


COURSE NAME : BLOCK CHAIN

GROUP NUMBER : 01

PROJECT TITLE : A BLOCKCHAIN-BASED FARMER INSURANCE CHAIN

YEAR : 4th YEAR

DEPARTMENT : B.E - CSE

SEMESTER : 07

GROUP MEMBERS :

1. ANJALI S J

2. ABI PREETHA R

3. JEBA BLESSA A

4. PAVITHRA R

GUIDED BY : JASMINE REJULA J

SPOC NAME : Mr. M. DIVIN KU

Project Report

ABSTRACT

Blockchain is a method in which a confirmation of a transaction is kept by means of a crypto- currency. The record is maintained transversely, linking several computers in a peer to peer network. Contracts, transactions, and the records of them define the economic system of a country. They set boundaries and provide security to the assets. Considering the features of blockchain such as immutability and maintaining the footage of transaction details, this paper highlights the usage of blockchain technology with farmer's portal that keep the footage of selling and buying information of crops. The proposed solution uses the python as a programming language in integration with the blockchain system that will benefit the farmers or vendors and individuals by preserving the contract of trade. An interface for the farmers is designed using a python programming language in addition with blockchain technology, which is used to store the information related to seller, buyer, selling and buying an item and total value transacted.

INTRODUCTION

1.1. PROJECT OVERVIEW

The Farmer Insurance Chain project leverages are in blockchain technology to revolutionize the insurance industry by enhancing transparency, security, and efficiency. Through a decentralized and immutable ledger, this innovative solution simplifies and accelerates insurance processes, from policy issuance to claims settlement. By enabling smart contracts, it automates the verification and execution of insurance agreements, reducing the risk of fraud and ensuring swift payouts. The blockchain's distributed nature ensures that all stakeholders, including policyholders, insurers, and regulators, have real-time access to a single source of truth, facilitating trust and collaboration. This project not only streamlines operations but also promotes trust and integrity within the insurance ecosystem, ultimately benefiting farmers and insurers alike.. Through smart contracts, the project automates the claims process, reducing administrative overhead and expediting payouts. Additionally, the immutable nature of blockchain records enhances auditability and compliance, ultimately benefiting both insurers and policyholders. By harnessing the power of blockchain, the Farmer Insurance Chain project aims to bring efficiency, trust, and transparency to the insurance ecosystem, fostering a more seamless and equitable experience for all stakeholders involved.

1.2 .PURPOSE

Blockchain technology provides a transparent and immutable ledger that records all insurance transactions. This transparency builds

trust among policyholders, insurers, and regulators by ensuring that all parties can verify the accuracy of records and transactions. The decentralized nature of blockchain makes it more challenging for bad actors to manipulate or commit fraud within the insurance system. By securing policyholder data and claims on a blockchain, the project reduces the risk of fraudulent activities. Smart contracts, which are self-executing agreements on the blockchain, automate many insurance processes, such as claims processing. This reduces administrative costs, minimizes errors, and speeds up claims settlements, leading to an overall more efficient insurance ecosystem.

LITERATURE SURVEY

2.1 EXISTING PROBLEM

1.TITLE: an interface for Indianfarmer

AUTHORS: Ghosh, Soumalya, A. B. Garg, Sayan Sarcar, PSV S. Sridhar, Ojasvi Maleyvar, and Raveesh Kapoor

DESCRIPTION:

Rapid growth in the field of ICT helps in basic aspects of mankind like- agriculture, education, healthcare etc. However, the moderate technical growth of ICT applications is confined to the community of a limited number of people, who live in digital pockets. The illiterate people like –farmer, shopkeeper etc. are unable to take the advantages of the ICT revolution. According to the UNESCO report, population of such people inthe globe is 64% who are unable to use the

technology either language or technical barrier. Moreover the percentage (76%) must be increased in the context of developing countries. The essential agriculture information is very useful to a farmer for taking effective decision

thus we proposed to develop an iconic interface which is integrated with speech based interaction in Indian languages. The proposed interface is critically evaluated with the farmer from different states of India. The evaluation results proved the effectiveness of the proposed interface.

2.TITLE:Android based solut ion for Indian agriculture

AUTHORS: Singhal, Manav, Kshit ij Verma, and Anupam Shukla

DESCRIPTION:

Information and Communication Technology (ICT) in agriculture is an emerging field focusing on the enhancement of agricultural and rural development in India. It involves innovative applications using ICT in the rural domain. The advancement of ICT can be utilized for providing accurate and timely relevant information and services to the farmers, thereby facilitating an environment for remunerative agriculture. This paper describes a mobile based application for farmers which would help them in their farming activities. We propose an android based mobile application - Krishi Ville which would take care of the updates of the different agricultural commodities, weather forecast updates.

2.2 REFERENCES

[1] Lakhani, Karim R., and M. Iansit i. "The t ruth about blockchain." Harvard Business Review 95 (2017): 118-127.

[2] Hileman, Garrick, and Michel Rauchs. "2017 global blockchain benchmarking study." Available at SSRN 3040224 (2017).

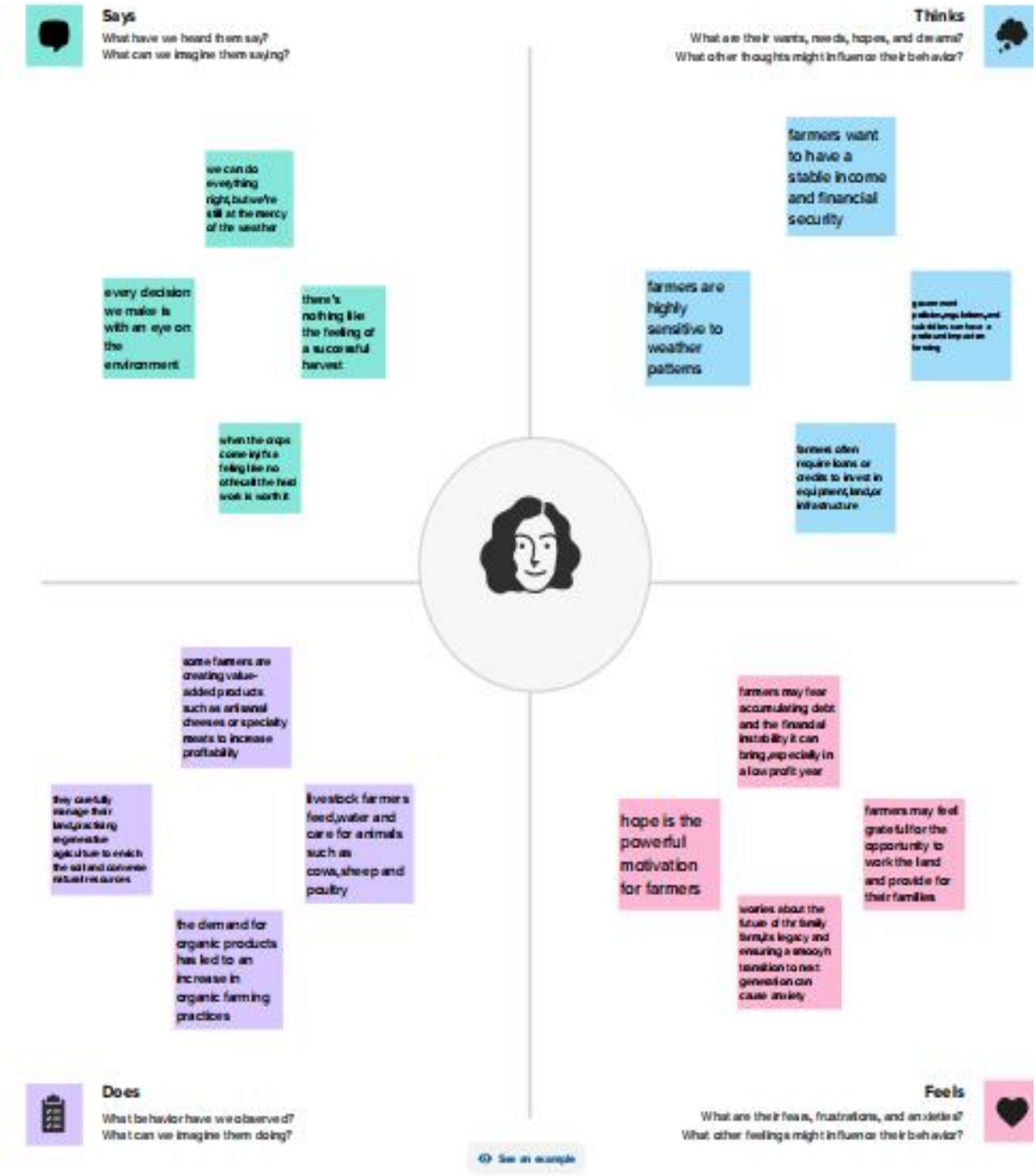
[3] Mohanta, Bhabendu K., Debasish Jena, Soumyashree S. Panda, and Srichandan Sobhanayak. "Blockchain Technology: A Survey on Applicat ions and Security Privacy Challenges." Internet of Things (2019): 100107.

2.3 PROBLEM STATEMENT DEFINITION

"Farmers face numerous risks and uncertainties in their agricultural endeavors, including crop failures, natural disasters, and volatile market conditions. Access to reliable insurance solutions is crucial to safeguard their livelihoods and investments. However, in many regions, there exists a gap in the availability and accessibility of affordable insurance tailored to the specific needs of farmers. This gap is often attributed to various factors, such as inadequate infrastructure, lack of awareness, and the high cost of insurance products. Consequently, the problem at hand is to establish an effective and efficient farmer insurance chain that addresses these issues and ensures that farmers can access and afford the insurance they need to protect their crops, livestock, and financial stability."

