Project Design Phase-I

Solution Architecture

Date	25 November 2023
Team ID	NM2023TMID11927
Project Name	FARMER INSURANCE CHAIN
Maximum Marks	4 Marks

Solution Architecture:

- **Smart Contracts:**
  Role: Deployed on the blockchain, smart contracts automate the execution of insurance policies, handling policy creation, claim processing, and payouts.
  Technology: Solidity (for Ethereum) or other blockchain-specific languages.
- **Blockchain Network:**
  Role: The underlying blockchain network where transactions are recorded and smart contracts are deployed.
  Technology: Ethereum, Binance Smart Chain, Hyperledger Fabric, or another suitable blockchain.
- **User Interface:**
  Role: Provides an interface for farmers to interact with the insurance chain, allowing them to create policies, check policy status, and file claims.
  Technology: Web framework (React, Angular, Vue.js), mobile app, or a combination.
- **Backend Server:**
  Role: Manages the communication between the frontend and the blockchain, handles business logic, and ensures the security of sensitive data.
  Technology: Node.js, Flask, Django, or any backend framework.

- **Database:**

  Role: Stores off-chain data, such as user information, policy details, and historical transactions.

  Technology: PostgreSQL, MongoDB, or another suitable database.

- **Identity Management:**

  Role: Manages user identity and access control, ensuring secure interactions with the insurance chain.

  Technology: Integration with decentralized identity solutions or traditional identity management systems.

  Oracles:

  Role: Fetches real-world data (e.g., weather information) and provides it to the smart contracts for decision-making.

  Technology: Chainlink, Oraclize, or custom oracles.

- **Workflow:**

  Policy Creation:

  Farmer interacts with the frontend to create an insurance policy.

  Backend server validates the input and sends a transaction to the smart contract to create a new policy.

- **Claim Filing:**

  In case of a loss, the farmer files a claim through the frontend.

  The backend server validates the claim and triggers a claim processing transaction on the smart contract.

- **Smart Contract Execution:**

  Smart contracts execute business logic, such as verifying the validity of a claim based on predefined conditions.

  Oracles may be used to fetch external data (e.g., weather conditions) to validate claims.

- **Payouts:**

  Once a claim is validated, the smart contract initiates the payout process.

  Payouts are automatically transferred to the farmer's account.

- **Data Storage:**

  Relevant data, such as policy details and transactions, is stored on-chain (smart contracts) and off-chain (database).

