Project Report

Team ID	**NM2023TMID03272** **EC6F7CB994F1CA2BEC94EBCE7F892D3D**
Project Name	Farmer Insurance Chain

Submitted By,

TEAM LEADER : SUBASHINI P
TEAM MEMBER 1 : JEEVITHA M
TEAM MEMBER 2 : SANKARI M
TEAM MEMBER 3 : GAYATHRI K

PROJECT REPORT FORMAT

1. **INTRODUCTION**
   - Project Overview
   - Purpose
2. **LITERATURE SURVEY**
   - Existing problem
   - References
   - Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
   - Empathy Map Canvas
   - Ideation & Brainstorming
4. **REQUIREMENT ANALYSIS**
   - Functional requirement
   - Non-Functional requirements
5. **PROJECT DESIGN**
   - Data Flow Diagrams &User Stories
   - Solution Architecture
6. **PROJECT PLANNING & SCHEDULING**
   - Technical Architecture
   - Sprint Planning & Estimation
   - Sprint Delivery Schedule
7. **CODING & SOLUTIONING**
   - Feature 1
   - Feature 2
   - Database Schema (if Applicable)
8. **PERFORMANCE TESTING**
   - Performace Metrics
9. **RESULTS**
   - Output Screenshots
10. **ADVANTAGES & DISADVANTAGES**
11. **CONCLUSION**
12. **FUTURE SCOPE**
13. **APPENDIX**

Source Code

GitHub & Project Demo Link

1. INTRODUCTION

The "Former Insurance Chain Ethereum Blockchain" Agriculture is risky. Indeed, the exposure to a wide variety, complexity, and scale of risks can make it one of those rare activities where the risks are too high and the rewards too low, especially for smallholder farm. Block chain is a ledger of accounts and transactions that are written and stored by all participants in the agriculture supply chain. Most block chains are designed as a decentralized database that functions as a distributed digital ledger.

PROJECT OVER VIEW:

Block chain technology in agricultural food crop insurance reduces costs, increases efficiency, and improves transparency and trust between insurers and farmers. Block chain technologies can track all kinds of information regarding plants, including the quality of the seed, how crops grow, and even create a record of a plant's journey once it leaves the farm. only 20% of smallholder farmers in the developing countries have access to agricultural insurance coverage, and in sub-Saharan Africa this falls further to just 3%. Block chain is a transformative Information and Communications Technology (ICT) that have the potential to revolutionize how data is used for agriculture insurance and enable smallholder-inclusive value chain development.

Consider a modern insurance company: It uses centralized electronic ledgers to track transactions (premium collected and claims paid) and store data such as the policyholders' personal information (e.g. the insured's name, address and national id number), risk-related information (e.g. type of crop grown, location, loss history) and policy terms (e.g. risk coverage, deductible and policy limits) in its internal system. With the advancement of technology, new data such as those from satellite, sensors, mobile phone, other digital platforms, and smart networks are seamlessly fed into insurer's in-house data and systems, providing richer, near real-time update of exposure.

Purpose

To provide insurance coverage and financial support to the farmers in the event of failure of any of the notified crop as a result of natural calamities, pests & diseases. 2. To encourage the farmers to adopt progressive farming practices, high value inputs and higher technology in Agriculture. It aims to reduce the financial risk and uncertainty faced by farmers and help them manage their production and income more effectively.

It provides damage protection to standing and post-harvest crops. Like car insurance provides coverage against damage caused because of accidents, theft, vandalism, and natural disasters, crop insurance provides coverage against crop damages happening due to: Natural disasters like drought, hailstorms, floods, and rain.

2. LITERATURE SURVEY

1.Blockchain for Farmer Insurance Chain

For programs aiming to be gender inclusive, an explicit objective is to increase access and uptake of agricultural insurance products among both women and men. Recent literature has highlighted the genderbased disparities in access and usage of agricultural insurance products. Field experiments conducted in Senegal and Burkina-Faso show that insurance demand is much stronger among men than women (Delavallade et al. 2015). Similar results were obtained through experiments in southern Asia (Clarke and Kumar, 2016; Akter et al. 2016). In Ethiopia, although insurance is equally accessed by both men and women, women are likely to purchase lower value coverage (Bageant and Barrett, 2017). These studies attribute low access and usage of agricultural insurance among women farmers to both demand and supply-side barriers

2.A Blockchain-Based Approach for Farmer Insurance Chain

Recently, the government plans to use the technology across all export-driven crops to increase the country's food shipments and incentivise farmers to take up chemical-free processes. India's natural farming could soon get a technological push through blockchain. Smart contracts on the blockchain can automate transactions, payments, and other processes, reducing the need for intermediaries and cutting transaction costs. For example, blockchain can enable farmers to sell their crops directly to consumers or retailers, bypassing traditional middlemen and reducing costs.Blockchain has also been recognised as an important tool in climate action in agriculture, fostering registration of carbon foot emissions, the development of trust among actors that use more sustainable solutions in their work and providing the basis for carbon labelling and certification (FAO, 2021).

3. A semantic blockchain-based system for drug traceability

In addition, smart contracts help prevent fraud and compliance breaches and improve data security by providing immutable records of insurance transactions and events. Weather Based Crop Insurance aims to mitigate the hardship of the insured farmers against the likelihood of financial loss on account of anticipated crop loss resulting from incidence of adverse conditions of weather parameters like rainfall, temperature, frost, humidity etc. All farmers including sharecroppers and tenant farmers growing the notified crops in the notified areas are eligible for coverage. However, farmers should have insurable interest for the notified/insured crops.This policy covers the loss of life due to accident, disease, or surgical operation. The death and/or the permanent disablement of the insured due to an accident. Any loss or damage happening as a result of radioactive contamination. War or a war-like situation leading to losses to a farmer.

Existing problem

To ensure that its insurance program is equally accessible for women as for men, ACRE Africa uses village extension service providers (VESPs), basically community-based champion farmers, to provide insurance education, enroll farmers into the program, and market and distribute the insurance products. VESPs are equipped with smartphones and trained on how to use them. They are also trained on how to sell and distribute agricultural insurance products. Using extension workers drawn from the local community and partnering with local service providers (such as agrovets) is a gender-inclusive approach by improving trust and accessibility of insurance products for both women and men. In addition, using mobile-phones to purchase insurance cover and receive payouts can help overcome mobility and time burden-related barriers that women face in accessing insurance To analyze the extent to which this is indeed a gender-inclusive approach, we report data from an experiment that was conducted to elicit farmers' willingness to pay (WTP) for a variety of products, varying whether insurance payouts are made into one's own account ('self"), versus a spouse's account ('spouse'). The results indicate that overall, women's WTP was higher than men's WTP

Reference

Wikipedia Contributors. Farmers' Suicides in India. In Wikipedia, The Free Encyclopedia. Available online: https://en.wikipedia.org/w/index.php?title=Farmers%27_suicides_in_India&oldid=1030026013 (accessed on 23 June 2021).

2. Blog, R.; Concern, F. Farmer's Suicides-An Issue of Great Concern. Available online: https://timesofindia.indiatimes.com/read-ersblog/hail-to-feminism/farmers-suicides-an-issue-of-great-concern-27472/ (accessed on 23 June 2021).

3. Crop Insurance Schemes in India: Need, Importance and Benefits to Farmers. Available online: https://krishijagran.com/crop-in-surance-schemes-in-india-need-importance-and-benefits-to-farmers/ (accessed on 23 June 2021).

4. Scherer, M. Performance and Scalability of Blockchain Networks and Smart Contracts; Diva, Springer Umea University: Umeå, Sweden, 2017.

5. Rahim, S.R.M.; Mohamad, Z.Z.; Abu Bakar, J.; Mohsin, F.H.; Isa, N.M. Artificial Intelligence, Smart Contract and Islamic Finance. Asian Soc. Sci. 2018, 14, 145, doi:10.5539/ass.v14n2p145.

6. Nguyen, T.; Das, A.; Tran, L.; Tq, N.; Ak, D.; Lt, T. NEO Smart Contract for Drought-Based Insurance. In Proceedings of the 2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE), Edmonton, AB, Canada, 5–8 May 2019; pp. 1–4.

7. Iyer, V.; Shah, K.; Rane, S.; Shankarmani, R. Decentralised Peer-to-Peer Crop Insurance. In Proceedings of the Proceedings of the 3rd ACM International Symposium on Blockchain and Secure Critical Infrastructure; Virtual Event, Hong Kong, China, 7–11 June 2021; pp. 3–12.

8. Kevorchian, C.; Gavrilescu, C.; Hurduzeu, G. A Peer-to-Peer (p2p) Agricultural Insurance Approach Based on Smart Contracts in Blockchain Ethereum. In Agricultural Economics and Rural Development; New Series, Year XVII; 2020; pp. 29–45. Available online: ftp://www.ipe.ro/RePEc/iag/iag_pdf/AERD2001_29-45.pdf (accessed on 20 July 2021).

9. Sharifinejad, M.; Dorri, A.; Rezazadeh, J. BIS-A Blockchain-based Solution for the Insurance Industry in Smart Cities. arXiv 2020, arXiv:2001.05273.

10. Gera, J.; Palakayala, A.R.; Rejeti, V.K.K.; Anusha, T. Blockchain Technology for Fraudulent Practices in Insurance Claim Process. In Proceedings of the 2020 5th International Conference.

Problem Statement Definition

1. LIMITED INFRASTRUCTURE AND PRODUCTION CAPABILITIES:

One of the main disadvantages of blockchain is its high energy consumption. The technology is very uses a lot of energy , in order to keep a real-time ledger. Every time a new transaction is made, more energy needs to be used to solve it.Allow primary insurers, reinsurers, brokers, and regulators to share data securely in real-time. Automate risk modeling, audits, and compliance checks. Bind towers of risk and treaties on a single time-stamped smart contract. Blockchain technology or blockchain in agriculture can track all types of information about plants, such as seed quality, crop growth, and even the travel of a plant after it leaves the farm. This data can improve supply chain transparency and eliminate concerns associated with illegal and unethical operations.

2. AMBIGUOUS REGULATIONS

The lack of financial resources affects not only productivity but also affects the quality of agricultural produce. Farmers in some developing countries do not have access to adequate funds to invest in better technologies, machinery and equipment which results in poor-quality agricultural produce.The central government promulgated three Ordinances on June 5, 2020: (i) the Farmers' Produce Trade and Commerce (Promotion and Facilitation) Ordinance, 2020, (ii) the Farmers (Empowerment and Protection) Agreement on Price Assurance and Farm Services Ordinance, 2020, and (iii) the Essential Commodities (Amendment).