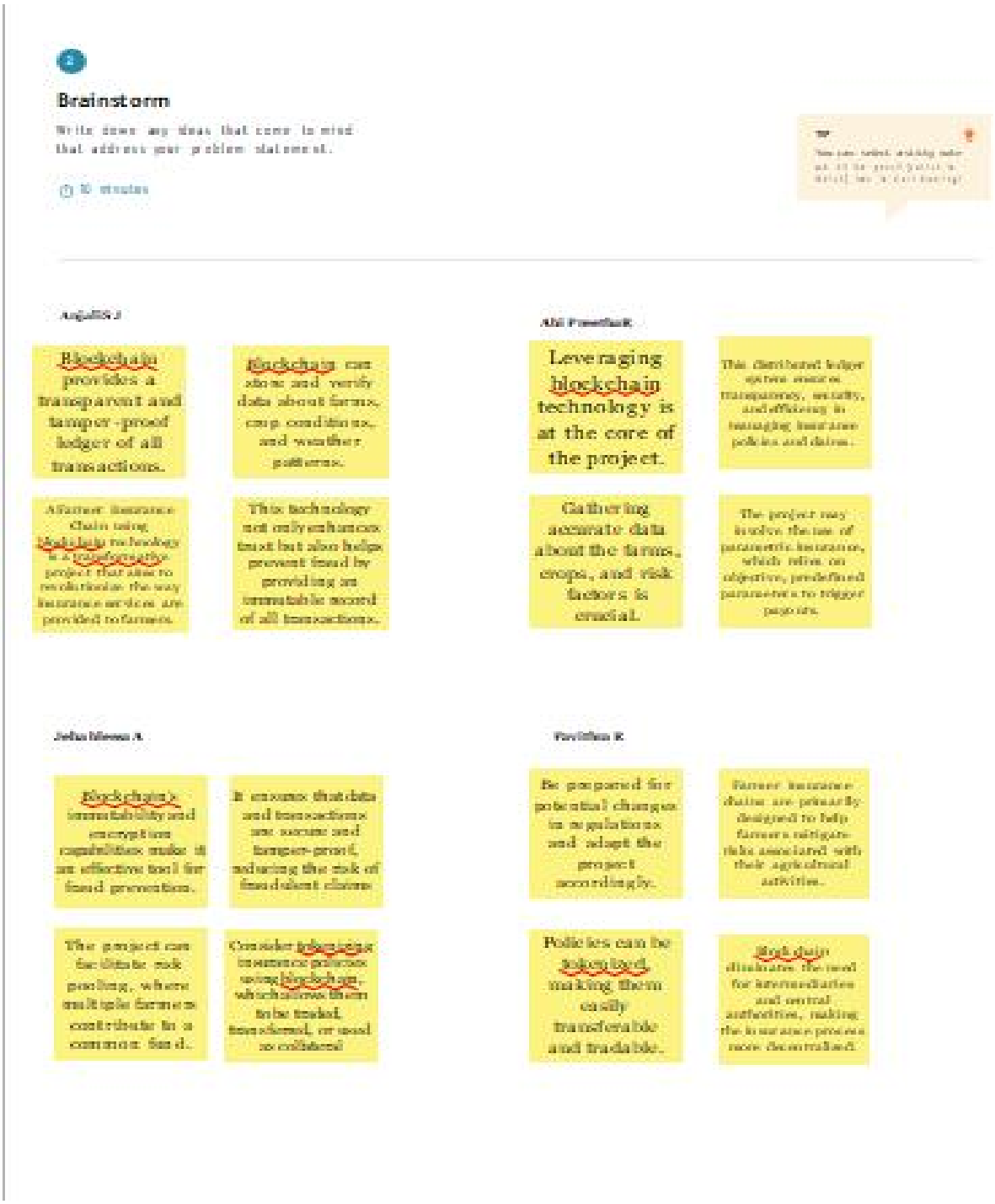
IDEATON & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

