment, one member, and three smart contract script codes (finance, storage environment, and traceability).

Figure 6 depicts the dashboard system, which allows the user to learn about the testing environment. Among these data, we can find information about network participants like the membership name, the membership ID, the primary contact e-mail, and the joining date.

The KALEIDO organization issues a membership certificate with a common name and a serial code. In this situation, the dashboard shows a list of applications directly connecting to our blockchain system, like our smart contract script codes.

Figure 7 shows the current condition of the blockchain system, such as the creation date, block height, network participants, release version, and protocol utilized (Geth/ Poa). We established two nodes (node 1 and node 2) for testing, and one of them is inserted by smart contract codes. There is also the node that has the name system monitor. These nodes have all got started.

The details of the block node injected with the smart contract script are shown in Figure 8. Among these details, we can find the node name (node 1), the node ID, the node size (small), the owner, the status (started or paused), and the AWS region because we have used the Amazon Web Services, along with the date it was formed and the date it was last modified.



Dashboard

CREATE ENVIRONMENT

Overview

Environments
1

Memberships
1

Apps
3

Home Region
AWS: us-east-2

VIEW MEMBERSHIPS

Environments (1)

Env1
Geth / Poa
✓ Live

CREATE ENVIRONMENT

Apps (3)

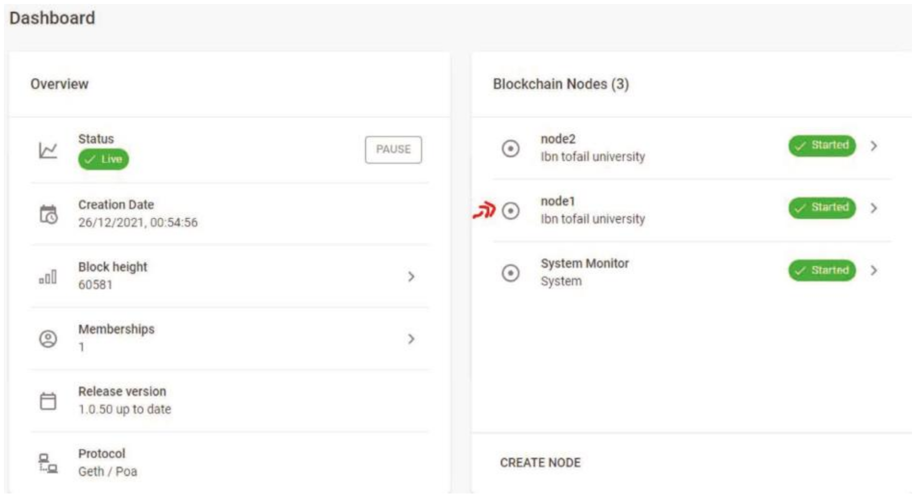
financing
ibn tofail university

storageEnv
ibn tofail university

traceability
ibn tofail university

VIEW APPS

Figure 6: Characteristics of the testing environment.



Dashboard

Overview

Status
✓ Live
PAUSE

Creation Date
26/12/2021, 00:54:56

Block height
60581

Memberships
1

Release version
1.0.50 up to date

Protocol
Geth / Poa

Blockchain Nodes (3)