3. INSUFFICIENT TECHNO-FUNCTIONAL RESOURCES FOR IMPLEMENTING AND SUSTAINING Former Insurance Chain

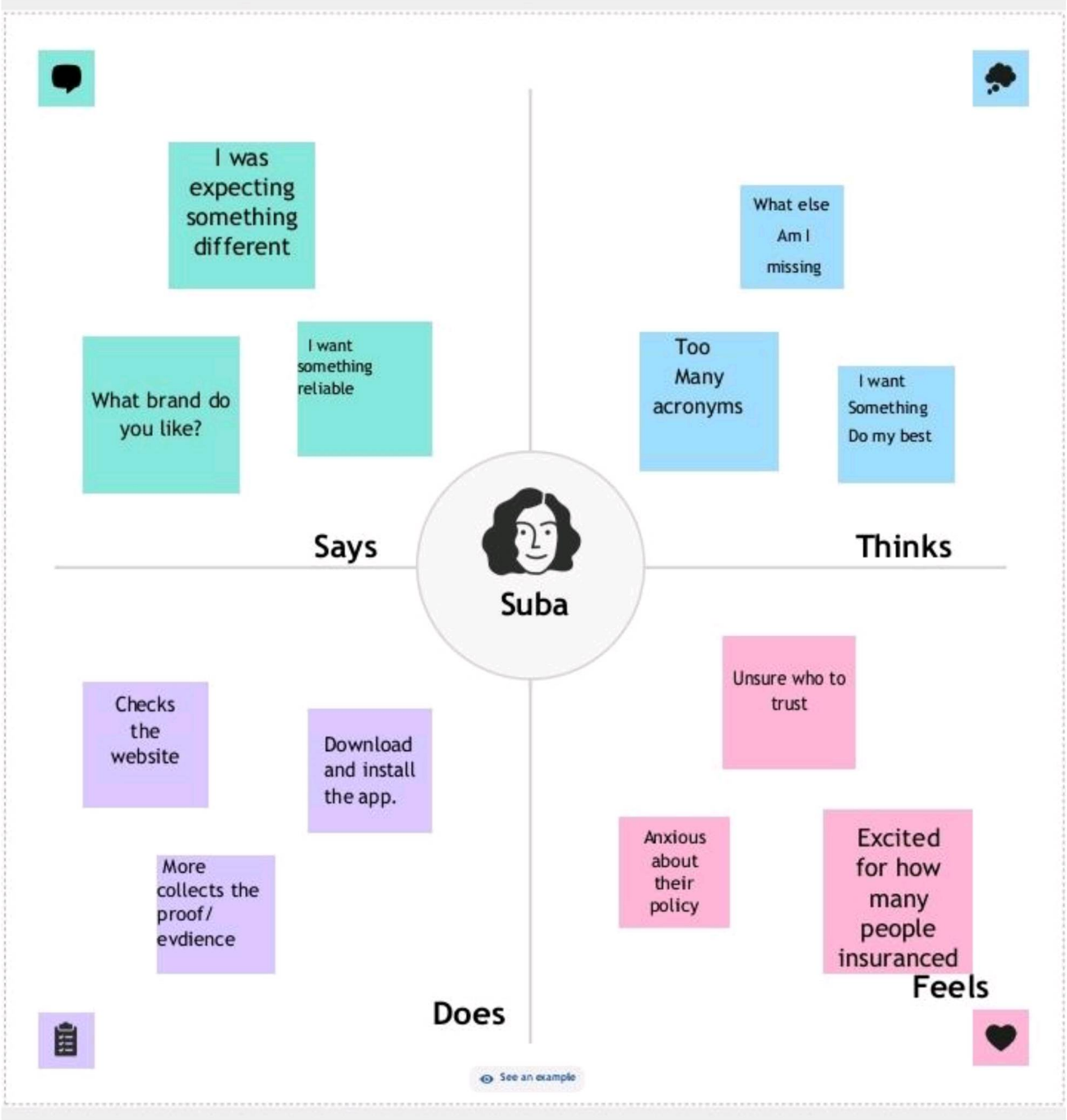
One of the key challenges of blockchain technology implementation in agriculture is that its benefits might be dependent on the size of the farm. For example, smaller farms can easily participate in blockchain-based insurance. The current blockchain infrastructure has limitations in terms of the number of transactions it can process and the amount of data it can store. This becomes a significant challenge for the insurance industry, which deals with large volumes of data on a daily basis.

4. UNSECURE AND UNRELIABLE TECHNICAL INFRASTRUCTURE FOR DIGITAL FARMER INSURANCE CHAIN

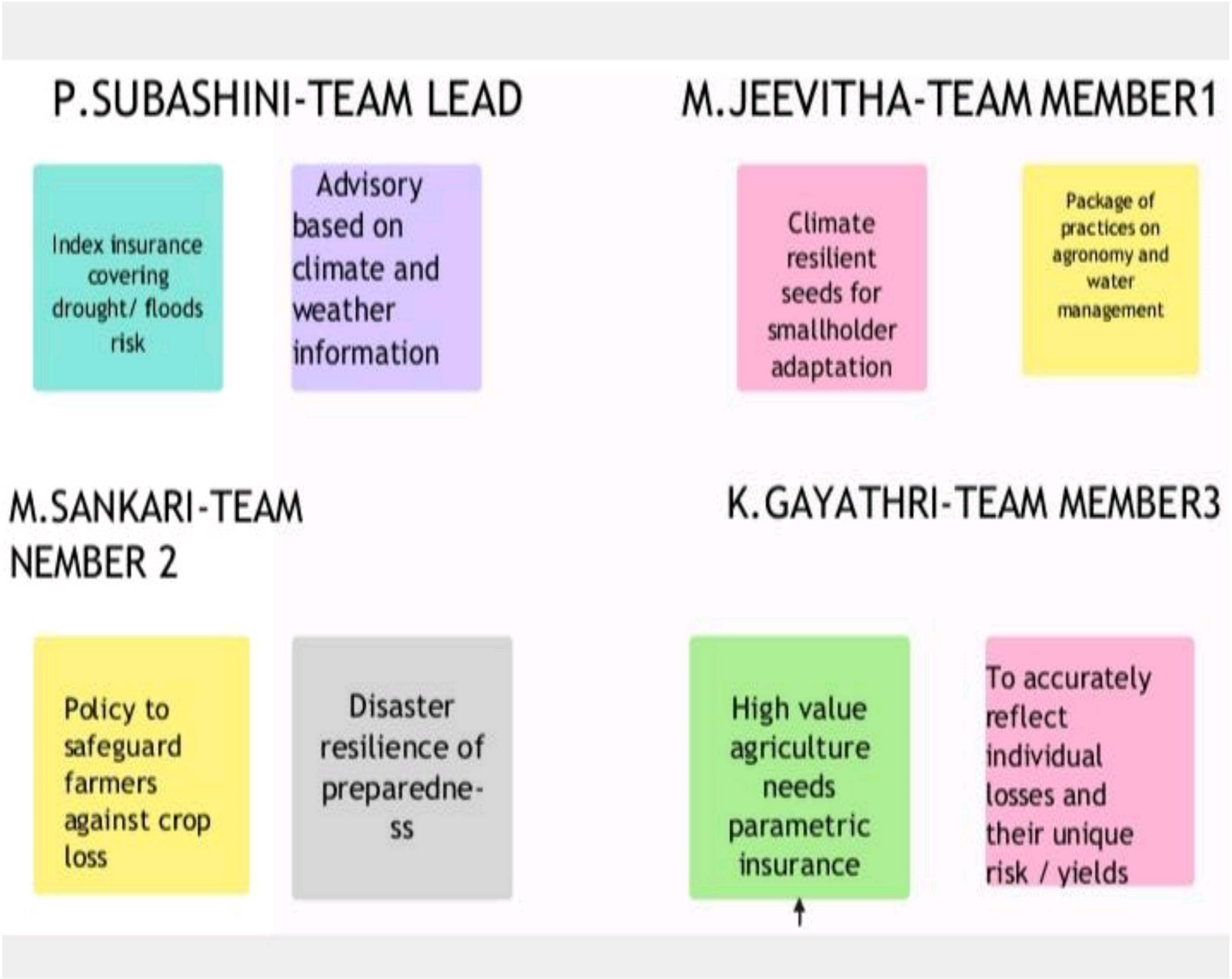
In an agri-food supply chain, blockchain technology creates transparency and stores data for all production and processing activities. Simultaneously, it guarantees compliance with agri-food product standards and improvements in sustainabilityWhile the potential accuracies of digital farming technologies indeed present significant challenges around questions such as data sovereignty, unequal access, and mounting technology input costs and debt, we argue that equally (or even more) important risks of data-driven solution.

IDEATION & PROPOSED SOLUTION

4Empathy Map Canvas



Ideation & Brainstorming



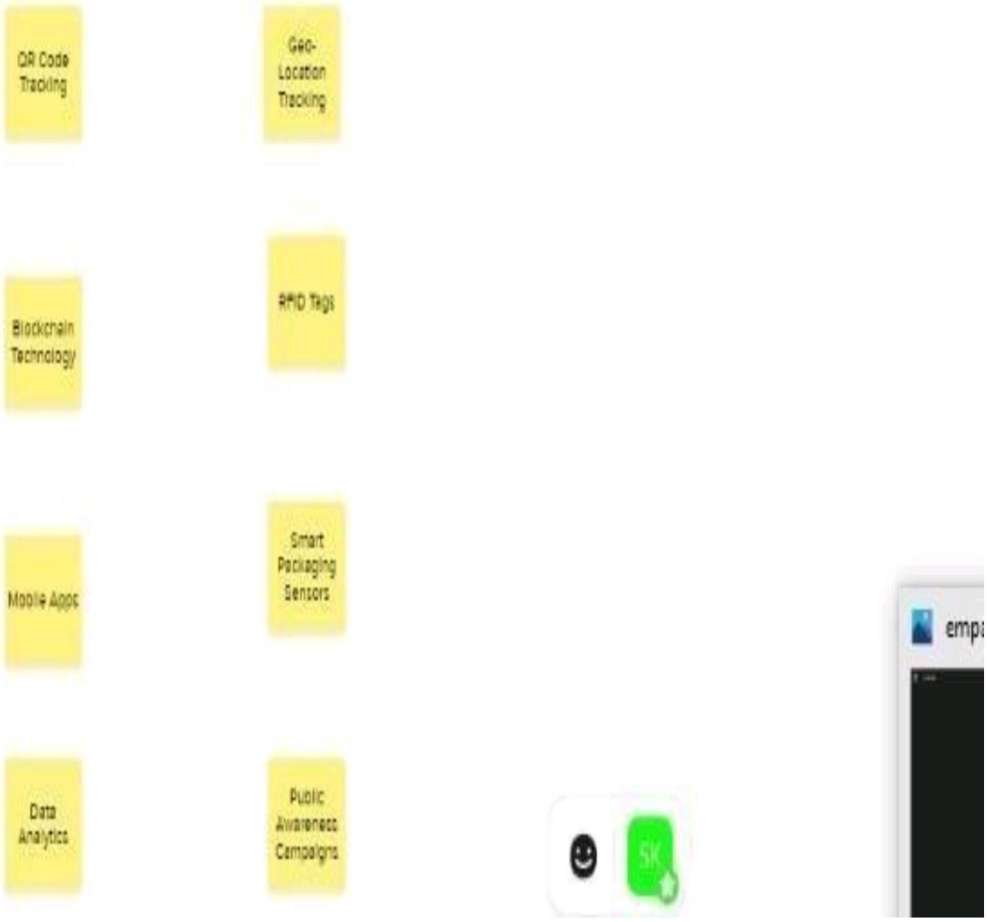
Group ideas

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.