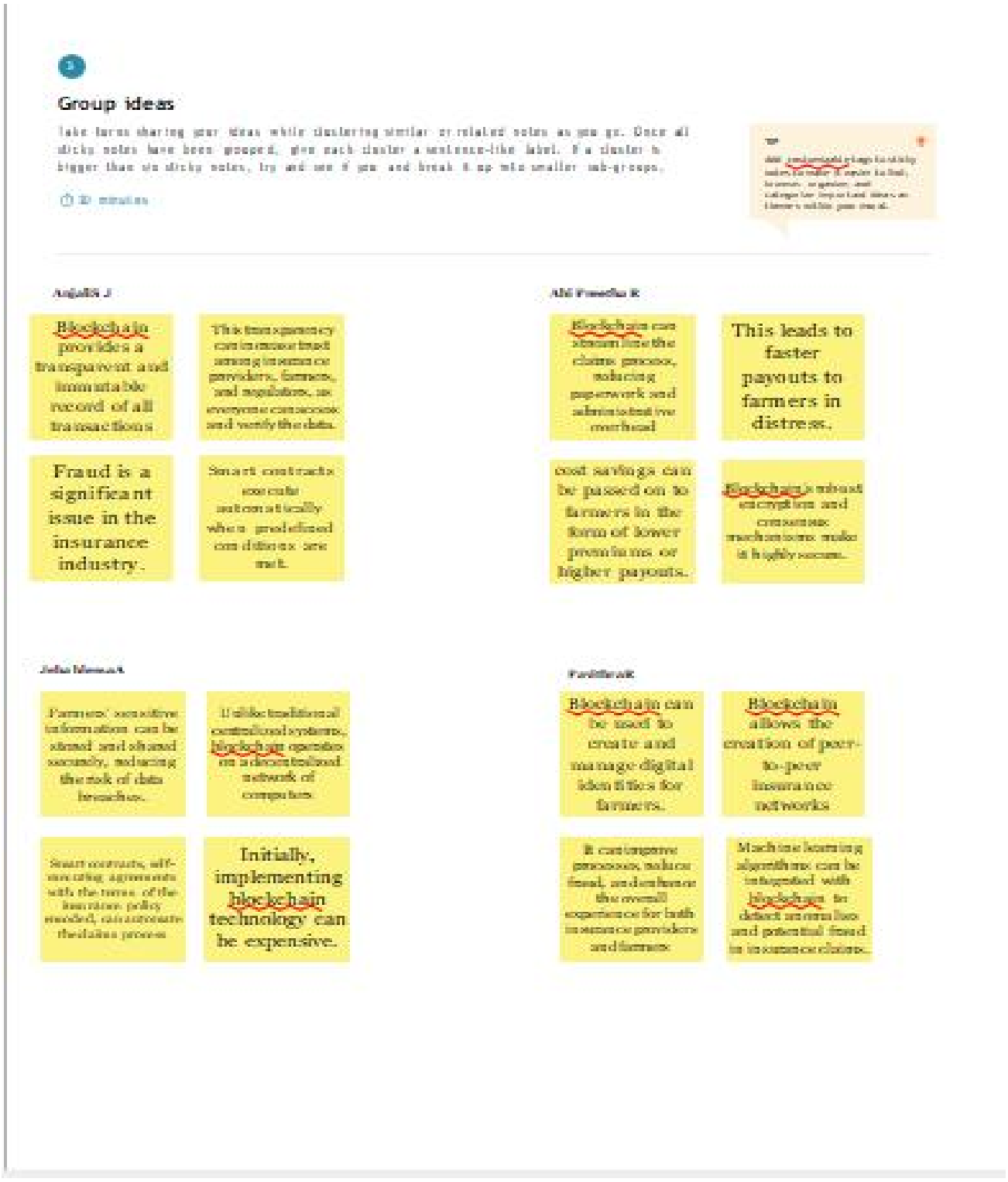


3.2 IDEATON & BRAINSTROMING

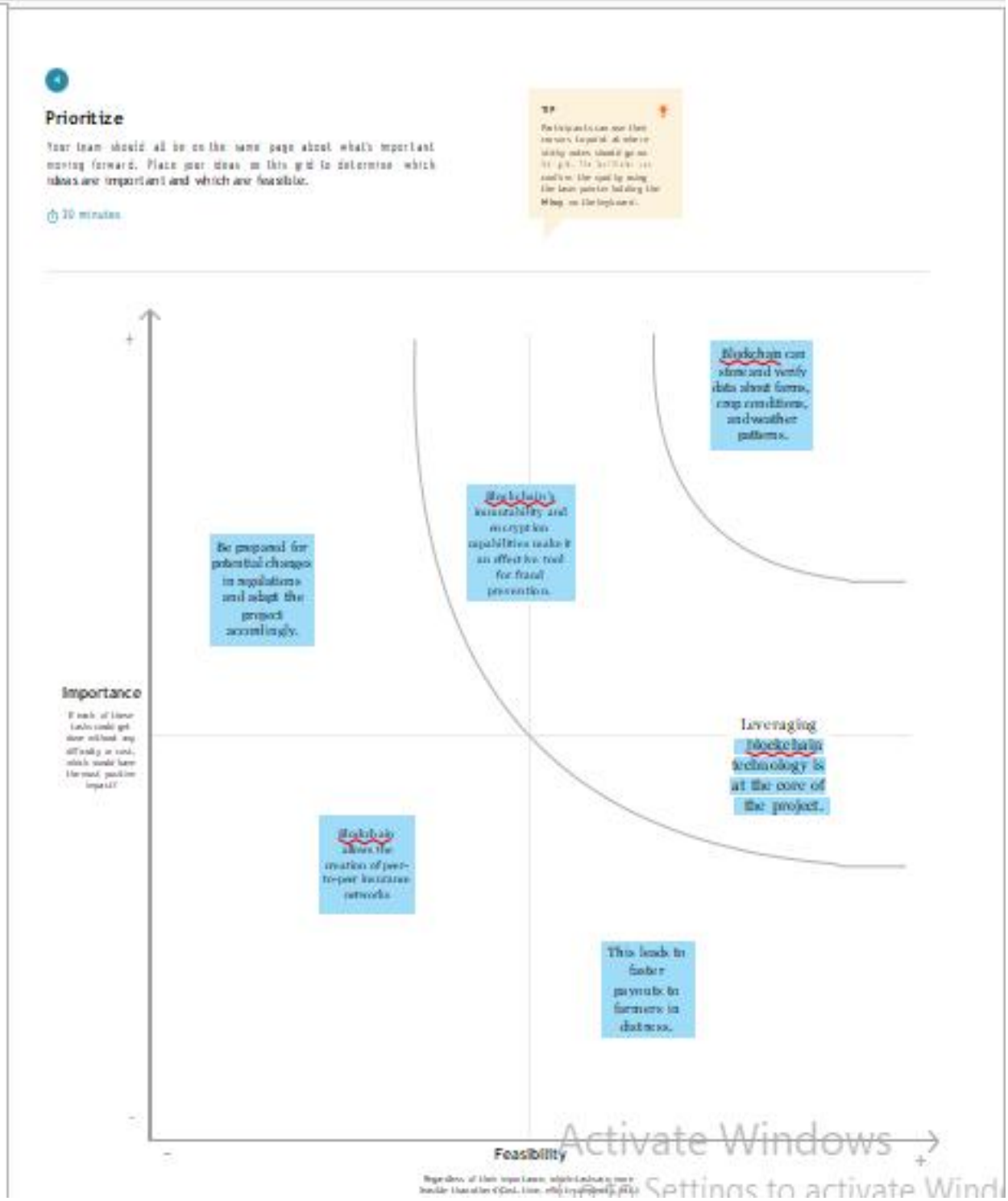
BRAINSTORM



GROUP IDEA


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
Anjali S J
Blockchain provides a transparent and immutable record of all transactions
This transparency can increase trust among insurance providers, farmers, and regulators, as everyone can access and verify the data.
Fraud is a significant issue in the insurance industry.
Smart contracts execute automatically when predefined conditions are met.
Abi Preetha R
Blockchain can streamline the claims process, reducing paperwork and administrative overhead
This leads to faster payouts to farmers in distress.
cost savings can be passed on to farmers in the form of lower premiums or higher payouts.
Blockchain's robust encryption and consensus mechanisms make it highly secure.
Farmers' sensitive information can be stored and shared securely, reducing the risk of data breaches.
Unlike traditional centralized systems, blockchain operates on a decentralized network of computers
Smart contracts, self-executing agreements with the terms of the insurance policy encoded, can automate the claims process
Initially, implementing blockchain technology can be expensive.
Blockchain can be used to create and manage digital identities for farmers.
Blockchain allows the creation of peer-to-peer insurance networks
It can improve processes, reduce fraud, and enhance the overall experience for both insurance providers and farmers
Machine learning algorithms can be integrated with blockchain to detect anomalies and potential fraud in insurance claims.

PRIORITISE


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.
10 minutes
Blockchain can store and verify data about farms, crop conditions, and weather patterns.
Blockchain's immutability and encryption capabilities make it an effective tool for fraud prevention.
Be prepared for potential changes in regulations and adapt the project accordingly.
Importance
Leveraging blockchain technology is at the core of the project.
Blockchain allows the creation of peer-to-peer insurance networks
This leads to faster payouts to farmers in distress.
Feasibility
Activate Windows
Settings to activate Windows

3.3 PROPOSED SOLUTION:

A proposed solution for the farmer insurance chain, leveraging blockchain technology, can address several key challenges in the agricultural insurance sector.

Blockchain's transparent, decentralized, and immutable ledger system offers an innovative approach to enhance the efficiency and trustworthiness of farmer insurance processes.

Firstly, blockchain can facilitate the creation of smart contracts, which automate insurance policies, claims, and payouts. This eliminates the need for intermediaries, reducing administrative costs and the potential for fraud.

Secondly, blockchain can enhance data accuracy and security. Farmers' information, such as land records, weather data, and crop yield information, can be securely stored on the blockchain. This trustable data can help insurers assess risks more accurately and reduce disputes during claims processing.

Furthermore, the blockchain's transparency can increase trust among all stakeholders. Farmers can verify the terms of their insurance policies, and insurers can ensure that the data provided by farmers is accurate. This trust can encourage more farmers to participate in insurance programs.

Lastly, the blockchain can enable the creation of parametric insurance products that automatically trigger payouts when pre-defined conditions, like adverse weather events, are met. This offers timely compensation to farmers without the need for lengthy claims processing.

In conclusion, the implementation of blockchain technology in the farmer insurance chain has the potential to

revolutionize the sector by streamlining processes, reducing costs, enhancing data accuracy, and increasing trust. This, in turn, can help ensure that more farmers have access to affordable and effective insurance coverage, ultimately safeguarding their livelihoods and investments.

3.4 PROBLEM SOLUTION FIT

The problem solution fit for a farmer insurance chain is a critical consideration when implementing innovations in this sector. In this context, the alignment of the proposed solutions with the specific challenges faced by farmers and the insurance industry is essential.

The challenges facing farmers, such as crop failures, natural disasters, and limited access to insurance, necessitate a solution that is both accessible and tailored to their needs. It is crucial that the solution addresses these challenges effectively and efficiently.

A well-fitted solution would involve developing insurance products and services that are affordable, easy to understand, and accessible to a broad range of farmers, including smallholders and those in remote areas. Additionally, the solution should take into account the variable nature of agricultural risks and provide flexibility to adapt coverage based on regional and seasonal variations.

The solution should also consider the technological constraints and limitations often found in rural agricultural areas, ensuring that farmers can easily access and utilize the insurance products. It should provide user-friendly interfaces and support to accommodate different levels of digital literacy among farmers.

Furthermore, the solution should establish a robust claims and payouts system, reducing the time and effort required for farmers to receive compensation. Timely and fair payouts are essential to

maintaining trust in the insurance chain.

To ensure a strong problem solution fit, the proposed system should continuously evolve based on feedback and real-world data, adapting to changing agricultural conditions and market dynamics. It should also foster partnerships with local agricultural organizations and government agencies to maximize its impact and reach.

In conclusion, a well-fitted solution for the farmer insurance chain should directly address the unique challenges faced by farmers and the insurance sector, providing accessible, affordable, and efficient coverage that safeguards farmers' livelihoods and investments while ensuring a sustainable and resilient agricultural sector.

REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

The functional requirements for a Farmer Insurance Chain describe the essential features and capabilities of the system to ensure it effectively serves the needs of farmers, insurance providers, and other stakeholders in the agriculture and insurance sectors. These requirements can be grouped into various categories:

1.User Registration and Authentication:

The system should provide a user-friendly registration process for farmers, insurance companies, and administrators. Users should be able to create accounts and log in securely to access the platform.

2. Policy Management:

Farmers should be able to view available insurance policies, purchase policies, and manage their policy information. The system should support different types of crop insurance and coverage options.

3. Premium Calculation:

The platform should calculate insurance premiums based on factors such as the type of crop, location, and historical weather data. Farmers should be able to receive quotes for different policies.

4. Claim Submission:

Farmers should be able to submit insurance claims online, providing relevant documentation and evidence in case of crop loss or damage due to covered events like adverse weather conditions, pests, or diseases.

5. Claim Processing:

Insurance providers and administrators should have the tools to review and process claims efficiently. The system should automate calculations for claim payouts based on policy terms and conditions.

6. Payment Integration:

The system should support payment gateways for farmers to pay premiums and receive claim payouts. It should also ensure secure and transparent transactions using blockchain technology.

7. Crop Monitoring:

The platform may integrate with IoT devices, weather APIs, and remote sensing technology to monitor crop conditions and validate claims.

8. Notifications:

The system should send notifications and updates to farmers regarding policy renewals, weather alerts, and claim status. It should also notify administrators and insurance companies about new claims.

9. Blockchain Integration:

Utilize blockchain technology to create transparent and immutable records of policy agreements, premium payments, and claim transactions. Smart contracts can automate policy terms and claim settlements.

10. Data Security and Privacy:

Ensure robust data security and privacy measures to protect users' personal and financial information. Comply with data protection regulations.

These functional requirements provide a comprehensive framework for a Farmer Insurance Chain, aiming to streamline insurance processes for farmers while improving transparency and trust within the agricultural insurance ecosystem

It is important to collaborate closely with stakeholders and domain experts to fine-tune these requirements and build a robust and user-friendly system that addresses the specific needs of the target market.

4.2 NON-FUNCTIONAL REQUIREMENT

In addition to the functional requirements, non-functional requirements are crucial for ensuring the reliability, security, performance, and usability of the Farmer Insurance Chain. These non-functional requirements describe the qualities and characteristics that the system should possess:

1. Security:

The system must implement robust security measures to protect sensitive user data, financial transactions, and smart contract functionality. It should follow best practices for secure coding, encryption, and access control.

2. Scalability:

The platform should be designed to handle increasing user loads and a growing database of policies and claims. Scalability is vital to

ensure that the system remains responsive as the user base expands.

3. Performance:

The system must provide fast response times for user interactions, including policy purchases, claim submissions, and premium payments. It should handle high concurrent user requests efficiently.

4. Reliability:

Farmers rely on the system for crucial financial protection. The system should be highly reliable, minimizing downtime, and ensuring data integrity and availability.

5. Availability:

The platform should aim for high availability, ensuring that users can access their accounts and services 24/7. Implement redundancy and failover mechanisms to minimize service disruptions.