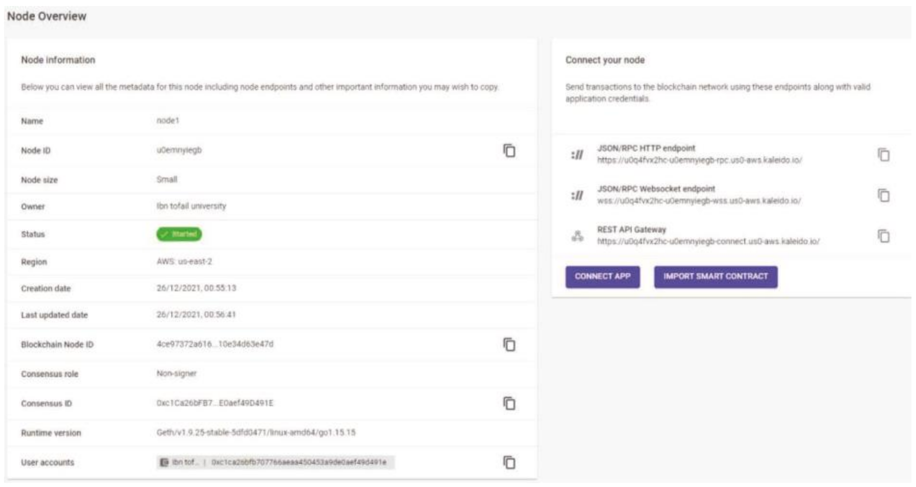
node2
Ibn tofail university
✓ Started

node1
Ibn tofail university
✓ Started

System Monitor
System
✓ Started

CREATE NODE

Figure 7: Implementing the smart contracts on node 1.



Node Overview

Node information

Below you can view all the metadata for this node including node endpoints and other important information you may wish to copy.

Name	node1
Node ID	u0emnyiegb
Node size	Small
Owner	Ibn tofail university
Status	✓ Started
Region	AWS: us-east-2
Creation date	26/12/2021, 00:55:13
Last updated date	26/12/2021, 00:56:41
Blockchain Node ID	4ce97372a616...10e34d63e47d
Consensus role	Non-signer
Consensus ID	0xc1Ca26bFB7...E0aef49D491E
Runtime version	Geth/v1.9.25-stable-5dfd0471/linux-amd64/go1.15.15
User accounts	Ibn tof... | 0xc1ca26bfb707766aeaa450453a9de0aef49d491e

Connect your node

Send transactions to the blockchain network using these endpoints along with valid application credentials.

JSON/RPC HTTP endpoint
https://u0q4fvx2hc-u0emnyiegb-rpc.us0-aws.kaleido.io/

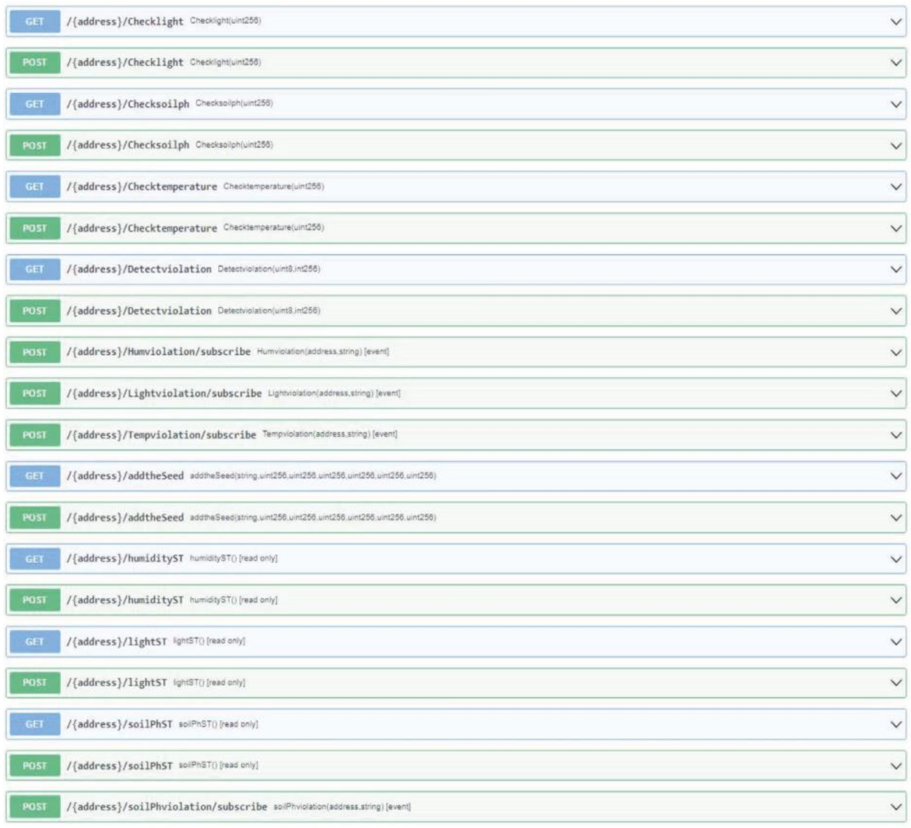
JSON/RPC Websocket endpoint
wss://u0q4fvx2hc-u0emnyiegb-wss.us0-aws.kaleido.io/