Ideation & Brainstorming

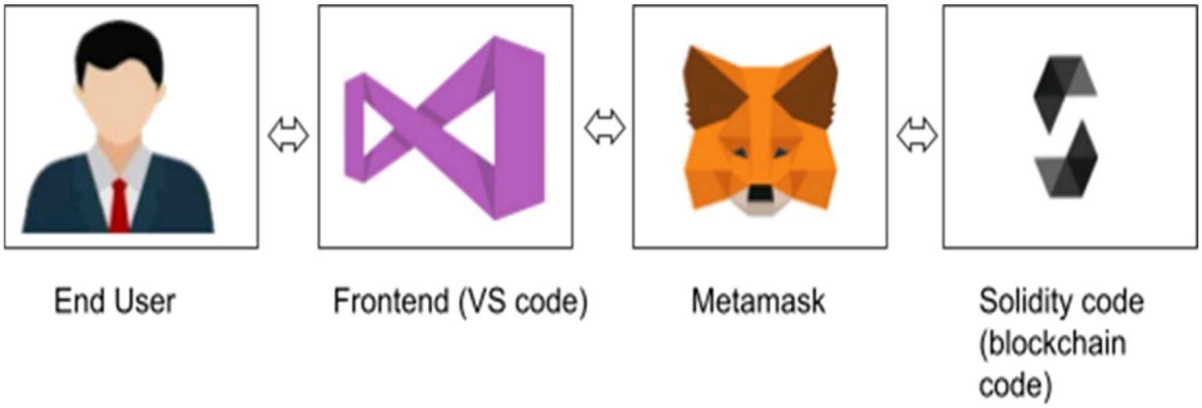
FR No.	Functional Requirement	Description
FR- 1	Define aspiration and set vision	Define aspirational goal and strategic use cases for data as farming insurance of block chain technology.
FR- 2	Shared economy insurance	e growing popularity of platforms based on sharing economy makes it imperative to provide real-time insurance coverage to the renters who book on this platform.
FR- 3	Insurance farmers importance	The policy meant to compensate farmers losses arising due toproduction problems,these losses lead to reduction in crop yield affecting the income of farmers.
FR- 4	Subrogation in claim	While the current subrogation is highly manual in nature, insurers can leverage smart contracts to automate the processes of claim notifi-cation, loss investigation and recoveries from other insurers and reinsurers.
FR- 5	Automatic claims settlement	The loss notification for an accident is trigged by the call center, smartphone app or connected car via the internet to the Blockchain and activates the smart contract.
FR- 6	Smart Contracts:	Utilize smart contracts for managing farmer insurance , transfers, and permissions. Implement contract functionality for executing predefined rules and logic.
FR- 7	Reinsurance in insurance	This entails recording details of claims so that insurers and reinsurers can accurately share policy contracts, risk sharing and loss information costs between them.
FR- 8	Group life insurance	Group life insurance policy involves employer as the owner of the policy with the employee as beneficiary. Therefore, all the changes are required to be routed through the employer to the insurer.
FR- 9	Workers' compensation (WC)	Insurers are facing huge number of claim frauds through workers' compensation. Blockchain technology allows sharing of information between multiple parties.
FR- 10	Smart homes and connected device	Blockchain in IoT can solve the trust issues businesses face when building smart devices that can communicate and operate autonomously.

Non-Functional requirements

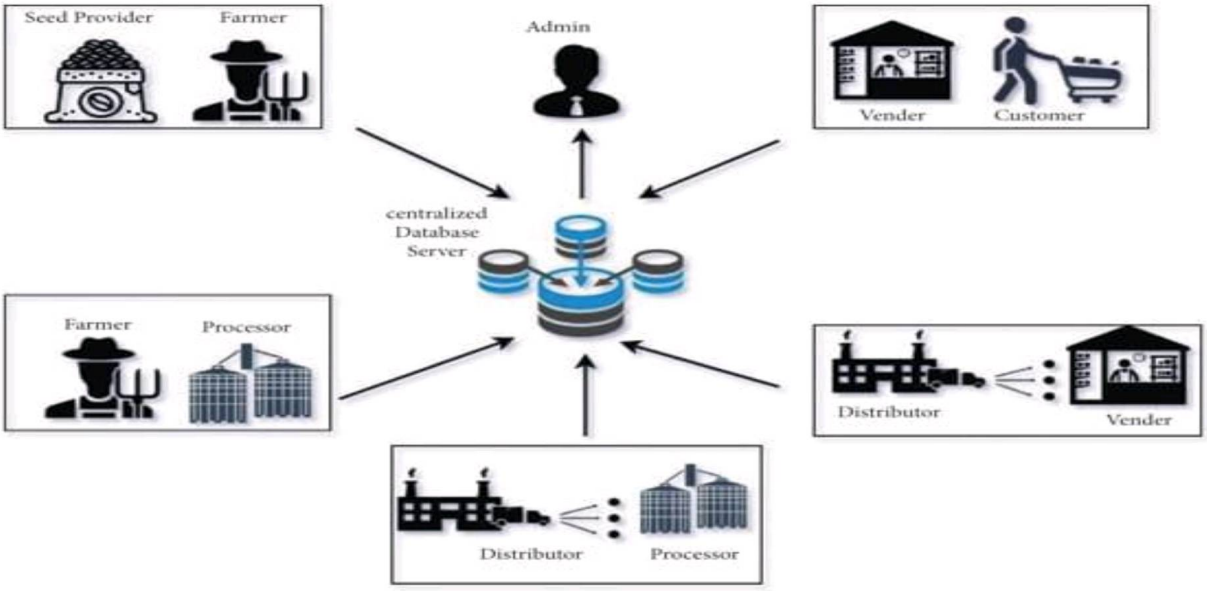
NFR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Blockchain can: Automate most or all parts of parametric insurance. Embed a policy's logic in a smart contract and let an oracle (digital feed) trigger execution upon a predefined loss event. Settle and clear all transactions without manual intervention.
NFR-2	**Security**	Blockchain technology or blockchain in agriculture can track all types of information about plants, such as seed quality, crop growth, and even the travel of a plant after it leaves the farm.
NFR-3	**Reliability**	Blockchain technology offers a more transparent and secure system that can reduce fraud and improve accountability. Additionally, blockchain-based systems can provide farmers with more efficient and timely access to insurance services, which can be crucial in the event of crop loss
NFR-4	**Performance**	Blockchain technology or blockchain in agriculture can track all types of information about plants, such as seed quality, crop growth, and even the travel of a plant after it leaves the farm.
NFR-5	**Availability**	AgriDigital: AgriDigital provides blockchain solutions for the agriculture sector, helping to simplify grain storage, logistics, and transactions
NFR-6	**Scalability**	One of the main disadvantages of blockchain is its high energy consumption. The technology is very uses a lot of energy , in order to keep a real-time ledger. Every time a new transaction is made, more energy needs to be used to solve it.