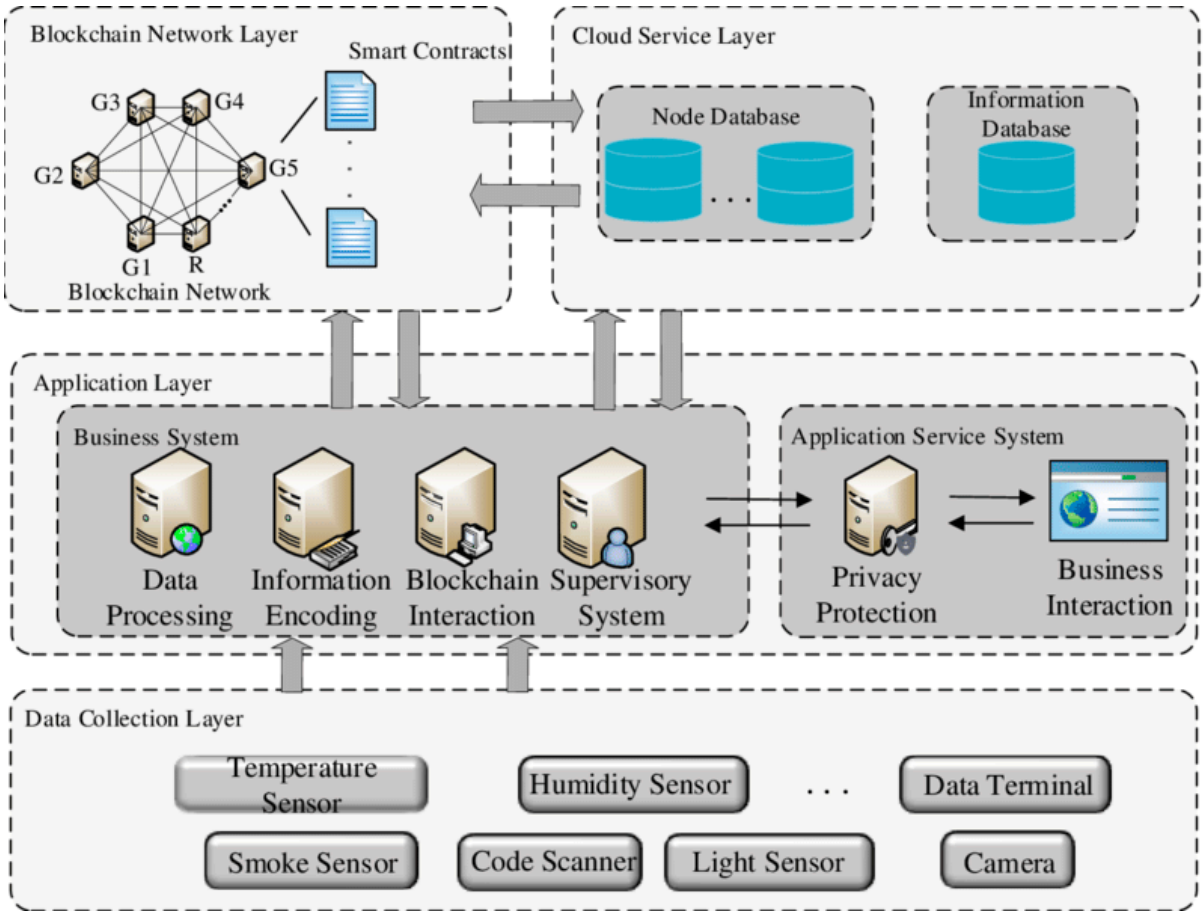
Security Considerations:

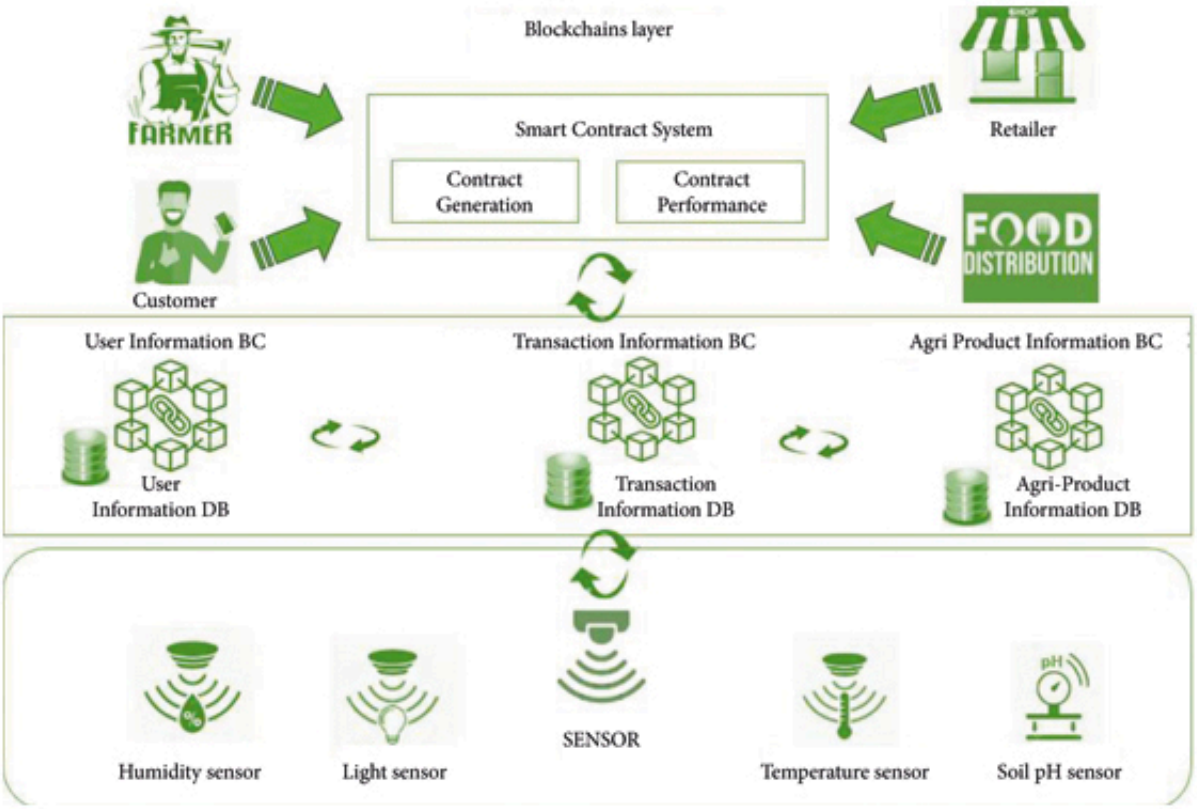
- **Immutable Ledger:**

  Blockchain's immutability ensures that once data is recorded, it cannot be tampered with.

- **Smart Contract Security:**
  Rigorous testing and auditing of smart contracts to prevent vulnerabilities.
- **Secure Communication:**
  Encryption and secure communication protocols to protect data during transmission.
- **Access Control:**
  Proper access controls to restrict unauthorized access to sensitive functions and data.
- **Privacy:**
  Consideration of privacy concerns, especially when handling personally identifiable information.
- **Scalability:**
  Choose a blockchain platform that meets scalability requirements.
- **Backend Scalability:**
  Horizontal scaling of backend servers to handle increased user interactions.
- **Integration:**
  Integration with External Systems:
  Integration with external data sources, such as weather APIs, for risk assessment.
- **Payment Integration:**
  Integration with payment gateways for premium payments and claim payouts.
- **Compliance:**
  Regulatory Compliance:
  Ensure compliance with local regulations and insurance industry standards.
  Smart Contract Audits:
  Periodic audits of smart contracts to ensure compliance and security.
- **Monitoring and Analytics:**
  Transaction Monitoring:
  Real-time monitoring of transactions on the blockchain.
- **Analytics Dashboard:**
  Creation of an analytics dashboard to track key metrics and performance indicators.

Solution Architecture Diagram:





Prerequisite

1 download node.js : Node.js

2 download vs code: **Li4nk**

3 download metamask : https://metamask.io/

Steps to complete the project

Step 1:-

1. Open the Zip file and download the zip file.

   Extract all zip files

Step 2 :

1. Open vs code in the left top select open folder. Select extracted file and open .
2. Select the projectname.sol file and copy the code.
3. Open the remix ide platform and create a new file by giving the name of projectname.sol and paste the code which you copied from vs code.
4. Click on solidity compiler and click compile the projectname.sol
5. Deploy the smart contract by clicking on the deploy and run transaction.

6. select injected provider - MetaMask. In environment

7. Click on deploy. Automatically MetaMask will open and give confirmation. You will get a pop up click on ok.

8. In the Deployed contract you can see one address copy the address.

9. Open vs code and search for the connector.js. In contract.js you can paste the address at the bottom of the code. In export const address.

10. Save the code.

Step 3:

open file explorer

1. Open the extracted file and click on the folder.

2. Open src, and search for utiles.

3. You can see the frontend files. Select all the things at the top in the search bar by clicking alt+ A. Search for cmd

4. Open cmd enter commands

    npm install

    npm bootstrap

    npm start

5. It will install all the packages and after completing it will open {LOCALHOST IP ADDRESS} copy the address and open it to chrome so you can see the frontend of your project.