REST API Gateway
https://u0q4fvx2hc-u0emnyiegb-connect.us0-aws.kaleido.io/

CONNECT APP

IMPORT SMART CONTRACT

Figure 8: The metadata of node



```
GET   /{address}/Checklight  Checklight(uint256)
POST  /{address}/Checklight  Checklight(uint256)
GET   /{address}/Checksoilph  Checksoilph(uint256)
POST  /{address}/Checksoilph  Checksoilph(uint256)
GET   /{address}/Checktemperature  Checktemperature(uint256)
POST  /{address}/Checktemperature  Checktemperature(uint256)
GET   /{address}/Detectviolation  Detectviolation(uint8,int256)
POST  /{address}/Detectviolation  Detectviolation(uint8,int256)
POST  /{address}/Humviolation/subscribe  Humviolation(address,string) [event]
POST  /{address}/Lightviolation/subscribe  Lightviolation(address,string) [event]
POST  /{address}/Tempviolation/subscribe  Tempviolation(address,string) [event]
GET   /{address}/addtheSeed  addtheSeed(string,uint256,uint256,uint256,uint256,uint256,uint256)
POST  /{address}/addtheSeed  addtheSeed(string,uint256,uint256,uint256,uint256,uint256,uint256)
GET   /{address}/humidityST  humidityST() [read only]
POST  /{address}/humidityST  humidityST() [read only]
GET   /{address}/lightST  lightST() [read only]
POST  /{address}/lightST  lightST() [read only]
GET   /{address}/soilPhST  soilPhST() [read only]
POST  /{address}/soilPhST  soilPhST() [read only]
POST  /{address}/soilPhviolation/subscribe  soilPhviolation(address,string) [event]
```