3. PROJECT DESIGN

Data Flow Diagrams & User Stories



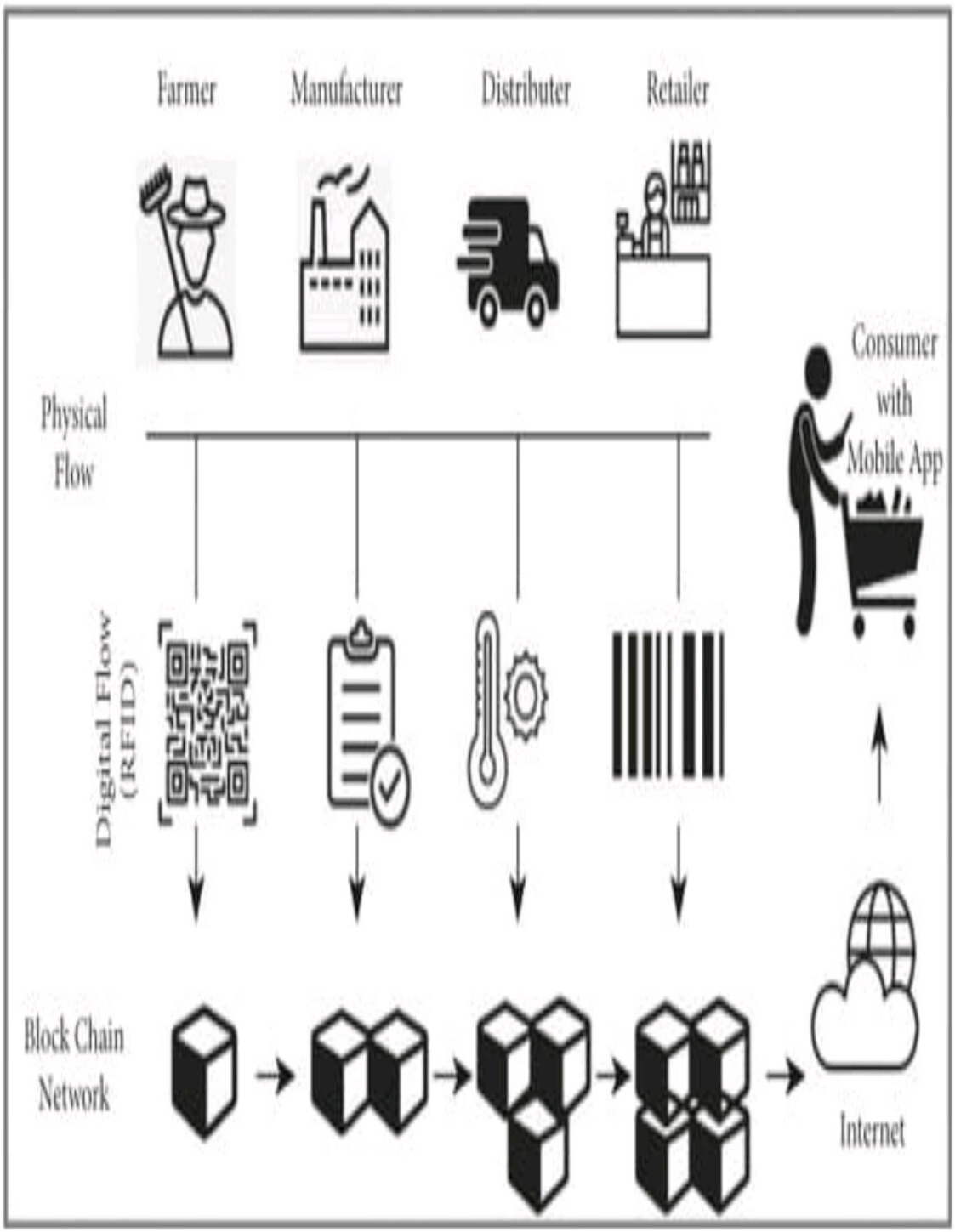
Data Flow Diagrams



User Stories

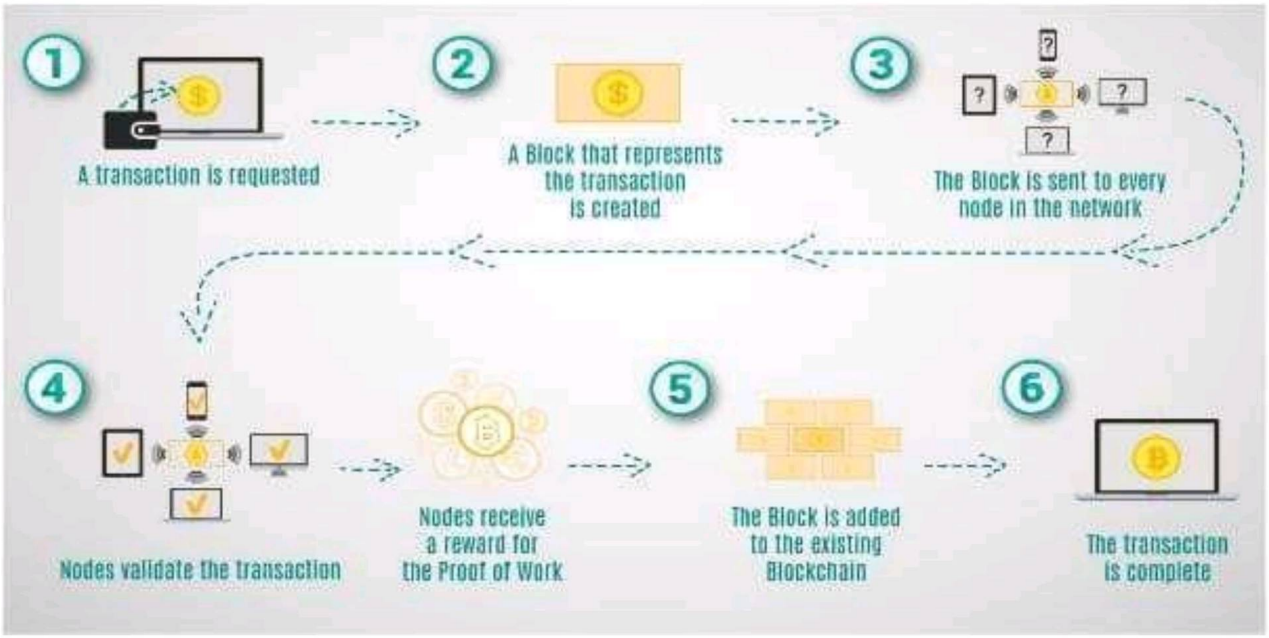
User Type	Functional Requireme nt (Epic)	Story Number	User Story / Task	Acceptance criteria	Priority	team Member
Content Creator	Insurance Management	USN-1	As a content creator, I want to upload multiple image and video assets to the farmer with insurance - and-drop functionality	Users should be able to insurance and drop multiple assets onto the land traceability nterface, and the system should process and upload them efficiently	High	Subashini
Content Creator	Insurance Management	USN-2	As a content creator, I need to add detailed metadata to my crop, including titles, descriptions, and copyright information, to keep them wellorganized	Metadata fields should be easily accessible and editable, and changes should be immediately reflected in farmer detail information	Medium	Jeevitha
Marketing Manager	Farmer Organizationand Access Control	USN-3	As a marketing manager, I want to create and assign tags to ID for easy categorization, facilitating efficient farmer retrieval.	Tags should be customizable, and land use shouldbe sortable and filterable by assigned tags	Medium	Sankari
Marketing Manager	Insurance Organization and Access Control	USN-4	As a marketing manager, I need to restrict access to confidential agri product to authorized team members only	Access control settings should allow me to specify who can view, edit, and farming lands , with permissions easily adjustable	High	Gayathri
Administra tor	System Management	USN-5	As an administrator, I want to monitor and manage farmer access, user roles, and system performance.	The admin dashboard should provide insights into user activity, allow role assignments, and offer system performance metrics	High	Subashini
Administra tor	System Management	USN-6	As an administrator, I need to set up automated data backups and a disaster recovery plan for data safety.disciplie	The system should regularly back up data and provide a documented recovery plan to prevent data loss.	High	Jeevitha

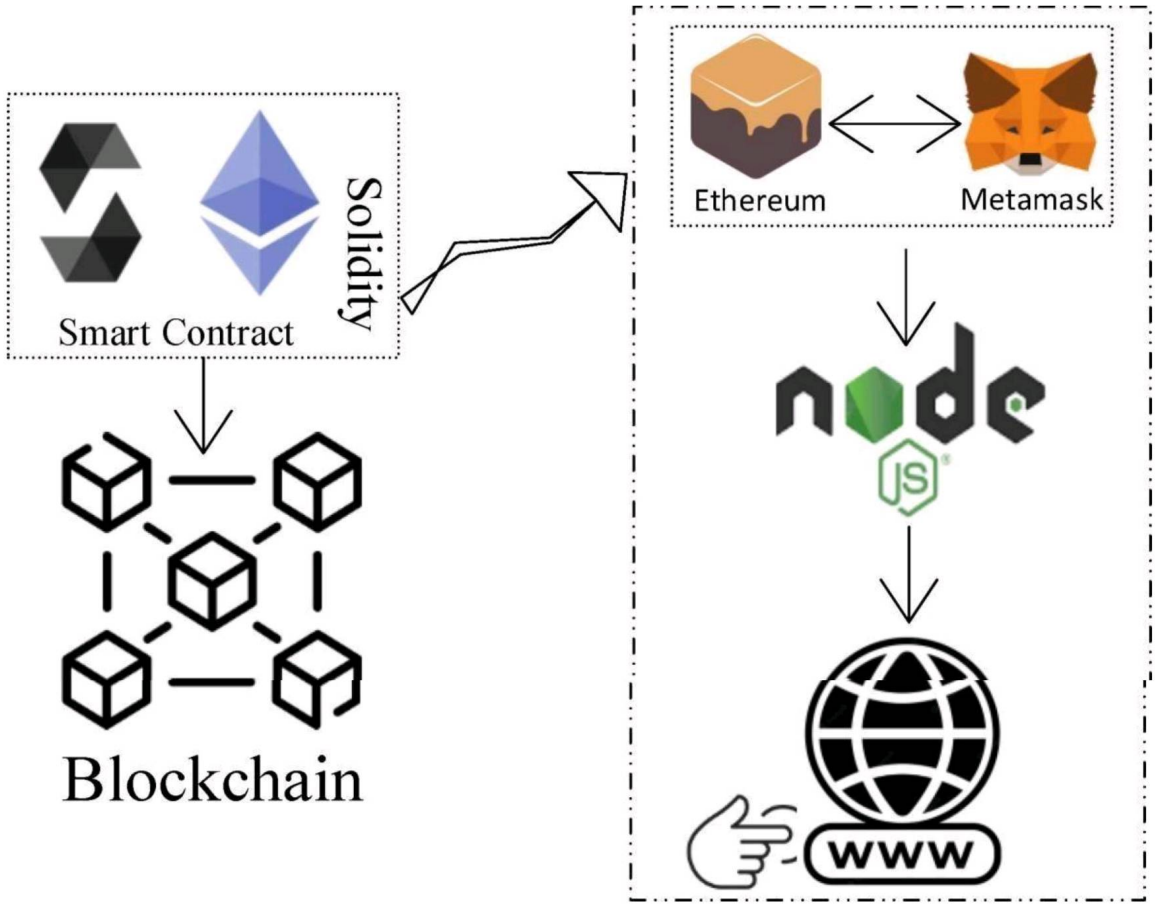
Solution Architecture



4. PROJECT PLANNING & SCHEDULING

Technical Architecture





Sprint Planning & Estimation

1. User Story Backlog:

Start by creating a backlog of user stories. These stories should represent the features, enhancements, or tasks needed for your drug traceability. Ensure they are well-defined, with clear acceptance criteria.

2. Prioritization:

Collaborate with stakeholders to prioritize the user stories based on their importance and impact. High-priority stories should be at the top of the backlog.

3. Sprint Planning Meeting:

Hold a sprint planning meeting with your development team. During this meeting, select a set of user stories from the backlog to work on during the upcoming sprint. Consider the team's capacity and the complexity of the stories.

4. Story Point Estimation:

Use a method like story point estimation to estimate the effort required for each user story. The team assigns relative points to stories to indicate their complexity. This helps in determining how many stories can be included in the sprint.

5. Sprint Goal:

Define a clear sprint goal, which should align with the project's objectives. The goal should provide a sense of purpose for the sprint.

6. Daily Stand-Ups:

Conduct daily stand-up meetings to keep the team updated on progress, discuss any challenges, and make necessary adjustments to the sprint plan.

7. Sprint Review:

At the end of the sprint, hold a sprint review meeting to showcase the completed work to stakeholders. Gather their feedback and insights.

8. Sprint Retrospective:

After the review, conduct a sprint retrospective to assess what went well and what could be improved. Use this feedback to make process enhancements for the next sprint.

9. Continuous Improvement:

Agile principles emphasize continuous improvement. Apply lessons learned from

each sprint to refine the process, including better estimation and planning.