6. Data Integrity:

Ensure that data remains accurate and consistent. Implement data validation checks to prevent errors and maintain the integrity of insurance policy and claim data

These non-functional requirements are essential for creating a robust and reliable Farmer Insurance Chain that can effectively serve the needs of both farmers and insurance providers while maintaining data security and privacy. Adhering to these requirements will contribute to a

successful and trustworthy platform.

PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS:

The DFD for the Farmer Insurance Chain illustrates the flow of information within the system. It typically consists of processes, data stores, data flows, and external entities. In this context, you have external entities like "Farmers," "Insurance Providers," and "Administrators," as well as processes like "Policy Management," "Claim Processing," and "User Registration."

External Entities:

- Farmers: Users who purchase insurance policies and submit claims.

- Insurance Providers: Organizations offering insurance policies and processing claims.

- Administrators: System administrators responsible for

managing the platform.

Processes:

- User Registration: This process involves farmers and administrators registering and verifying their accounts.

- Policy Management: This process allows farmers to browse, purchase, and manage insurance policies.

- Claim Processing:

Insurance providers and administrators evaluate and process insurance claims.

- Premium Calculation: Calculate premium amounts based on various factors.

- Payment Integration: Handle premium payments and claim disbursements.

- Reporting and Analytics: Generate reports for insurance providers and administrators.

Data Stores:

- User Database: Stores user information, including account details.

- Policy Database: Contains information about insurance policies.

- Claim Database: Stores information related to submitted claims.

- Transaction Records: Records all financial transactions, including premium payments and claim disbursements.

Data Flows:

- User Registration Data: Flow from external entities to the User Database.

- Policy Information: Flow between Policy Management and Policy Database.

- Claim Data: Flow from Farmers to the Claim Database for submission.

- Financial Transactions: Flow between the Payment Integration process and Transaction Records.

- Premium Calculation Data: Flow between Policy Management and Premium Calculation.

USER STORIES:

User stories offer narrative descriptions of how users interact with the system. They help define specific functionality from the user's perspective. Here are some user stories for the Farmer Insurance Chain:

1. User Registration:

- As a farmer, I want to create an account on the Farmer Insurance Chain platform so that I can access and purchase insurance policies.

- As an administrator, I want to verify user accounts to maintain the integrity of the platform.

2. Policy Management:

- As a farmer, I want to view a list of available insurance policies and their coverage options to make an informed decision.

- As a farmer, I want to purchase a crop insurance policy by selecting the type of crop and coverage.

- As a farmer, I want to receive a premium quote based on my policy selections.

- As a farmer, I want to renew or cancel an existing policy.

3. Claim Processing:

- As a farmer, I want to submit a claim online with all required documentation in case my crops are damaged by covered events.

- As an insurance provider, I want to review submitted claims, validate the data, and calculate the claim payout.

- As an administrator, I want to oversee and approve claim processing to ensure fairness and accuracy.

4. Premium Calculation:

- As a farmer, I want to know how the premium amount for my selected policy is calculated based on factors like crop type, location, and historical data.

5. Payment Integration:

- As a farmer, I want to pay my insurance premium securely through the platform using various payment methods.

- As a farmer, I want to receive claim payouts promptly and securely in case of covered events.

6. Reporting and Analytics:

- As an insurance provider, I want to generate reports on policy performance, claim settlements, and overall system health for decision-making and risk assessment.

These user stories provide a high-level view of how different types of users interact with the Farmer Insurance Chain system and the functionality they require. They serve as a basis for detailed feature development and user experience design.

5.2 SOLUTION ARCHITECTURE:

Creating a solution architecture diagram for the Farmer Insurance Chain involves representing the key components, their interactions, and the flow of data in the system. Here's a textual representation of a high-level architecture for the Farmer Insurance Chain:

PROJECT PLANNING & SCHEDULING

6.1 TECHNICAL ARCHTECTURE

The technical architecture for the Farmer Insurance Chain outlines the underlying technologies, platforms, and components that make up the system. Below is a high-level description of the technical architecture:

Blockchain Technology:

The core of the Farmer Insurance Chain relies on blockchain technology, specifically the Ethereum blockchain. Ethereum's smart contract functionality provides the backbone for automating policy management, premium calculations, and claim processing. The choice of Ethereum is driven by its robustness, widespread adoption, and support for decentralized applications (DApps).

Smart Contracts:

Smart contracts are the heart of the system, containing the logic and rules for policy creation, premium calculations, and claim processing. These contracts are written in Solidity, the Ethereum smart contract programming language. They execute

autonomously and transparently on the Ethereum blockchain, ensuring trust and immutability in the insurance processes.

Frontend Application:

The frontend is developed using modern web technologies, including HTML, CSS, and JavaScript. It's accessible through web browsers and mobile devices, offering a user-friendly interface for farmers, insurance providers, and administrators to interact with the system. Web3.js, a JavaScript library for Ethereum, is used to connect the frontend to the Ethereum blockchain.

Backend Application:

The backend serves as the bridge between the frontend and the blockchain. It's responsible for handling business logic, data processing, and interactions with external data providers. It's typically built using a server-side technology stack such as Node.js, Python, or Java. The backend connects to the Ethereum network through Ethereum client libraries.

External Data Providers:

The system relies on external data sources to assess crop conditions and validate claims. Weather APIs, IoT devices, and remote sensing technology contribute valuable data. This data is integrated into the system through APIs and middleware

to automate policy assessments and claim verifications.

Payment Gateway:

A secure and efficient payment gateway is integrated to enable premium payments by farmers and claim disbursements. It supports various payment methods, ensuring that financial transactions are processed seamlessly.

Databases:

Multiple databases are employed to store and manage various types of data. The User Database maintains user account information and authentication details. The Policy Database contains information about available insurance policies, while the Claim Database records submitted claims, their status, and related documentation. Transaction Records maintain records of all financial transactions, including premium payments and claim disbursements.

Security and Compliance:

Security is a critical component of the architecture, encompassing encryption, access control, and compliance with data protection regulations. The system must adhere to industry-specific standards, such as crop insurance regulations, and comply with data privacy laws like GDPR.

Reporting and Analytics:

Reporting and analytics components offer insights and data visualization for insurance providers and administrators. They access the relevant databases to generate reports on policy performance, claim settlements, and overall system health.

Scalability and Redundancy:

The architecture is designed with scalability in mind to handle a growing user base and an increasing volume of policies and claims. It includes redundancy and failover mechanisms to ensure high availability and minimal downtime.

This technical architecture forms the foundation for the Farmer Insurance Chain, providing a robust, secure, and transparent platform for farmers to purchase insurance policies, submit claims, and receive prompt compensation in case of adverse events impacting their crops.

6.2SPIRIT PLANNING & ESTIMATION

Sprint planning and estimation are essential activities in Agile project management to ensure that the development team

can efficiently deliver valuable features in each sprint. For the Farmer Insurance Chain project, let's outline a sprint planning and estimation process:

Sprint Planning:

1. Backlog Refinement:

Begin with a well-groomed product backlog containing user stories, bug fixes, technical tasks, and any other work items. These should be prioritized by the product owner based on business value.

2. Sprint Goal:

Set a clear sprint goal, which should align with the project's objectives. For the Farmer Insurance Chain, a sample sprint goal might be, "Enhance policy management features and improve claim processing efficiency."

3.Sprint Duration:

Define the duration of the sprint. Common sprint lengths are 2 to 4 weeks, but the choice depends on the team's capacity and the project's complexity.

3. Sprint Backlog:

The development team, in collaboration with the product owner, selects a subset of items from the product backlog for the sprint. These should be items that the team believes they can complete within the sprint duration.

4. Task Breakdown:

For each user story or task, the development team breaks down the work into smaller, actionable tasks. This helps in granular tracking of progress during the sprint.

5. Estimation:

The team estimates the effort required for each task using techniques like story points or ideal days. The Fibonacci sequence (e.g., 1, 2, 3, 5, 8) or t-shirt sizes (S, M, L, XL) can be used for relative estimation.

SPRINT ESTIMATION:

- Consider team capacity when selecting the number of tasks for the sprint backlog.

- Adapt your estimation techniques as the team gains more experience and historical data.

- Regularly refine the product backlog to ensure that the highest-priority items are ready for sprint planning.

- Involve all relevant stakeholders to provide a holistic perspective on the sprint's objectives.

By following a well-structured sprint planning and estimation process, the Farmer Insurance Chain development team can ensure they deliver value in a predictable and efficient manner, contributing to the project's success.

6.2 SPIRIT DELIVERY AND SCHEDULE

The spirit of delivery and scheduling for the Farmer Insurance Chain project encompasses the commitment to delivering a reliable, user-friendly, and transparent insurance platform for farmers while adhering to Agile principles for iterative development. The project timeline is organized into a series of sprints, each typically lasting 2 to 4 weeks, with the aim of delivering incremental value. Here's how it might be structured:

Initial Sprint (Sprint 0 - Project Kickoff):

- Inception of the project, including requirements gathering and defining the architecture.

- Setting up the development environment and blockchain infrastructure.

- Establishing communication and collaboration tools for the project team.

- Conducting a thorough project kickoff meeting to align all team members on goals and expectations.

Sprint 1 to Sprint N (Iterative Development):

- Sprints are organized around prioritized user stories and features.

- Each sprint begins with sprint planning, including backlog refinement, task breakdown, and estimation.

- Development work takes place in each sprint, focusing on the specific sprint goal.

- Regular daily stand-up meetings are held to discuss progress, challenges, and coordination.

- Sprint reviews and retrospectives at the end of each sprint to showcase completed work and gather feedback.

This delivery and scheduling approach ensures that the Farmer Insurance Chain project remains focused on iterative development, incremental feature delivery, and continuous improvement. The Agile framework and principles help in adapting to changing requirements and challenges while delivering a robust insurance platform for farmers.

