A Blockchain-Based Fault Detection and Logging System

Digital Protection in Power Systems

Project Report

Submitted by

G. Rakesh	M240386EE
C. Charan Kumar	M241012EE

Under the guidance of

Dr. Deepak M

M tech Power systems

Department of Electrical Engineering

National Institute of Technology, Calicut

ABSTRACT

This project introduces a blockchain-based fault detection and logging system designed to enhance the reliability and transparency of power system monitoring. Traditional fault recording mechanisms often suffer from centralized vulnerabilities, including data tampering risks and single points of failure. To address these limitations, we propose a decentralized solution leveraging Ethereum smart contracts to create an immutable, auditable ledger of electrical faults. The system automatically detects and logs critical events such as over currents, short circuits, and voltage fluctuations while providing real-time alerts and historical analytics.

The core innovation lies in the integration of smart contracts with existing power infrastructure through a Python-based interface. Fault data—including bus identifiers, fault types, current magnitudes, and precise timestamps—is permanently recorded on the blockchain, ensuring cryptographic integrity and non-repudiation through reporter address tracking. A configurable current threshold (default: 100A) enables automatic overcurrent detection, while permissioned functions allow authorized operators to update system parameters.

Developed using Solidity for the smart contract logic, Hardhat for local blockchain testing, and Web3.py for grid integration, this system demonstrates three key advantages: (1) Tamper-proof data storage through blockchain immutability, (2) Decentralized verification eliminating single points of control, and (3) Programmable automation of fault responses via Python scripts. The implementation includes a full-featured API for querying faults, generating reports, and triggering maintenance workflows.

CONTENTS

List of Figures

CHAPTER 1

INTRODUCTION

1.1 Introduction

Modern power systems demand robust fault monitoring solutions, yet conventional centralized logging systems suffer from vulnerabilities including data tampering risks, single points of failure, and inefficient manual processes. To overcome these limitations, we present a blockchain-powered fault detection and logging system that leverages smart contract technology to establish a secure, decentralized record of electrical anomalies. This innovative framework captures critical fault data such as overcurrent events, short circuits, and voltage deviations while maintaining real-time monitoring capabilities through adjustable thresholds. System architecture combines Ethereum smart contracts with existing power infrastructure through Python-based interfaces, creating an immutable ledger of fault occurrences. Each recorded event includes comprehensive details - bus identifiers, fault classifications, current measurements, and precise timestamps - all cryptographically secured through blockchain technology. The integration of reporter addresses provides undeniable proof of origin, establishing a verifiable audit trail for compliance and diagnostic purposes. Built using Solidity for contract development and Hardhat for testing, this solution delivers multiple advantages: tamper-resistant data preservation through blockchain's inherent immutability, distributed verification that eliminates centralized control points, and automated fault response mechanisms. Practical implementations span from real-time grid monitoring to predictive maintenance scheduling, offering energy providers a transparent and secure platform for fault management that simultaneously strengthens grid reliability and satisfies regulatory mandate.

The modular nature of this system facilitates future enhancements, including potential integration with IoT sensor networks and advanced diagnostic tools. By merging blockchain's security features with power system requirements, this approach represents a significant advancement in electrical fault monitoring, providing utilities with an unprecedented combination of data integrity, operational transparency, and system resilience.

1.2 Objectives

Objective 1: Develop a Secure Smart Contract for Fault Logging

- Design and deploy a Solidity smart contract on Ethereum Virtual Machine (EVM) that:
    - Stores fault data (bus ID, fault type, current, timestamp) immutably
    - Implements threshold checking (default 100A)
    - Tracks reporter addresses for accountability
    - Provides query functions for fault history.

Objective 2: Create Python Interface for Smart Contract Interaction.

- Build a Python program using Web3.py that:
    - Connects to local Hardhat blockchain network
    - Submits new fault data to the smart contract
    - Retrieves and displays historical fault records
    - Handles transaction signing and gas estimation

Objective 3: Test End-to-End System Functionality

- Implement comprehensive testing that verifies:
    - Smart contract correctly logs and retrieves faults
    - Python program successfully interacts with contract
    - Threshold detection works as designed
    - All components work on local Hardhat network
    - Error cases are properly handled

1.3 Description

This project develops an innovative blockchain-based solution for detecting and logging faults in power systems, addressing critical challenges in grid monitoring and maintenance. Traditional fault recording systems often rely on centralized databases that are vulnerable to tampering, single points of failure, and lack transparency. Our decentralized approach leverages Ethereum smart contracts to create an immutable, transparent ledger of electrical faults including over currents, short circuits, and other anomalies. The system consists of three core components: a Solidity smart contract for fault recording and threshold monitoring, a Python interface for seamless integration with existing power infrastructure, and a Hardhat-based testing framework for validation.