10. Blockchain Integration Considerations:

When estimating and planning, consider the complexities related to blockchain integration, such as smart contract development, security measures, and the use of Ethereum's capabilities.

Sprint Delivery Schedule

1. Divide the Project into Sprints:

Begin by dividing the overall drug traceability project into sprints. Sprints are time-bound iterations, usually lasting 2-4 weeks, during which specific sets of features or tasks are completed.

2. Prioritize User Stories:

Review the prioritized user stories from your backlog and select those that will be addressed in each sprint. Ensure that each sprint has a clear focus and goal.

3. Define Sprint Durations:

Decide on the duration of each sprint. Agile sprints are typically 2-4 weeks long, but you can choose the duration that works best for your team and project.

3. Create a Sprint Backlog:

For each sprint, create a sprint backlog that includes the user stories, tasks, and features that will be tackled during that sprint.

4. Assign Story Points:

Estimate the effort required for each user story in the sprint backlog using story points or other estimation methods. This helps in understanding the capacity of the sprint.

5. Distribute Workload:

Based on the team's capacity and story point estimates, distribute the workload evenly across the sprint backlog items. Ensure that the team can realistically complete the planned work during the sprint.

6. Define Milestones:

Within each sprint, set specific milestones or checkpoints for key tasks or features. This helps in tracking progress and ensuring that the team is on target.

7. Adjust for Blockchain Integration:

Consider the complexities of blockchain integration in your delivery schedule. Tasks related to smart contract development, security testing, and Ethereumspecific considerations should be accounted for.

8. Iterative Development:

Remember that in Agile development, work is delivered incrementally. At the end of each sprint, you should have a potentially shippable product increment.

9. Continuous Review and Adaptation:

After each sprint, hold sprint reviews and retrospectives to gather feedback, evaluate progress, and make necessary adjustments to the delivery schedule or project priorities.

10. Release Planning:

Based on the progress in each sprint and the feedback received, plan releases of the drug traceability. These releases can be scheduled according to the completion of major features or project milestones.

11. Maintain a Release Calendar:

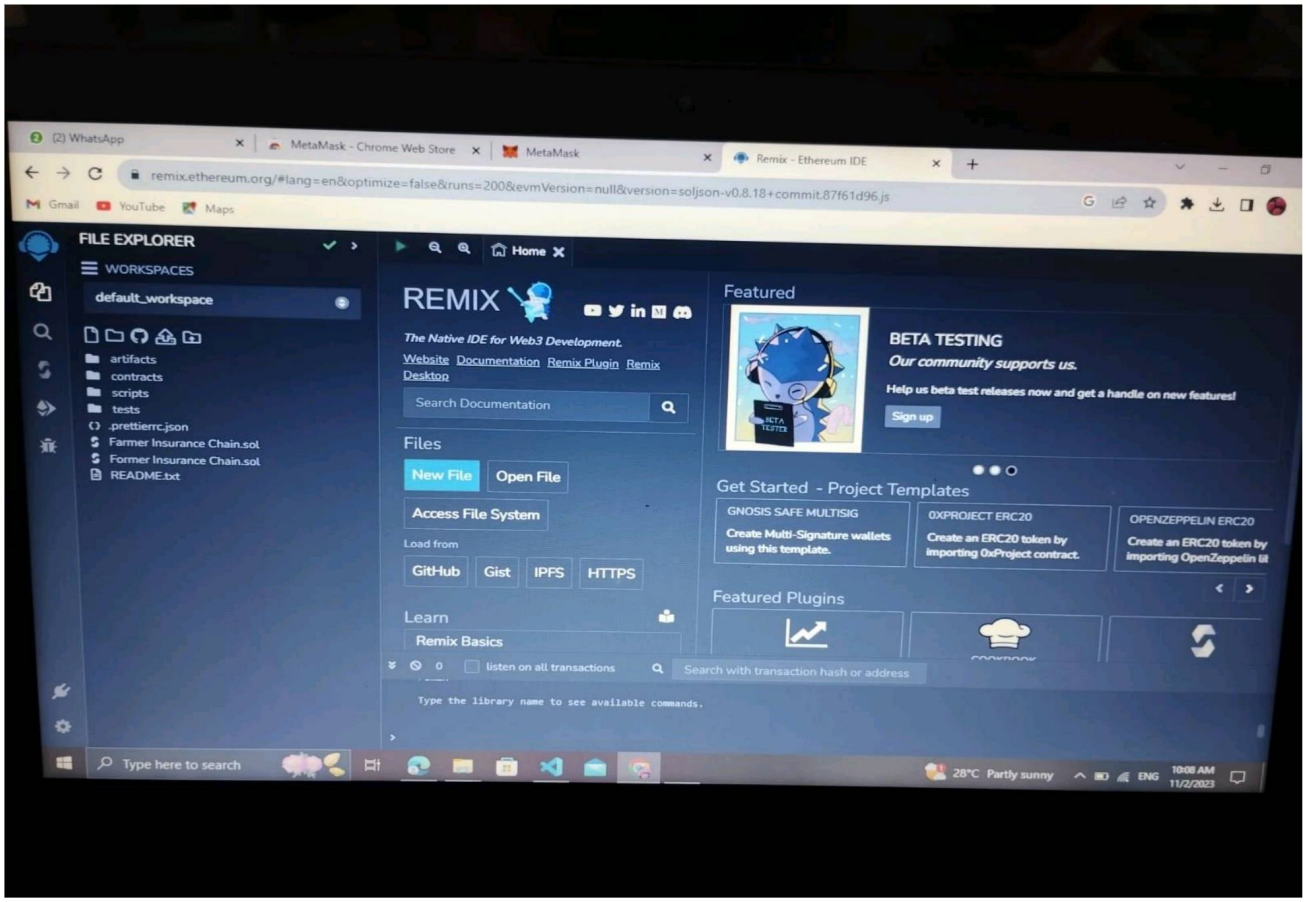
Maintain a release calendar that outlines when each sprint's deliverables or major releases are expected to be available to users or stakeholders. Share this calendar with the team and relevant parties.

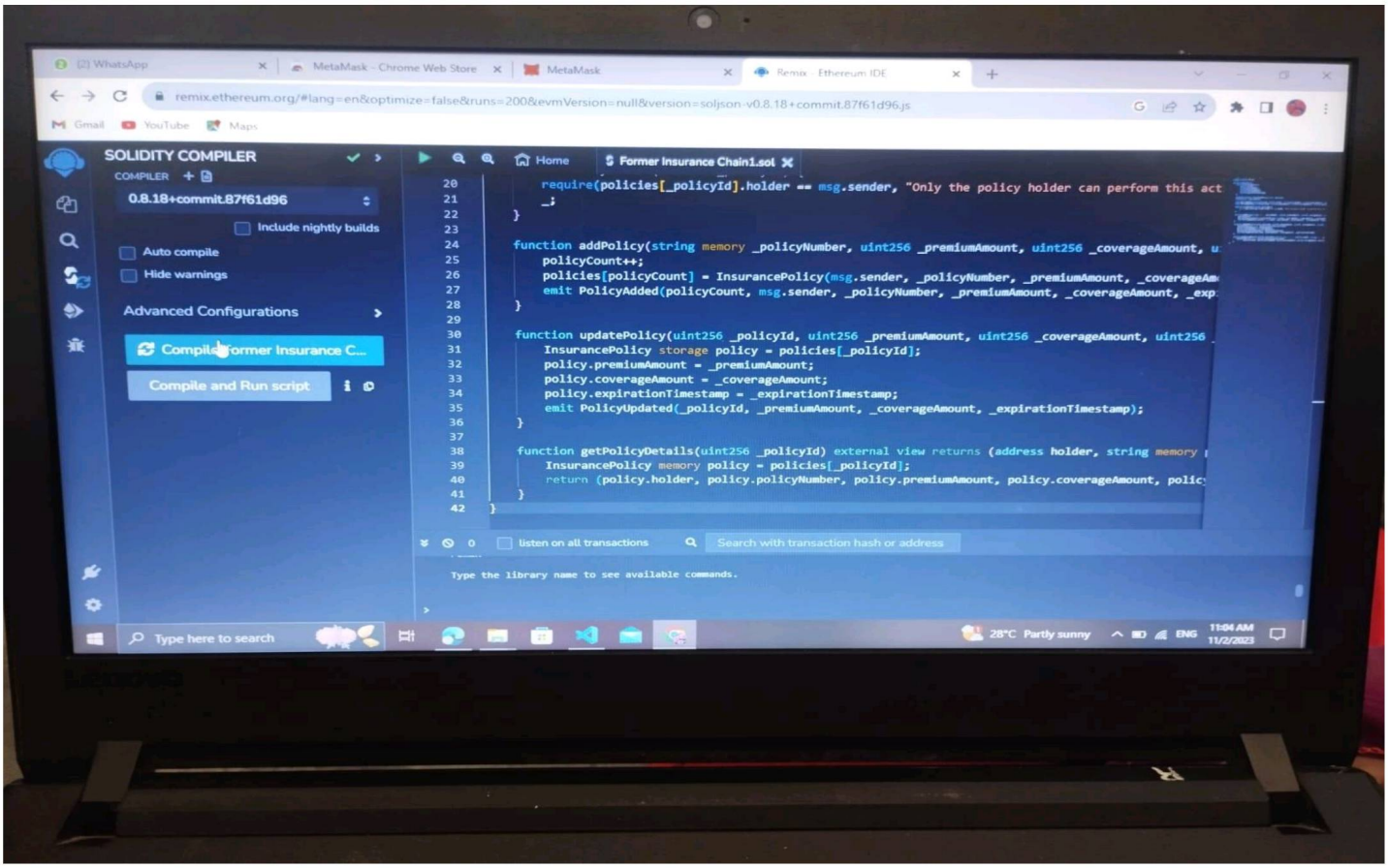
7. CODING & SOLUTIONING (Explain the features added in the projectalong with code)