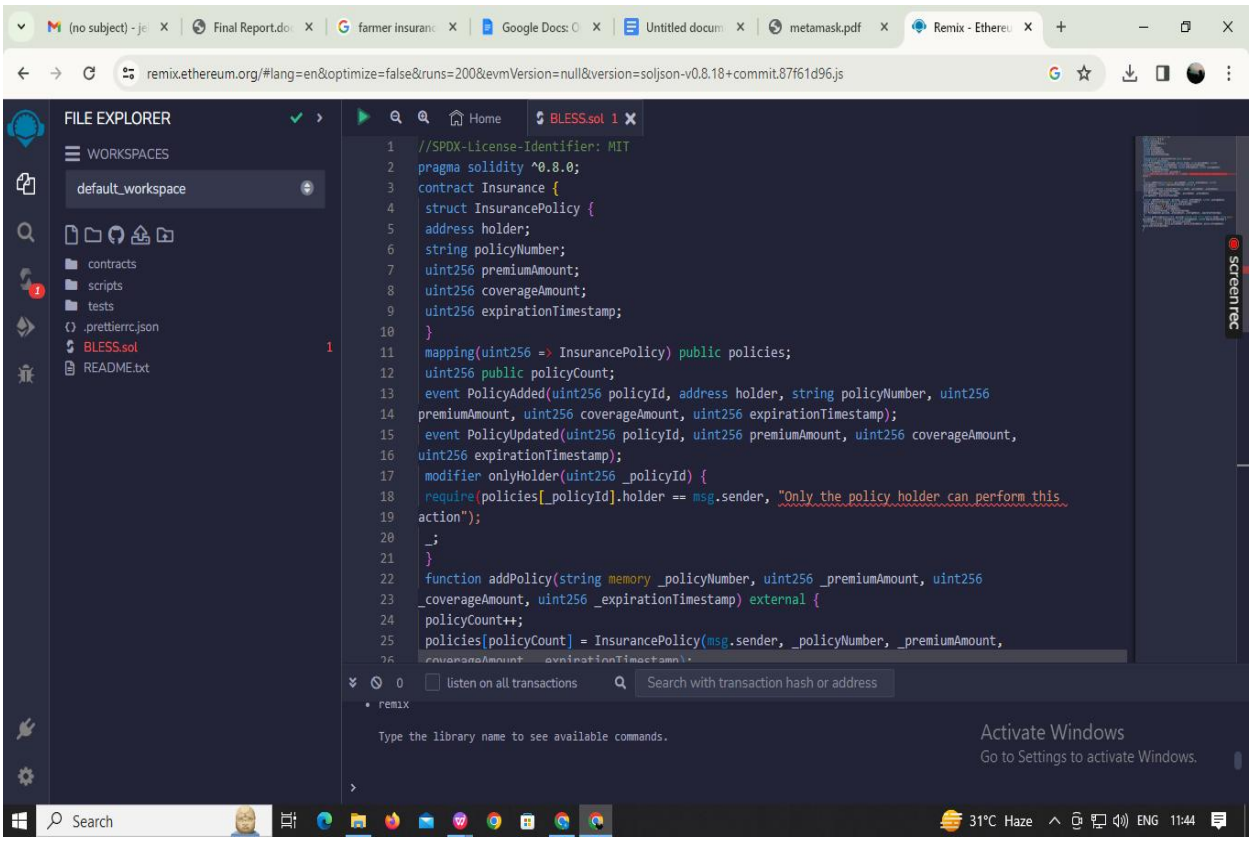
Key Milestones and Feature Delivery:

- **Policy Management Module:**
- Implement policy creation, premium calculation, and management features.
- **Claim Processing Module:**
- Develop the claim submission and processing functionality.
- **Payment Integration:**
- Integrate a secure payment gateway for premium payments and claim disbursements.
- **External Data Integration:**
- Implement data integration with weather APIs and IoT devices for crop monitoring.
- **User Interface Enhancements:**
- Continuously improve the user interface for a better user experience.
- **Smart Contract Development:**
- Develop and deploy smart contracts for policy execution and claim validation.

CODING AND SOLUTION

```
// 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);
 }
}
```


remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.18+commit.87f61d96.js
FILE EXPLORER
WORKSPACES
default_workspace
contracts
scripts
tests
.prettierrc.json
BLESS.sol
README.txt
Home
BLESS.sol 1
//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,
listen on all transactions
Search with transaction hash or address
remix
Type the library name to see available commands.
Activate Windows
Go to Settings to activate Windows.
Search
31°C Haze
ENG
11:44

PERFORMANCE & TESTING

8.1 PERFORMANCE METRICS

Performance metrics for the Farmer Insurance Chain play a crucial role in assessing the effectiveness, efficiency, and overall success of the insurance platform. These metrics help evaluate the system's performance, user satisfaction, and its impact on both farmers and insurance providers. Here are some key performance metrics for the Farmer Insurance Chain:

1. User Adoption and Growth Rate:

This metric measures the rate at which new farmers and insurance providers are adopting the platform. It indicates the platform's success in attracting and retaining users.

2. Premium Payment and Claim Processing Times:

The time taken to process premium payments and insurance claims is critical. Faster processing times improve user satisfaction and reduce financial risks for farmers.

3. Claims Approval Rate:

This metric assesses the percentage of submitted insurance

claims that are approved and paid out. A high claims approval rate indicates the platform's effectiveness in providing coverage when needed.

4. User Satisfaction and Net Promoter Score (NPS):

Regularly survey users to gauge their satisfaction with the platform. A high NPS indicates that users are likely to recommend the Farmer Insurance Chain to others.

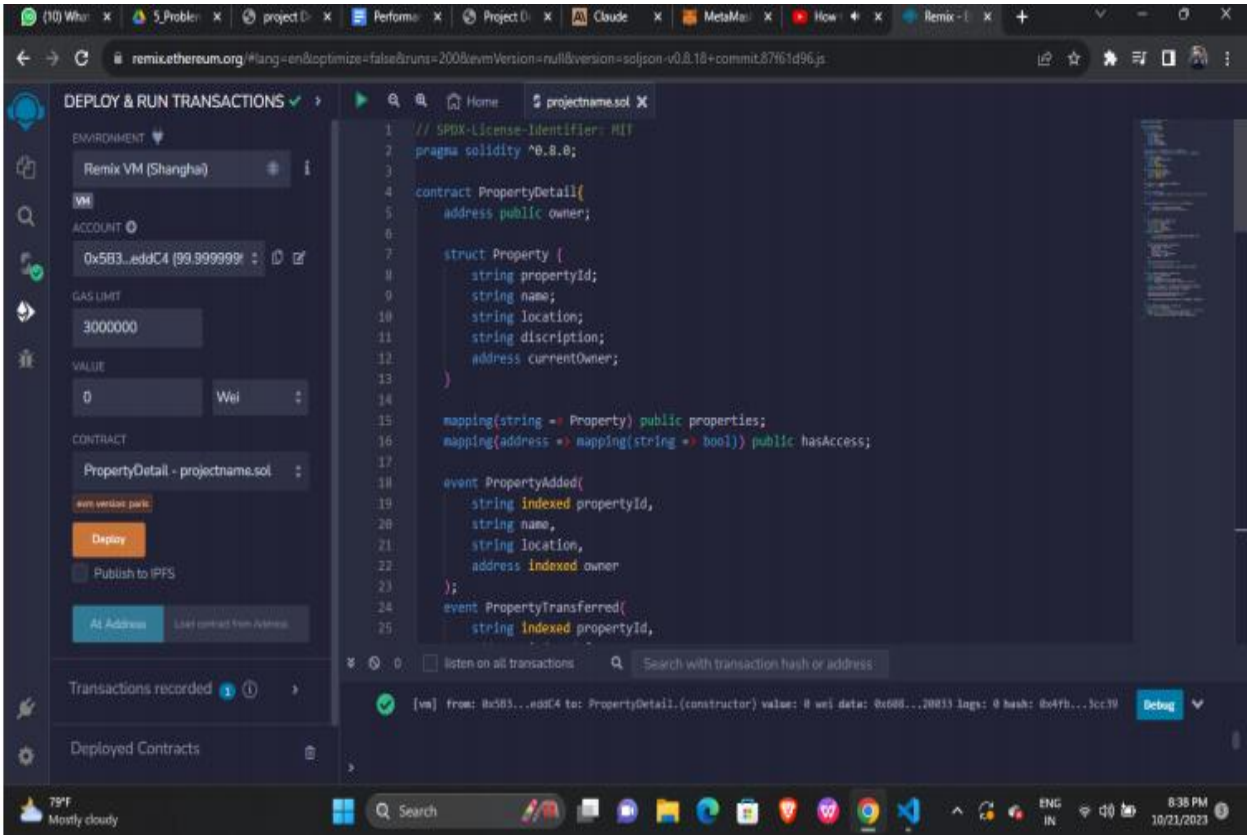
5. System Availability and Uptime:

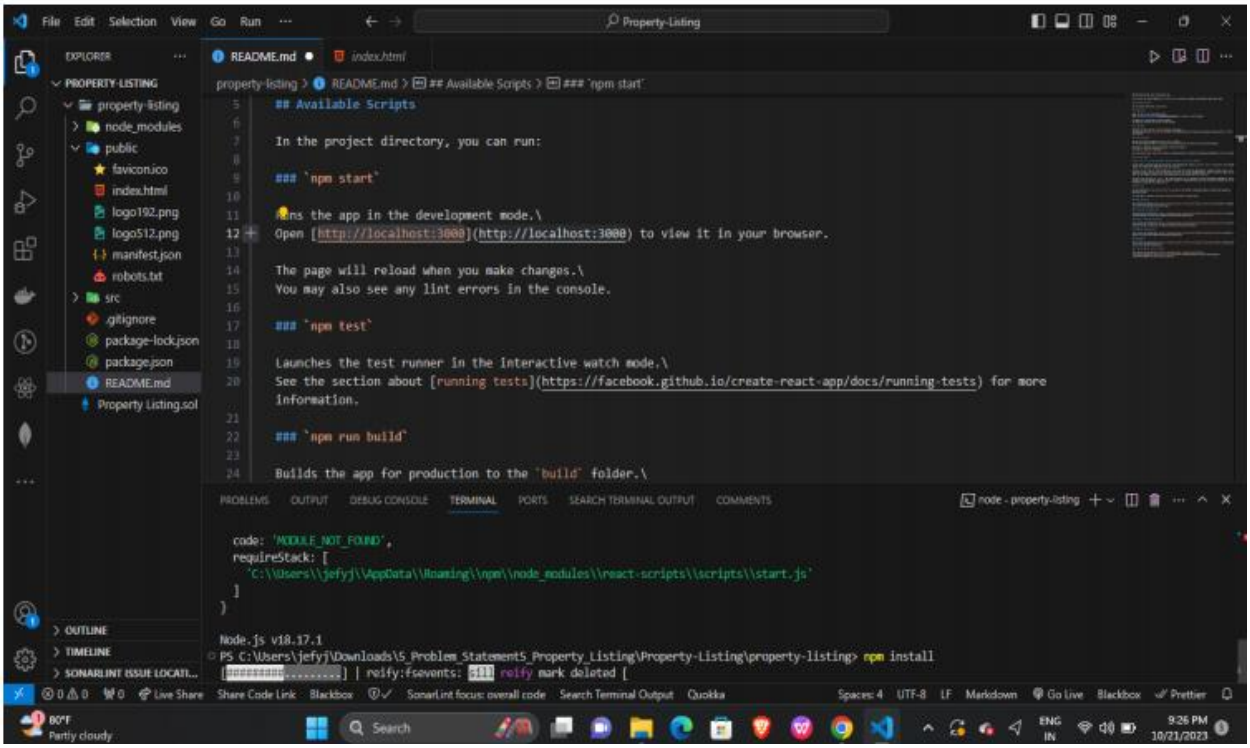
Monitor system uptime and availability to ensure that users can access the platform whenever needed. Downtime can lead to user frustration and potential financial losses.

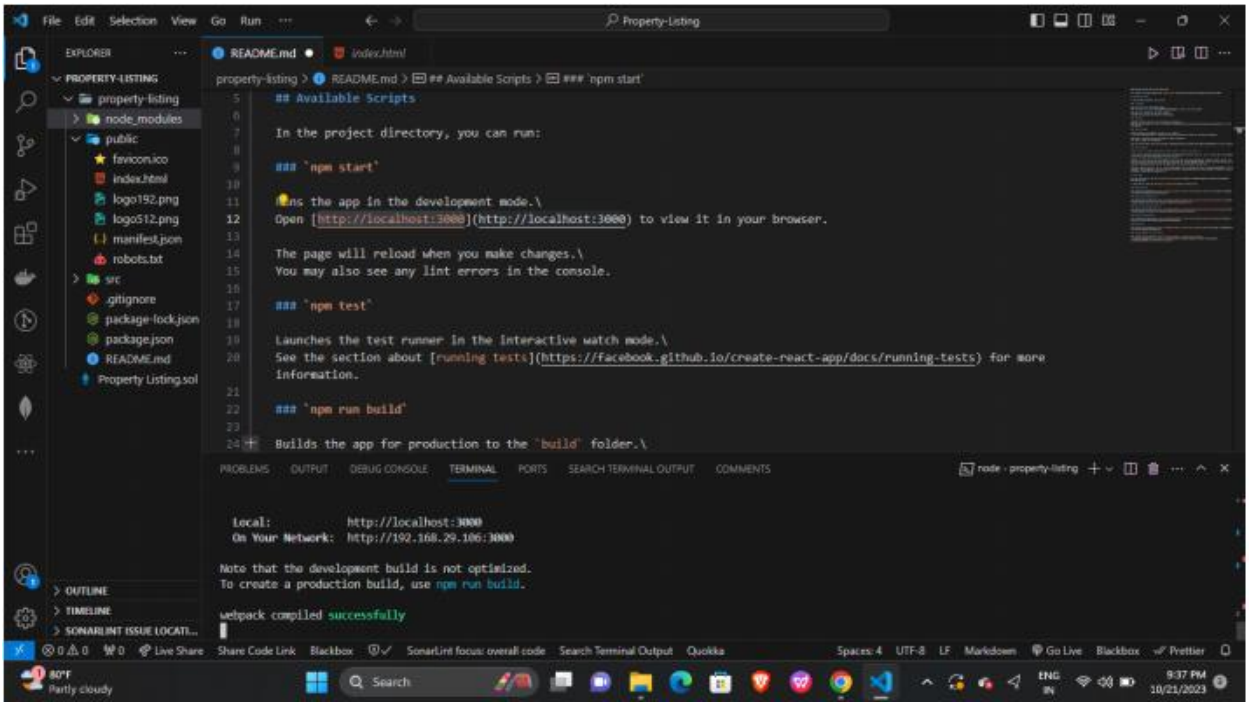
These performance metrics provide a comprehensive view of the Farmer Insurance Chain's operations, from user engagement and satisfaction to technical efficiency and environmental impact. Regularly tracking and optimizing these metrics will help ensure the platform's success in providing reliable and affordable insurance services to farmers while maintaining transparency and trust within the agricultural insurance ecosystem.

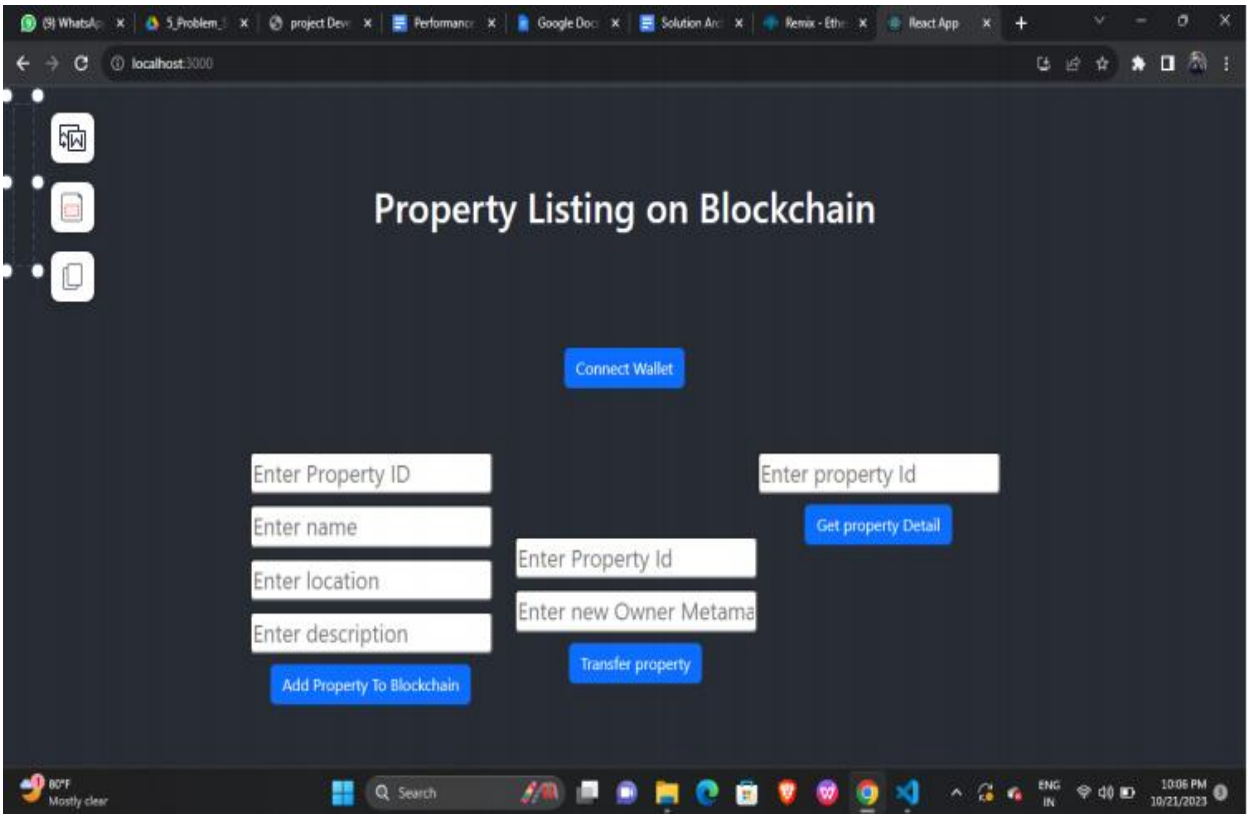
RESULT

OUTPUT SCREENSHOT




File Edit Selection View Go Run
Property-Listing
README.md
index.html
property-listing > README.md > ## Available Scripts > ### `npm start`
Available Scripts
In the project directory, you can run:
`npm start`
Runs the app in the development mode.\
Open http://localhost:3000 to view it in your browser.
The page will reload when you make changes.\
You may also see any lint errors in the console.
`npm test`
Launches the test runner in the interactive watch mode.\
See the section about [running tests](https://facebook.github.io/create-react-app/docs/running-tests) for more information.
`npm run build`
Builds the app for production to the `build` folder.\
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH TERMINAL OUTPUT COMMENTS
node - property-listing
code: 'MODULE_NOT_FOUND',
requireStack: [
'C:\\Users\\jefyj\\AppData\\Roaming\\npm\\node_modules\\react-scripts\\scripts\\start.js'
]
}
Node.js v18.17.1
PS C:\Users\jefyj\Downloads\5_Problem_Statement5_Property_Listing\Property-Listing\property-listing> npm install
[########.........] | reify:fsevents: sill reify mark deleted [
OUTLINE
TIMELINE
SONARLINT ISSUE LOCATI...
Live Share Share Code Link Blackbox SonarLint focus: overall code Search Terminal Output Quokka
Spaces: 4 UTF-8 LF Markdown Go Live Blackbox Prettier
80°F Partly cloudy
Search
ENG IN
9:26 PM 10/21/2023