Figure 10: Some of the functionalities of the smart contract

Figure 12: Several successful block node transactions

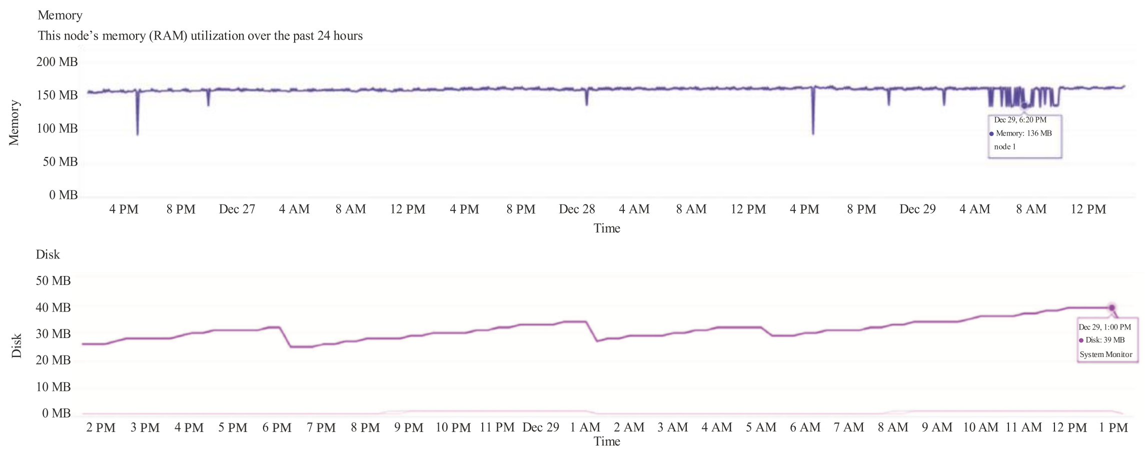
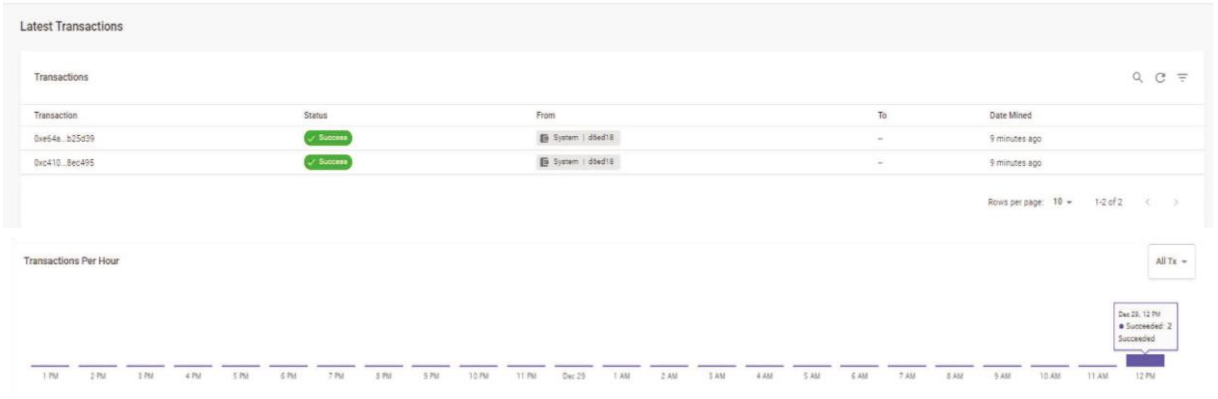


Figure 11: The minor impact of the smart contract script on the machine resources.



Latest Transactions

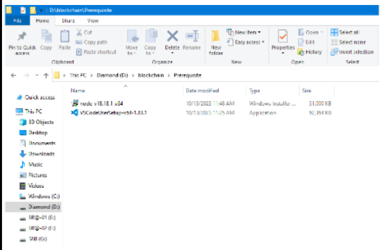
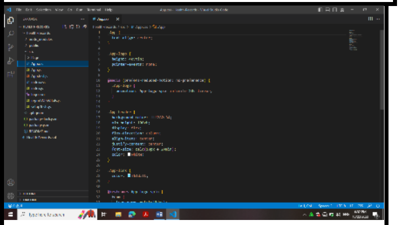
Transactions

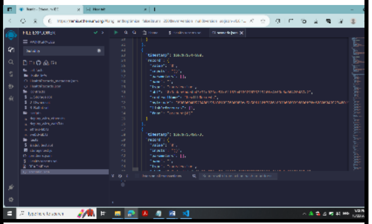
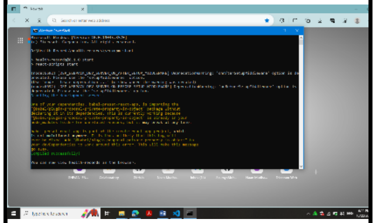
Transaction	Status	From	To	Date Mined
0xe64a...b25d39	✓ Success	System | d6ed18	–	9 minutes ago
0xc410...8ec495	✓ Success	System | d6ed18	–	9 minutes ago