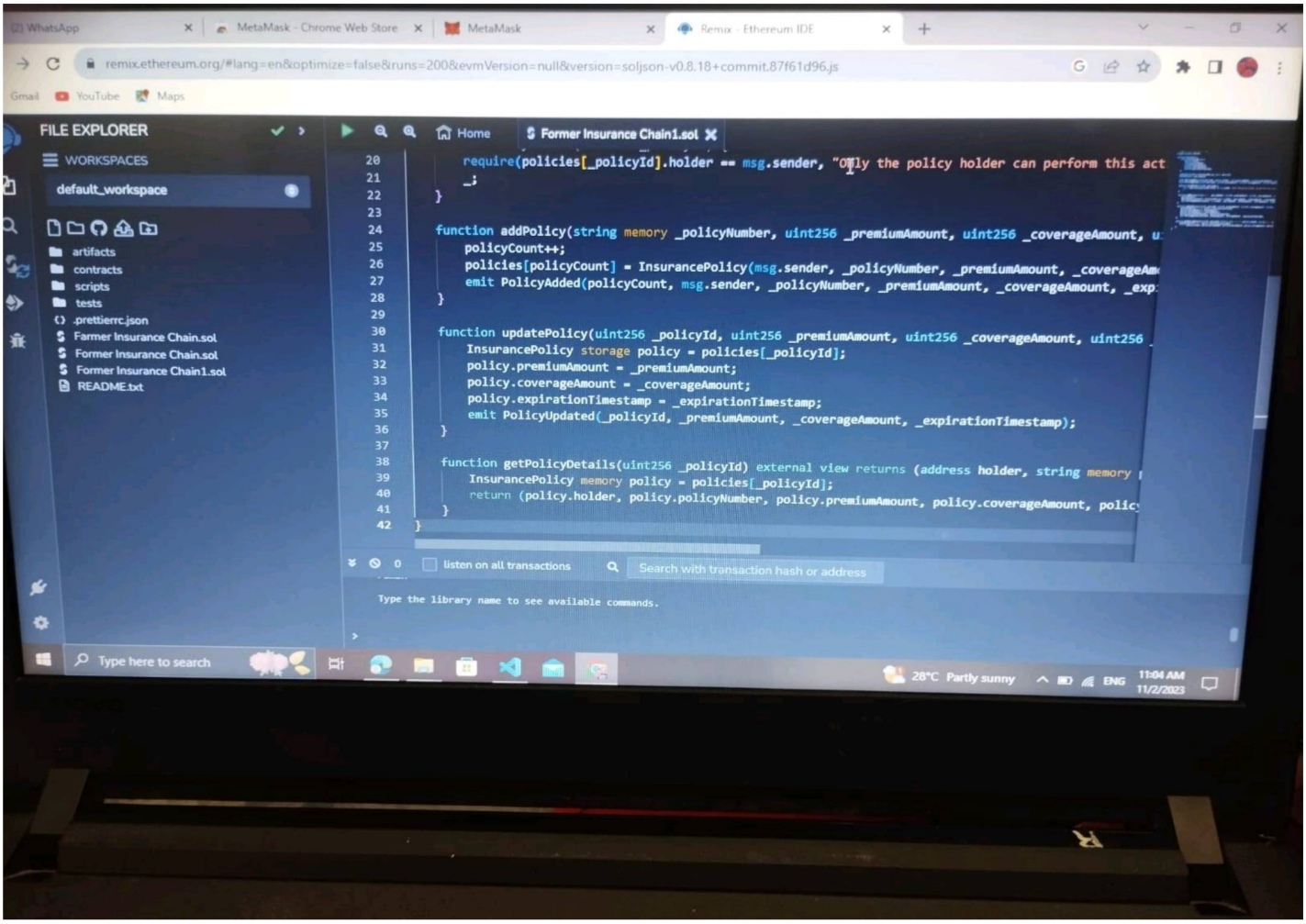
Smartcontrast solidity

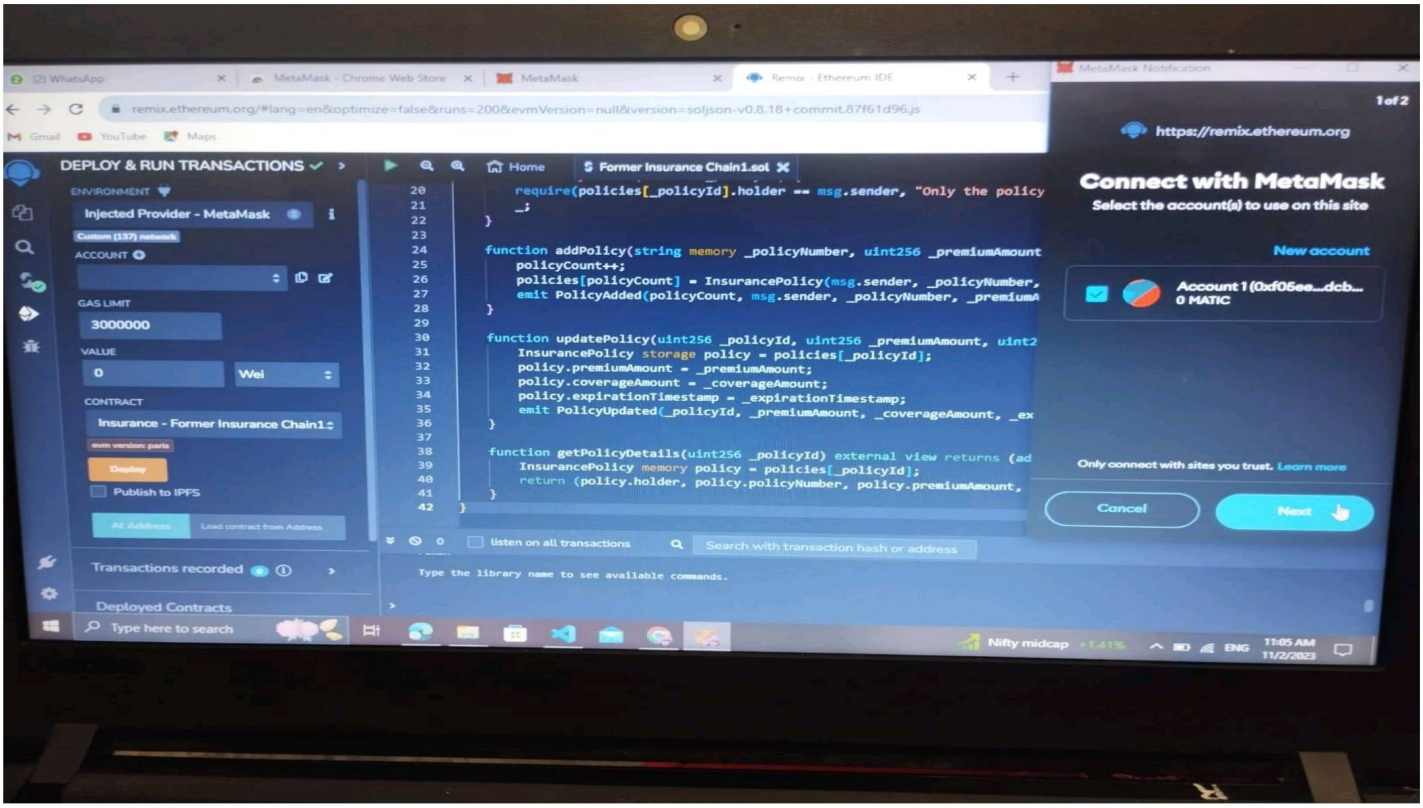
- The "Farmer Insurance" struct represents the properties of a insurance , including title, description, IPFS hash, and the owner's Ethereumaddress.
- The "Insurance" array stores registered Farmer.
- The " register Insurance" function allows a user to register a new by providing the title, description, and the IPFS hash of the Farmer data. It also records the user's Ethereum address as the owner.
- You can emit an event to log the Insurance registration, providing information about the newly registered former insurance.

Smart Contract (Solidity):




SOLIDITY COMPILER
COMPILER
0.8.18+commit.87f61d96
Include nightly builds
Auto compile
Hide warnings
Advanced Configurations
Compile Former Insurance C...
Compile and Run script
Home
Former Insurance Chain1.sol
require(policies[_policyId].holder == msg.sender, "Only the policy holder can perform this act
_;
}
function addPolicy(string memory _policyNumber, uint256 _premiumAmount, uint256 _coverageAmount,
policyCount++;
policies[policyCount] = InsurancePolicy(msg.sender, _policyNumber, _premiumAmount, _coverageAm
emit PolicyAdded(policyCount, msg.sender, _policyNumber, _premiumAmount, _coverageAmount, _exp
}
function updatePolicy(uint256 _policyId, uint256 _premiumAmount, uint256 _coverageAmount, uint256
InsurancePolicy storage policy = policies[_policyId];
policy.premiumAmount = _premiumAmount;
policy.coverageAmount = _coverageAmount;
policy.expirationTimestamp = _expirationTimestamp;
emit PolicyUpdated(_policyId, _premiumAmount, _coverageAmount, _expirationTimestamp);
}
function getPolicyDetails(uint256 _policyId) external view returns (address holder, string memory
InsurancePolicy memory policy = policies[_policyId];
return (policy.holder, policy.policyNumber, policy.premiumAmount, policy.coverageAmount, polic
}
}
listen on all transactions
Search with transaction hash or address
Type the library name to see available commands.
Type here to search
28°C Partly sunny
ENG
11:04 AM
11/2/2023


WhatsApp
MetaMask - Chrome Web Store
MetaMask
Remix - Ethereum IDE
remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.18+commit.87f61d96.js
Gmail
YouTube
Maps
FILE EXPLORER
WORKSPACES
default_workspace
artifacts
contracts
scripts
tests
.prettierrc.json
Farmer Insurance Chain.sol
Former Insurance Chain.sol
Former Insurance Chain1.sol
README.txt
Home
Former Insurance Chain1.sol
require(policies[_policyId].holder == msg.sender, "Only the policy holder can perform this act
_;
}
function addPolicy(string memory _policyNumber, uint256 _premiumAmount, uint256 _coverageAmount, u
policyCount++;
policies[policyCount] = InsurancePolicy(msg.sender, _policyNumber, _premiumAmount, _coverageAm
emit PolicyAdded(policyCount, msg.sender, _policyNumber, _premiumAmount, _coverageAmount, _exp
}
function updatePolicy(uint256 _policyId, uint256 _premiumAmount, uint256 _coverageAmount, uint256
InsurancePolicy storage policy = policies[_policyId];
policy.premiumAmount = _premiumAmount;
policy.coverageAmount = _coverageAmount;
policy.expirationTimestamp = _expirationTimestamp;
emit PolicyUpdated(_policyId, _premiumAmount, _coverageAmount, _expirationTimestamp);
}
function getPolicyDetails(uint256 _policyId) external view returns (address holder, string memory
InsurancePolicy memory policy = policies[_policyId];
return (policy.holder, policy.policyNumber, policy.premiumAmount, policy.coverageAmount, policy
}
}
listen on all transactions
Search with transaction hash or address
Type the library name to see available commands.
Type here to search
28°C Partly sunny
ENG
11:04 AM
11/2/2023