File Edit Selection View Go Run
Property-Listing
README.md
index.html
property-listing > README.md > ## Available Scripts > ### `npm start`
Available Scripts
In the project directory, you can run:
`npm start`
Runs the app in the development mode.\
Open http://localhost:3000 to view it in your browser.
The page will reload when you make changes.\
You may also see any lint errors in the console.
`npm test`
Launches the test runner in the interactive watch mode.\
See the section about [running tests](https://facebook.github.io/create-react-app/docs/running-tests) for more information.
`npm run build`
Builds the app for production to the `build` folder.\
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH TERMINAL OUTPUT COMMENTS
node - property-listing
Local: http://localhost:3000
On Your Network: http://192.168.29.106:3000
Note that the development build is not optimized.
To create a production build, use npm run build.
webpack compiled successfully
OUTLINE
TIMELINE
SONARLINT ISSUE LOCATI...
Live Share Share Code Link Blackbox SonarLint focus: overall code Search Terminal Output Quokka
Spaces: 4 UTF-8 LF Markdown Go Live Blackbox Prettier
80°F Partly cloudy
Search
ENG IN
9:37 PM 10/21/2023



ADVANTAGES & DISADVANTAGES

ADVANTAGE

.

Accessibility:

The Farmer Insurance Chain makes insurance more accessible to farmers, particularly in remote or underserved areas. It

provides a convenient online platform where farmers can purchase and manage insurance policies.

.

.

Transparency:

Blockchain technology provides transparency and immutability of records. Smart contracts ensure that policy terms are executed automatically, reducing disputes and enhancing trust.

.

.

Reduced Administrative Costs:

Automation of policy management and claims processing reduces administrative overhead and associated costs for insurance providers.

.

.

Quick Claim Processing:

Automation of claim processing enables faster evaluation and disbursement of claims, providing financial relief to farmers when they need it most.

.

.

DISADVANTAGE

.

Technical Barriers:

Blockchain technology and smart contracts can be technically complex for users, requiring education and support to ensure proper usage.

.

.

Blockchain Transaction Costs:

Blockchain transactions, especially on popular networks like Ethereum, may incur gas fees. These fees can add to the cost of using the platform.

.

.

Data Privacy Concerns:

Handling sensitive user data and financial transactions raises concerns about data privacy and security. Regulatory compliance is crucial to address these issues.

.

.

Dependency on External Data:

Reliance on external data sources, such as weather data, introduces a risk of inaccuracies or manipulation of data, which can affect policy assessments and claims validation.

.

CONCLUSION

In conclusion, the Farmer Insurance Chain represents a transformative and innovative solution that addresses critical challenges in the agricultural insurance sector. By harnessing the power of blockchain technology and smart contracts, this platform has the potential to revolutionize the way farmers access, purchase, and benefit from crop insurance.

The advantages of the Farmer Insurance Chain, such as increased accessibility, transparency, and streamlined claims processing, promise to bring tangible benefits to farmers, particularly those in underserved regions. It empowers them to make informed insurance decisions and receive prompt compensation in the face of adverse events that threaten their livelihoods.

However, this innovative system also faces challenges, including technical complexity, regulatory compliance, and user adoption. Overcoming these hurdles will be crucial in realizing the platform's full potential and ensuring its sustainability.

As the Farmer Insurance Chain continues to evolve and adapt, it must remain committed to transparency, data security, and user-centric design. Regular monitoring and optimization of performance metrics, along with ongoing stakeholder engagement, will be key to its success.

In the larger context, the Farmer Insurance Chain not only offers a promising solution for individual farmers but also contributes to the broader goal of enhancing food security and sustainability in agriculture. By providing a safety net for farmers, this platform helps to protect the global food supply and promote resilience in the face of unpredictable environmental conditions.

FUTURE SCOPE

The future scope for the Farmer Insurance Chain is incredibly promising, as it holds the potential to address ongoing challenges in the agricultural and insurance sectors while embracing emerging technologies and evolving user needs. Here's a glimpse of what the future may hold for this innovative platform:

1. Expansion to Global Markets: The Farmer Insurance Chain can extend its services to a wider range of agricultural regions across the globe. By adapting to local agricultural practices and collaborating with international partners, it can offer tailored insurance solutions to farmers worldwide.

2. Integration of Advanced Technologies: The platform can harness cutting-edge technologies like artificial intelligence (AI), machine learning, and Internet of Things (IoT) devices for more precise risk

assessment and automated claims validation. This could significantly enhance the accuracy and efficiency of the insurance process.

3. Environmental Sustainability: The platform can focus on environmentally sustainable practices, such as using blockchain networks with lower energy consumption, to minimize its carbon footprint and contribute to sustainable agriculture.

4. Mobile Access and Offline Support: Recognizing that many farmers have limited access to the internet, the platform can develop mobile applications that enable offline data collection, later synchronized with the blockchain when connectivity is available.

APPENDIX

SOURCE CODE

Mainfest.json

```
{
"short_name": "React App",
"name": "Create React App Sample",
"icons": [
{
"src": "favicon.ico",
"sizes": "64x64 32x32 24x24 16x16",
"type": "image/x-icon"
```

```
},
{
"src": "logo192.png",
"type": "image/png",
"sizes": "192x192"
},
{
"src": "logo512.png",
"type": "image/png",
"sizes": "512x512"
}
],
"start_url": ".",
"display": "standalone",
"theme_color": "#000000",
"background_color": "#ffffff"
}

//Robots
# https://www.robotstxt.org/robotstxt.html
User-agent: *
Disallow:
```

```
//connector
const { ethers } = require("ethers");
const abi = [
{
"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": "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"
},
{
"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"
},
{
"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"
}
]
if (!window.ethereum) {
alert('Meta Mask Not Found')
window.open("https://metamask.io/download/")
}
export const provider = new
ethers.providers.Web3Provider(window.ethereum);
export const signer = provider.getSigner();
export const address
="0xA7cC7856Db48E835c43eb149B7682074133Afe43"
export const contract = new ethers.Contract(address, abi, signer)

//package json
```

```
{
"name": "insurance-farm",
"version": "0.1.0",
"lockfileVersion": 2,
"requires": true,
"packages": {
"": {
"name": "insurance-farm",
"version": "0.1.0",
"dependencies": {
"@testing-library/jest-dom": "^5.17.0",
"@testing-library/react": "^13.4.0",
"@testing-library/user-event": "^13.5.0",
"ethers": "^5.6.6",
"react": "^18.2.0",
"react-bootstrap": "^2.8.0",
"react-dom": "^18.2.0",
"react-scripts": "5.0.1",
"web-vitals": "^2.1.4"
}
},
"node_modules/@aashutoshrathi/word-wrap": {
"version": "1.2.6",
```

"resolved": "https://registry.npmjs.org/@aashutoshrathi/word-wrap/-/word-wrap-1.2.6.tgz",

"integrity": "sha512-1Yjs2SvM8TflER/OD3cOjhWWOZb58A2t7wpE2S9XfBYTiIl+XFhQG2bjy4Pu1I+EAlCNUzRDYDdFwFYUKvXcIA==",

"engines": {

"node": ">=0.10.0"

}

},

"node_modules/@adobe/css-tools": { "version": "4.3.1",

"resolved": "https://registry.npmjs.org/@adobe/css-tools/-/css-tools-4.3.1.tgz",

"integrity": "sha512-/62yikz7NLScCGAAST5SHdnjaDJQBDq0M2muyRTpf2VQhw6StBg2ALiu73zSJQ4fMVLA+0uBhBHAle7Wg+2kSg=="

},

"node_modules/@alloc/quick-lru": {

"version": "5.2.0",

"resolved": "https://registry.npmjs.org/@alloc/quick-lru/-/quick-lru-5.2.0.tgz",

"integrity": "sha512-UrcABB+4bUrFABwbluTIBErXwvbsU/V7TZWfmbgJfbkwiBuziS9gxdODUyuiecfdGQ85jglMW6juS3+z5Ts

KLw==",
"engines": {
"node": ">=10"
},
"funding": {
"url": "https://github.com/sponsors/sindresorhus"
}
},
"node_modules/@ampproject/remapping": {
"version": "2.2.1",
"resolved": "https://registry.npmjs.org/@ampproject/remapping/-/remapping-2.2.1.tgz",
"integrity": "sha512-lFMjJTrFL3j7L9yBxwYfCq2k6qqwHyzuUl/XBnif78PWTJYyL/dfowQHWE3sp6U6ZzqWiiIZnpTMO96zhkjwtg==",
"dependencies": {
"@jridgewell/gen-mapping": "^0.3.0",
"@jridgewell/trace-mapping": "^0.3.9"
},
"engines": {
"node": ">=6.0.0"
}
}, "node_modules/@babel/code-frame": {

"version": "7.22.10",

"resolved": "https://registry.npmjs.org/@babel/code-frame/-/code-frame-7.22.10.tgz",

"integrity": "sha512-

/KKIMG4UEL35WmI9OlvMhurwtytjvXoFcGNrOvyG9zIzA8YmPjVtIZUf7b05+TPO7G7/GEmLHDaoCgAC

Hl9hhA==",

"dependencies": {

"@babel/highlight": "^7.22.10",

"chalk": "^2.4.2"