Rows per page: 10 1-2 of 2

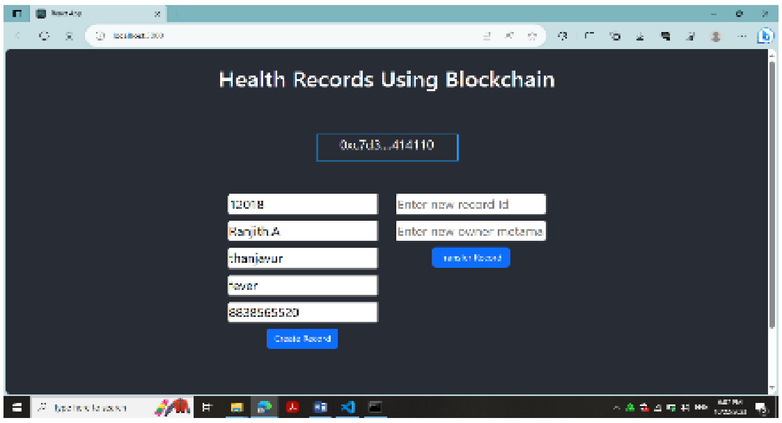
Model Performance Testing:

Project team shall fill the following information when working for blockchain.

S.No.	Parameter	Values	Screenshot
1.	Information gathering	Setup all the Prerequisite:	 D:\blockchain\Prerequisite This PC > Diamond (D:) > blockchain > Prerequisite Name / Date modified / Type / Size node v18.18.1 x64 / 10/13/2023 11:48 AM / Windows Installer ... / 31,000 KB VSCodeUserSetup-x64-1.83.1 / 10/13/2023 11:25 AM / Application / 92,364 KB
2.	Extract the zip files	Open to vs code	

3.	Remix Ide platform exploring	Deploy the smart contract code Deploy and run the transaction. Byselecting the environment -inject the MetaMask.	
4	Open file explorer	Open the extracted file and click onthe folder. Open src, and search for utiles. Open cmd enter commands1.npm install 2. npm bootstrap 3. npm start	
5	{LOCALHOST PADDRESS	copy the address and open it tc chrome so you can see the fron end of your project.	

Results:



Health Records Using Blockchain

0x7d3...414110

12018

Ranjith.A

thanjavur

fever

8838565520

Create Record

Enter new record Id

Enter new owner metamask

Transfer Record

10. Advantages ansd disadvantages

Blockchain technology has several advantages and disadvantages. It's important to understand both sides to make informed decisions about its use. Here's a summary of some of the key advantages and disadvantages of blockchain technology:

Advantages of Blockchain:

1. **Security:**

   - **Immutability:** Once data is recorded on the blockchain, it is extremely difficult to alter, providing a high level of data integrity.

   - **Cryptography:** Strong cryptographic techniques are used to secure transactions, making it highly resistant to fraud and unauthorized access.

2. **Transparency:**

   - **Public Ledgers:** Public blockchains are transparent and allow anyone to view transactions, promoting trust and accountability.

   - **Immutable History:** Historical data is permanently recorded and can be audited, making it useful for various applications like supply chain and voting systems.

3. **Decentralization:**

   - **Removal of Intermediaries:** Blockchain can eliminate the need for intermediaries, reducing costs and increasing efficiency in various processes.

   - **Resilience:** Distributed data is less vulnerable to single points of failure or attacks, making the system more resilient.

4. **Trust and Consensus:**

   - **Decentralized Trust:** Trust is established through consensus mechanisms, reducing the need for centralized authorities.

   - **Permissioned Systems:** In private blockchains, participants are known and trusted, further enhancing trust.

5. **Data Ownership and Control:**

   - **Personal Data Control:** Users can have more control over their data and decide who can access it.

   - **Data Portability:** Data can be easily transferred between applications or services.

6. **Efficiency and Speed:**

   - **Quick Settlement:** Blockchain can facilitate faster settlement of transactions, particularly in financial services.

   - **Reduced Intermediaries:** Eliminating intermediaries can streamline processes and reduce delays.

7. **Cost Reduction:**

   - **Operational Efficiency:** Lowering costs by reducing the need for middlemen and streamlining operations.