DEPLOY & RUN TRANSACTIONS
ENVIRONMENT
Injected Provider - MetaMask
ACCOUNT
GAS LIMIT
3000000
VALUE
0
Wei
CONTRACT
Insurance - Former Insurance Chain1
Deploy
Publish to IPFS
At Address
Load contract from Address
Transactions recorded
Deployed Contracts
Home
Former Insurance Chain1.sol
require(policies[_policyId].holder == msg.sender, "Only the policy
_;
}
function addPolicy(string memory _policyNumber, uint256 _premiumAmount
policyCount++;
policies[policyCount] = InsurancePolicy(msg.sender, _policyNumber,
emit PolicyAdded(policyCount, msg.sender, _policyNumber, _premiumA
}
function updatePolicy(uint256 _policyId, uint256 _premiumAmount, uint2
InsurancePolicy storage policy = policies[_policyId];
policy.premiumAmount = _premiumAmount;
policy.coverageAmount = _coverageAmount;
policy.expirationTimestamp = _expirationTimestamp;
emit PolicyUpdated(_policyId, _premiumAmount, _coverageAmount, _ex
}
function getPolicyDetails(uint256 _policyId) external view returns (ad
InsurancePolicy memory policy = policies[_policyId];
return (policy.holder, policy.policyNumber, policy.premiumAmount,
}
}
listen on all transactions
Search with transaction hash or address
Type the library name to see available commands.
MetaMask Notification
1 of 2
https://remix.ethereum.org
Connect with MetaMask
Select the account(s) to use on this site
New account
Account 1 (0xf06ee...dcb...
0 MATIC
Only connect with sites you trust. Learn more
Cancel
Next
Type here to search
Nifty midcap +1.41%
ENG
11:05 AM
11/2/2023

!

Aadditional detial

This off-chain storage can include a traditional relational database for userprofiles and audit trail data or decentralized storage systems like IPFS for storing farmer metadata and potentially even the drug files themselves.It's important to note that the Ethereum blockchain is primarily used for storingcritical asset ownership and transaction data, ensuring immutability a transparency. Off-chain storage is often used to handle less critical data and to optimize data access and retrieval times.The database schema can vary significantly based on the specific requirementsof the farmer insurance, and you may need to expand or modify the schema to suityour application's needs. The goal is to balance the benefits of blockchain's immutability withefficient data management and retrieval for users.

8. PERFORMANCE TESTING

8.1 Performace Metrics

Asset Upload and Retrieval Speed:

Metric: Average time taken to upload and retrieve farmer insurance. Importance: Measures the speed of crop yield ensuring quickaccess to insurance.

Blockchain Transaction Throughput:

Metric: Transactions per second (TPS) on the Ethereum blockchain. Importance: Indicates how well the system handles blockchain transactions, which is crucial for scalability.

Smart Contract Execution Time:

Metric: Average time taken for smart contract execution. Importance: Evaluates the efficiency of the blockchain-based logic governing drug ownership and access.

Drug Metadata Search Time:

Metric: Time it takes to search for drug based on metadata. Importance: Measures the responsiveness of the system's search functionality.

User Authorization Latency:

Metric: Time it takes to validate and authorize user access to farmer insurance. Importance: Ensures that authorized users can access insurance promptly whilemaintaining security.

Storage Space Usage:

Metric: Amount of block chain storage used by drug and associated data. Importance: Evaluates the cost and efficiency of storage on the block chain.

insurance Accessibility Uptime:

Metric: Percentage of time accessible farmer land.
Importance: Measures the system's reliability and availability for users.

Security Audit Findings:

Metric: Number and severity of security vulnerabilities discovered during audits.
Importance: Identifies potential risks and the need for security improvements.

User Feedback and Satisfaction:

Metric: User surveys or feedback on system usability and performance.
Importance: Provides insights into user satisfaction and areas for improvement.

Ethereum Network Gas Costs:

Metric: Total gas costs incurred for transactions and contract interactions.
Importance: Measures the cost-efficiency of system operations.

Scalability Metrics:

Metric: System's ability to handle an increasing number of users and farmers

Importance: Assesses how well the system can scale with growing demands.

Audit Trail Accuracy:

Metric: Accuracy and completeness of the audit trail for insurance
Importance: Ensures a reliable record of farmer details history for compliance andaccountability.

Data Backup and Recovery Time:

Metric: Time taken for data backup and recovery operations.
Importance: Evaluates the system's readiness for data recovery in case of failures.

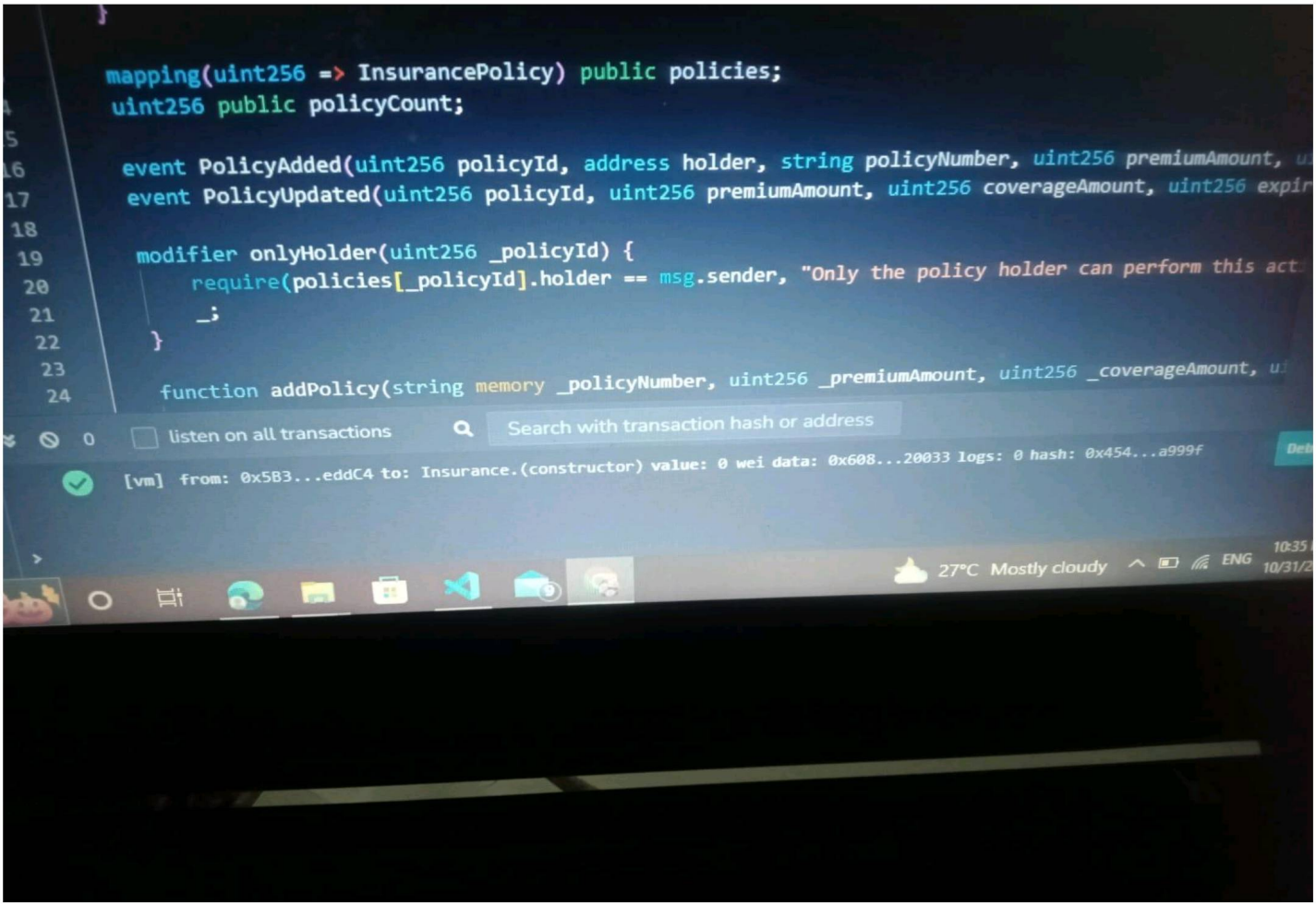
9. RESULTS

Output Screenshots

The " former insurance" feature allows users to officially record and store information about a tracking on the Ethereum blockchain. This process establishes proof of ownership and a public record of the drug ‘s existence. It ismarked as "public" and can be viewed or

accessed by anyone on the Ethereum blockchain.

Get farmer insurance using blockchain is showing a status of the insurance.



10. CONCLUSION

First, further research is required on the transacting parties' motivation to provide genuine and precise information to the blockchain ledger. This might be especially important in the case of smallholder farming. The information generated in the farming process is scattered and owned by individual farmers. Blockchain technologies' benefits for farmers might be dependent on the size of the farm. On the one hand, smaller farms could easily participate in a blockchain-based insurance market. On the other hand, collecting and integrating on-farm data might be more convenient for larger farms. Thus, future research should try to anticipate which farms could benefit an which could lose from the introduction of blockchain-based solutions.Second, obtaining the data uploaded to a blockchain can be very costly, which will be a barrier to the adoption of blockchain technology in the sector. The setup of distributed ledger itself may be relatively cheap, whereas collecting data required for making the ledger useful, e.g., DNA of livestock animals could be expensive. Sampling can reduce the cost, but it requires that the population of products for data collection is large. This means the average cost of data collection is lower for larger farms than smaller ones, which raises the concern of increasing the income discrepancy