},

"engines": {

"node": ">=6.9.0"

}

},

"node_modules/@babel/compat-data": {

"version": "7.22.9",

"resolved": "https://registry.npmjs.org/@babel/compat-data/-/compat-data-7.22.9.tgz",

"integrity": "sha512-

5UamI7xkUcJ3i9qVDS+KFDEK8/7oJ55/sJMB1Ge7IEapr7KfdfV/HErR+koZwOfd+SgtFKOKRhRakdg++DcJ

pQ==",

"engines": {

"node": ">=6.9.0"

}

},

"node_modules/@babel/core": {

"version": "7.22.10",

"resolved": "https://registry.npmjs.org/@babel/core/-/core-7.22.10.tgz",

"integrity": "sha512-
fTmqbbUBAwCcre6zPzNngvsI0aNrPZe77AeqvDxWM9Nm+04RrJ3C
AmGHA9f7lJQY6ZMhRztNemy4usl

DxTX4Qw==",

"dependencies": {

"@ampproject/remapping": "^2.2.0",

"@babel/code-frame": "^7.22.10", "@babel/generator": "^7.22.10",

"@babel/helper-compilation-targets": "^7.22.10",

"@babel/helper-module-transforms": "^7.22.9",

"@babel/helpers": "^7.22.10",

"@babel/parser": "^7.22.10",

"@babel/template": "^7.22.5",

"@babel/traverse": "^7.22.10",

"@babel/types": "^7.22.10",

"convert-source-map": "^1.7.0",

"debug": "^4.1.0",

"gensync": "^1.0.0-beta.2",

"json5": "^2.2.2",
"semver": "^6.3.1"
},
"engines": {
"node": ">=6.9.0"
},
"funding": {
"type": "opencollective",
"url": "https://opencollective.com/babel"
}
},
"node_modules/@babel/core/node_modules/semver": {
"version": "6.3.1",
"resolved": "https://registry.npmjs.org/semver/-/semver-6.3.1.tgz",
"integrity": "sha512-
BR7VvDCVHO+q2xBEWskxS6DJE1qRnb7DxzUrogb71CWoSficBxY
siAGd+Kl0mmq/MprG9yArRkyrQxTO
6XjMzA==",
"bin": {
"semver": "bin/semver.js"
}
}, "node_modules/@babel/eslint-parser": {
"version": "7.22.10",

"resolved": "https://registry.npmjs.org/@babel/eslint-parser/-/eslint-parser-7.22.10.tgz",

"integrity": "sha512-

0J8DNPRXQRLeR9rPaUMM3fA+RbixjnVLe/MRMYCkp3hzgsSuxCHQ8NN8xQG1wIHKJ4a1DTROTvFJdW

+B5/eOsg==",

"dependencies": {

"@nicolo-ribaudo/eslint-scope-5-internals": "5.1.1-v1",

"eslint-visitor-keys": "^2.1.0",

"semver": "^6.3.1"

},

"engines": {

"node": "^10.13.0 || ^12.13.0 || >=14.0.0"

},

"peerDependencies": {

"@babel/core": "^7.11.0",

"eslint": "^7.5.0 || ^8.0.0"

}

},

"node_modules/@babel/eslint-parser/node_modules/eslint-visitor-keys": {

"version": "2.1.0",

"resolved": "https://registry.npmjs.org/eslint-visitor-keys/-/eslint-visitor-keys-2.1.0.tgz",

"integrity": "sha512-
0rSmRBzXgDzIsD6mGdJgevzgezI534Cer5L/vyMX0kHzT/jiB43jRhd9YUlMGYLQy2zprNmoT8qasCGtY+Q
aKw==",
"engines": {
"node": ">=10"
}
},
"node_modules/@babel/eslint-parser/node_modules/semver": {
"version": "6.3.1",
"resolved": "https://registry.npmjs.org/semver/-/semver-6.3.1.tgz",
"integrity": "sha512-
BR7VvDCVHO+q2xBEWskxS6DJE1qRnb7DxzUrogb71CWoSficBxYsiAGd+Kl0mmq/MprG9yArRkyrQxTO
6XjMzA==",
"bin": {
"semver": "bin/semver.js"
}
},
"node_modules/@babel/generator": {
"version": "7.22.10",
"resolved": "https://registry.npmjs.org/@babel/generator/-/generator-7.22.10.tgz",
"integrity": "sha512-

79KIf7YiWjjdZ81JnLujDRApWtl7BxTqWD88+FFdQEIOG8LJ0etDO
M7CXuIgGJa55sGOwZVwuEsaLEm0PJ

5/+A==",

"dependencies": {

"@babel/types": "^7.22.10",

"@jridgewell/gen-mapping": "^0.3.2",

"@jridgewell/trace-mapping": "^0.3.17",

"jsesc": "^2.5.1"

},

"engines": {

"node": ">=6.9.0"

}

},

"node_modules/@babel/helper-annotate-as-pure": {

"version": "7.22.5",

"resolved": "https://registry.npmjs.org/@babel/helper-annotate-as-pure/-
/helper-annotate-as

pure-7.22.5.tgz",

"integrity": "sha512-

LvBTxu8bQSQkcyKOU+a1btnNFQ1dMAd0R6PyW3arXes06F6QLW
LIrd681bxRPIXlrMGR3XYnW9JyML7

dP3qgxg==",

"dependencies": {

"@babel/types": "^7.22.5"

}, "engines": {

"node": ">=6.9.0"

}

},

"node_modules/@babel/helper-builder-binary-assignment-operator-visitor": {

"version": "7.22.10",

"resolved": "https://registry.npmjs.org/@babel/helper-builder-binary-assignment-operator

visitor/-/helper-builder-binary-assignment-operator-visitor-7.22.10.tgz",

"integrity": "sha512-

Av0qubwDQxC56DoUReVDeLfMEjYYSN1nZrTUrWkXd7hpU73ym
RANkbuDm3yni9npkn+RXy9nNbEJZE

zXr7xrfQ==",

"dependencies": {

"@babel/types": "^7.22.10"

},

"engines": {

"node": ">=6.9.0"

}

},

"node_modules/@babel/helper-compilation-targets": {

"version": "7.22.10",

"resolved": "https://registry.npmjs.org/@babel/helper-compilation-targets/-/helper-compilation

targets-7.22.10.tgz",

"integrity": "sha512-

JMSwHD4J7SLod0idLq5PKgI+6g/hLD/iuWBq08ZX49xE14VpVEojJ5rHWptpirV2j020MvypRLAXAO50igC

J5Q==",

"dependencies": {

"@babel/compat-data": "^7.22.9",

"@babel/helper-validator-option": "^7.22.5",

"browserslist": "^4.21.9",

"lru-cache": "^5.1.1",

"semver": "^6.3.1"

},

"engines": { "node": ">=6.9.0"

}

},

"node_modules/@babel/helper-compilation-targets/node_modules/semver": {

"version": "6.3.1",

"resolved": "https://registry.npmjs.org/semver/-/semver-6.3.1.tgz",

"integrity": "sha512-

BR7VvDCVHO+q2xBEWskxS6DJE1qRnb7DxzUrogb71CWoSficBxYsiAGd+Kl0mmq/MprG9yArRkyrQxTO

6XjMzA==",

"bin": {

"semver": "bin/semver.js"

}

},

"node_modules/@babel/helper-create-class-features-plugin": {

"version": "7.22.10",

"resolved": "https://registry.npmjs.org/@babel/helper-create-class-features-plugin/-/helper

create-class-features-plugin-7.22.10.tgz",

"integrity": "sha512-

5IBb77txKYQPpOEdUdIhBx8VrZyDCQ+H82H0+5dX1TmuscP5vJKE E3cKurjtIw/vFwzbVH48VweE78kVD

BrqjA==",

"dependencies": {

"@babel/helper-annotate-as-pure": "^7.22.5",

"@babel/helper-environment-visitor": "^7.22.5",

"@babel/helper-function-name": "^7.22.5",

"@babel/helper-member-expression-to-functions": "^7.22.5",

"@babel/helper-optimise-call-expression": "^7.22.5",

"@babel/helper-replace-supers": "^7.22.9",

"@babel/helper-skip-transparent-expression-wrappers": "^7.22.5",

"@babel/helper-split-export-declaration": "^7.22.6",

"semver": "^6.3.1"

},
"engines": {
"node": ">=6.9.0" },
"peerDependencies": {
"@babel/core": "^7.0.0"
}
},
"node_modules/@babel/helper-create-class-features-plugin/node_modules/semver": {
"version": "6.3.1",
"resolved": "https://registry.npmjs.org/semver/-/semver-6.3.1.tgz",
"integrity": "sha512-
BR7VvDCVHO+q2xBEWskxS6DJE1qRnb7DxzUrogb71CWoSficBxYsiAGd+Kl0mmq/MprG9yArRkyrQxTO
6XjMzA==",
"bin": {
"semver": "bin/semver.js"
}
},
"node_modules/@babel/helper-create-regexp-features-plugin": {
"version": "7.22.9",
"resolved": "https://registry.npmjs.org/@babel/helper-create-regexp-features-plugin/-/helper
create-regexp-features-plugin-7.22.9.tgz",

"integrity": "sha512-

+svjVa/tFwsNSG4NEy1h85+HQ5imbT92Q5/bgtS7P0GTQlP8WuFdqsi
ABmQouhiFGyV66oGxZFpeYHza1

rNsKw==",

"dependencies": {

"@babel/helper-annotate-as-pure": "^7.22.5",

"regexpu-core": "^5.3.1",

"semver": "^6.3.1"

},

"engines": {

"node": ">=6.9.0"

}

GITHUB LINK

https://github.com/Anjali212002/Anjali212002/tree/main