   - **Cross-Border Transactions:** Reducing fees associated with cross-border transactions in finance.

Disadvantages of Blockchain:

1. **Scalability:**

   - **Network Congestion:** Public blockchains may suffer from slow transaction processing times and high fees during network congestion.

   - **Scaling Challenges:** Scaling to accommodate a high volume of transactions can be technically challenging.

2. **Energy Consumption:**

   - **Proof of Work (PoW):** PoW blockchains, like Bitcoin and Ethereum, consume substantial energy, which is an environmental concern.

3. **Lack of Regulation:**

   - **Legal and Regulatory Challenges:** The evolving nature of blockchain technology presents legal and regulatory challenges, particularly in relation to cryptocurrencies.

4. **User Experience:**

   - **Complexity:** Blockchain can be complex for non-technical users, making mass adoption a challenge.

   - **Lost Private Keys:** Users risk losing access to their assets if they lose their private keys.

5. **Data Privacy Concerns:**

   - **Public Ledgers:** Public blockchains may expose sensitive information, leading to concerns about data privacy.

   - **Privacy Technology:** Privacy-enhancing technologies like zero-knowledge proofs are still evolving.

6. **Interoperability:**

   - **Isolated Blockchains:** Different blockchains may not easily communicate or interoperate, limiting their potential for widespread use.

7. **Security Risks:**

   - **Smart Contract Vulnerabilities:** Smart contracts can have coding vulnerabilities that, if exploited, may lead to financial losses.

   - **51% Attacks:** In PoW blockchains, a malicious actor with majority computational power could compromise the network.

8. **Legal and Ethical Issues:**

- **Use in Illicit Activities:** Blockchain technology can be used for illegal activities, such as money laundering and black-market transactions.

The advantages and disadvantages of blockchain technology vary depending on the use case, blockchain type, and the specific implementation. It's important to carefully consider these factors when evaluating whether blockchain is the right solution for a particular application or organization.

11.Conclusion and Future work:

In conclusion, blockchain and IoT technologies can aid in developing a secure, transparent, open, and innovative ecological agriculture system that involves all participants.

This work aims to provide a possible technique to build practical blockchain-based applications and change the agriculture industry, even though the evolution of blockchain and agriculture research studies is still in its infancy. This model is considered a prototype for reducing financial loss and agricultural pollution. The system defines the three primary entities in the agriculture domain: data, process, and stakeholders. Adding a cryptocurrency process for the interaction between the entities and registering/tracking the seller’s land will be a big step for this blockchain system model.

12.Apendix :

Source code:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Agriculture Registry
{
    struct food Product
{
        string name;
        string description;
        uint256 quantity;
        address owner;
    }

    mapping(uint256 => foodProduct) public products;
    uint256 public productCount;

    event ProductAdded(uint256 productId, string name, string description,
uint256 quantity, address owner);
    event ProductUpdated(uint256 productId, string name, string description,
uint256 quantity);

    modifier onlyOwner(uint256 _productId) {
        require(products[_productId].owner == msg.sender, "Only the owner can
perform this action");
        _;
    }

    function addProduct(uint256 ProductId, string memory _name, string memory
_description, uint256 _quantity) external {

        products[ProductId] = foodProduct(_name, _description, _quantity,
msg.sender);
        productCount++;
        emit ProductAdded(productCount, _name, _description, _quantity,
msg.sender);
    }

    function updateProduct(uint256 _productId, string memory _name, string
memory _description, uint256 _quantity) external onlyOwner(_productId) {
        foodProduct storage product = products[_productId];
        product.name = _name;
        product.description = _description;
        product.quantity = _quantity;
        emit ProductUpdated(_productId, _name, _description, _quantity);
    }

    function getProductDetails(uint256 _productId) external view returns
(string memory name, string memory description, uint256 quantity, address
owner) {
        foodProduct memory product = products[_productId];
        return (product.name, product.description, product.quantity,
product.owner);
    }
}
```