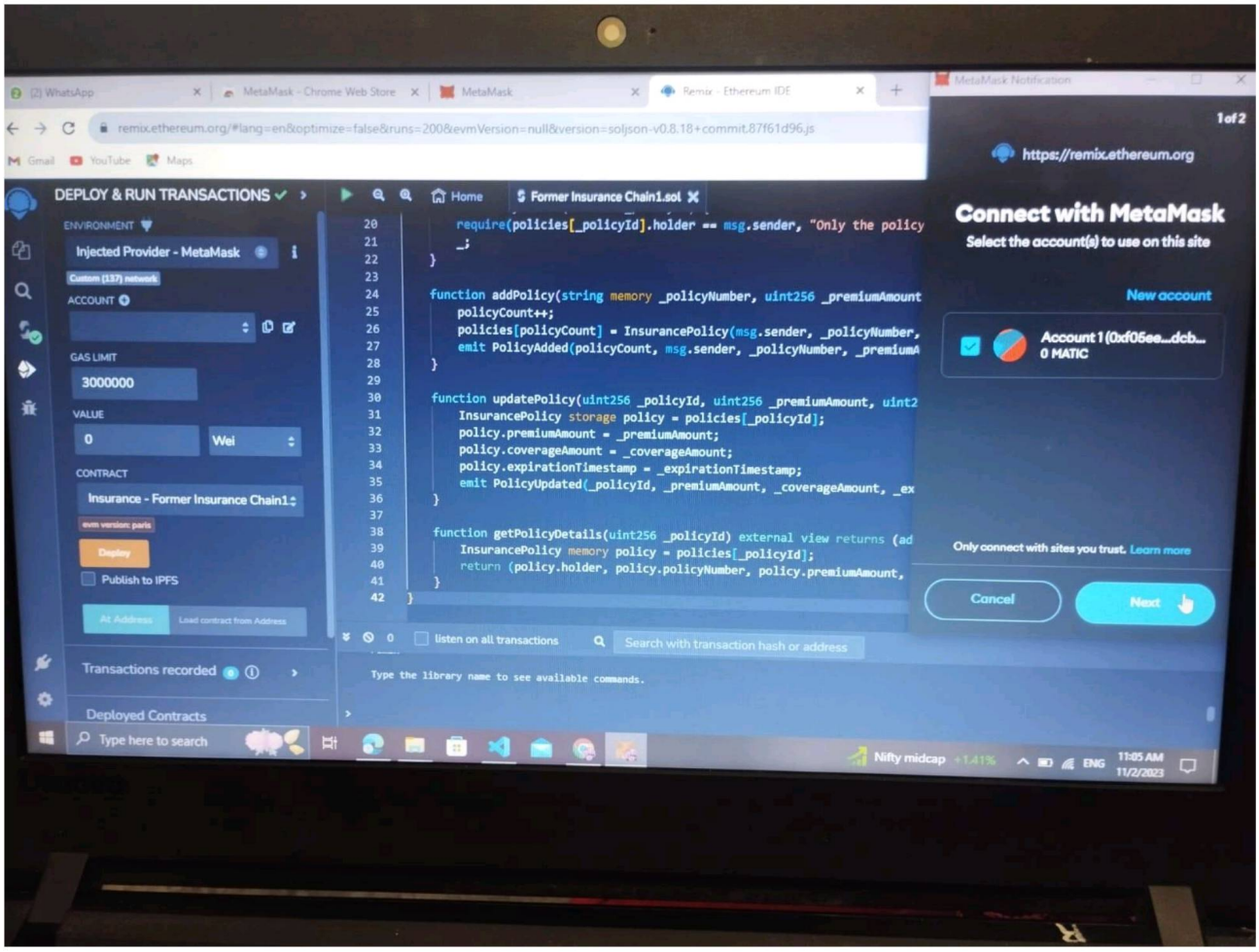
11. FUTURE SCOPE

The use of blockchain technology in the agricultural field enhances the transparency, efficiency, and sustainability of the product. Moreover, it has the potential to alter how we produce, process, and consume agricultural commodities.There are several blockchain-based applications that can benefit buyers, sellers, purchasers, and consumers. Just a few of these applications include supply chain management, smart contracts, agricultural management, carbon credits, and traceability.As more and more stakeholders in the agriculture industry realize the potential of blockchain agriculture technology, we can expect to see a wave of innovation that transforms the industry for the better. The initial value of Blockchain in insurance will result from initiatives which are not broad based across value chain and enterprises. It could find best use cases to build proprietary systems and reduce the costs, while at the same time increase efficiencies of the large financial institutions.Instead of deriving value from a new exchange, eliminating significant information discontinuities between multiple parties in existing business processes will be beneficial. The central goal will be to reduce or eliminate gaps in service and inefficiencies that have traditionally been marked as ’inherent costs’ of insurance transactions - extended wait times, high settlement costs, and extensive negotiations on facts. Figure 4 details near-term applications of Blockchain technologies In the coming future, the growth of blockchain technology has the ability to drastically change the agriculture sector., which are either under way or can be expected soon.

13.APPENDIX

Source Code

Solidity coding:



Java script :

```
  [
{
      "inputs": [
            {
                  "internalType": "string",
                  "name": "_policyNumber",
                  "type": "string"
            },
            {
                  "internalType": "uint256",
                  "name": "_premiumAmount",
                  "type": "uint256"
            },
            {
                  "internalType": "uint256",
                  "name": "_coverageAmount",
                  "type": "uint256"
            },
            {
                  "internalType": "uint256",
                  "name": "_expirationTimestamp",
                  "type": "uint256"
            }
      ],
      "name": "addPolicy",
      "outputs": [],
      "stateMutability": "nonpayable",
      "type": "function"
},
{
      "anonymous": false,
      "inputs": [
            {
                  "indexed": false,
                  "internalType": "uint256",
                  "name": "policyId",
                  "type": "uint256"
            },
            {
                  "indexed": false,
                  "internalType": "address",
                  "name": "holder",
                  "type": "address"
            },
            {
                  "indexed": false,
                  "internalType": "string",
                  "name": "policyNumber",
                  "type": "string"
            },
```

```
            {
                  "indexed": false,
                  "internalType": "uint256",
                  "name": "premiumAmount",
                  "type": "uint256"
            },
            {
                  "indexed": false,
                  "internalType": "uint256",
                  "name": "coverageAmount",
                  "type": "uint256"
            },
            {
                  "indexed": false,
                  "internalType": "uint256",
                  "name": "expirationTimestamp",
                  "type": "uint256"
            }
      ],
      "name": "PolicyAdded",
      "type": "event"
},
{
      "anonymous": false,
      "inputs": [
            {
                  "indexed": false,
                  "internalType": "uint256",
                  "name": "policyId",
                  "type": "uint256"
            },
            {
                  "indexed": false,
                  "internalType": "uint256",
                  "name": "premiumAmount",
                  "type": "uint256"
            },
            {
                  "indexed": false,
                  "internalType": "uint256",
                  "name": "coverageAmount",
                  "type": "uint256"
            },
            {
                  "indexed": false,
                  "internalType": "uint256",
                  "name": "expirationTimestamp",
                  "type": "uint256"
            }
      ],
      "name": "PolicyUpdated",
      "type": "event"
```

```
},
{
	"inputs": [
		{
			"internalType": "uint256",
			"name": "_policyId",
			"type": "uint256"
		},
		{
			"internalType": "uint256",
			"name": "_premiumAmount",
			"type": "uint256"
		},
		{
			"internalType": "uint256",
			"name": "_coverageAmount",
			"type": "uint256"
		},
		{
			"internalType": "uint256",
			"name": "_expirationTimestamp",
			"type": "uint256"
		}
	],
	"name": "updatePolicy",
	"outputs": [],
	"stateMutability": "nonpayable",
	"type": "function"
},
{
	"inputs": [
		{
			"internalType": "uint256",
			"name": "_policyId",
			"type": "uint256"
		}
	],
	"name": "getPolicyDetails",
	"outputs": [
		{
			"internalType": "address",
			"name": "holder",
			"type": "address"
		},
		{
			"internalType": "string",
			"name": "policyNumber",
			"type": "string"
		},
		{
			"internalType": "uint256",
			"name": "premiumAmount",
```

```
                "type": "uint256"
            },
            {
                "internalType": "uint256",
                "name": "coverageAmount",
                "type": "uint256"
            },
            {
                "internalType": "uint256",
                "name": "expirationTimestamp",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "name": "policies",
        "outputs": [
            {
                "internalType": "address",
                "name": "holder",
                "type": "address"
            },
            {
                "internalType": "string",
                "name": "policyNumber",
                "type": "string"
            },
            {
                "internalType": "uint256",
                "name": "premiumAmount",
                "type": "uint256"
            },
            {
                "internalType": "uint256",
                "name": "coverageAmount",
                "type": "uint256"
            },
            {
                "internalType": "uint256",
                "name": "expirationTimestamp",
                "type": "uint256"
            }
        ],
```

```
            "stateMutability": "view",
            "type": "function"
      },
      {
            "inputs": [],
            "name": "policyCount",
            "outputs": [
                  {
                        "internalType": "uint256",
                        "name": "",
                        "type": "uint256"
                  }
            ],
            "stateMutability": "view",
            "type": "function"
      }
]
```

HTML coding:

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

contract Insurance {
    struct InsurancePolicy {
        address holder;
        string policyNumber;
        uint256 premiumAmount;
        uint256 coverageAmount;
        uint256 expirationTimestamp;
    }

    mapping(uint256 => InsurancePolicy) public policies;
    uint256 public policyCount;

    event PolicyAdded(uint256 policyId, address holder, string policyNumber,
uint256 premiumAmount, uint256 coverageAmount, uint256 expirationTimestamp);
    event PolicyUpdated(uint256 policyId, uint256 premiumAmount, uint256
coverageAmount, uint256 expirationTimestamp);

    modifier onlyHolder(uint256 _policyId) {
        require(policies[_policyId].holder == msg.sender, "Only the policy holder
can perform this action");
        _;
    }

    function addPolicy(string memory _policyNumber, uint256 _premiumAmount,
uint256 _coverageAmount, uint256 _expirationTimestamp) external {
        policyCount++;
        policies[policyCount] = InsurancePolicy(msg.sender, _policyNumber,
_premiumAmount, _coverageAmount, _expirationTimestamp);
```

```
        emit PolicyAdded(policyCount, msg.sender, _policyNumber, _premiumAmount,
_coverageAmount, _expirationTimestamp);
    }

    function updatePolicy(uint256 _policyId, uint256 _premiumAmount, uint256
_coverageAmount, uint256 _expirationTimestamp) external onlyHolder(_policyId) {
        InsurancePolicy storage policy = policies[_policyId];
        policy.premiumAmount = _premiumAmount;
        policy.coverageAmount = _coverageAmount;
        policy.expirationTimestamp = _expirationTimestamp;
        emit PolicyUpdated(_policyId, _premiumAmount, _coverageAmount,
_expirationTimestamp);
    }

    function getPolicyDetails(uint256 _policyId) external view returns (address
holder, string memory policyNumber, uint256 premiumAmount, uint256
coverageAmount, uint256 expirationTimestamp) {
        InsurancePolicy memory policy = policies[_policyId];
        return (policy.holder, policy.policyNumber, policy.premiumAmount,
policy.coverageAmount, policy.expirationTimestamp);
    }
}
```

GitHub :

https://github.com/Subashini.P/Naan-mudhalvan-

Project Video Demo Link

https://github.com/subashini/Naanmudhalvan/commits?author=Subashini.P