The smart contract serves as the backbone of the system, permanently storing fault details such as bus identifiers, fault types, current measurements, and precise timestamps on the blockchain. It implements configurable current thresholds (defaulting to 100A) to automatically detect abnormal conditions, while cryptographic signatures ensure each entry is verifiable and tamper-proof. The contract maintains a complete historical record of all faults, enabling comprehensive auditing and analysis. This decentralized approach eliminates reliance on any single authority and provides utilities with an indisputable record of grid events.

For practical integration, we developed a Python-based interface using Web3.py that connects power system sensors and control systems to the blockchain. This interface handles all interactions with the smart contract, including fault submission, threshold configuration, and data retrieval. It features robust error handling and transaction management, ensuring reliable operation even in unstable network conditions. The Python component also includes a command-line interface for manual operations and debugging, making the system accessible to both automated systems and human operators.

To validate the solution, we implemented a complete testing environment using Hardhat, allowing for comprehensive verification of all functionalities on a local blockchain network. This includes unit tests for the smart contract, integration tests for the Python interface, and end-to-end tests of the complete system. The modular architecture ensures the solution can be extended with additional features like IoT sensor integration or advanced analytics while maintaining compatibility with existing power system infrastructure. By combining blockchain's security advantages with practical power system requirements, this project demonstrates a significant advancement in grid monitoring technology that enhances reliability, transparency, and maintenance efficiency.

Comparison of Traditional vs. Blockchain-Based Fault Detection and Logging Systems:

Feature	**Traditional System**	**Blockchain-Based System**
Data Integrity	Vulnerable to tampering or manual errors	Immutable,records (cryptographically secured)
Auditability	Requires manual log verification	Transparent, timestamped history (via TX hashes)
Decentralization	Centralized database (single point of failure)	Distributed across nodes (no single authority)
Threshold Enforcement	Manual checks or basic software rules	Automated via smart contracts (isOverCurrent)
Access Control	Role-based in database (can be bypassed)	Cryptographic signatures (private key required)
Alert Generation	Email/API calls (delays possible)	Real-time events (FaultLogged emitted)
Storage Costs	Low (local/server storage)	lower (gas fees for on-chain storage)
Compliance	Periodic audits needed	Built-in regulatory compliance (permanent logs)
Maintenance	Requires backups and upgrades	Self-sustaining (no admin intervention)

Fig 1.3: Comparison of Traditional vs. Blockchain

CHAPTER 2
METHODOLOGY

2.1 Smart Contract Development: (Ethereum)

The project begin with designing and implementing a Solidity-based smart contract to serve as the core fault logging mechanism. The contract was structured to store critical fault parameters including bus ID, fault type, current value, and timestamp while enforcing data immutability through blockchain's inherent cryptographic security.

Key functions were implemented to handle fault recording (addFault), threshold validation (isOverCurrent), and data retrieval (getFault).

The contract incorporates event logging for efficient off-chain monitoring and uses uint256 for gas-efficient numerical storage. Special attention was given to access control, with designated owner privileges for threshold adjustments while maintaining public read access for audit purposes.

The development followed an iterative process using Hardhat's testing environment, with each function undergoing unit tests to verify proper data handling and edge case management before deployment on a local Ethereum Virtual Machine (EVM) network.

2.2 Python Interface Implementation:

A robust Python interface was developed using Web3.py to bridge traditional power systems with the blockchain infrastructure. The implementation focused on three key aspects: First, establishing secure connections to the blockchain network through HTTP providers with proper transaction signing using private key management. Second, creating wrapper functions that abstract complex blockchain interactions such as gas estimation and transaction receipt handling into simple commands like log_fault() and get_fault_history(). Third, building error handling mechanisms for common scenarios like network timeouts or insufficient gas. The interface was designed with modularity in mind, allowing easy adaptation to different blockchain networks or smart contract versions. Particular attention was given to data formatting, ensuring seamless conversion between Python data types and Solidity's strict type requirements, especially for string handling and large integer values.

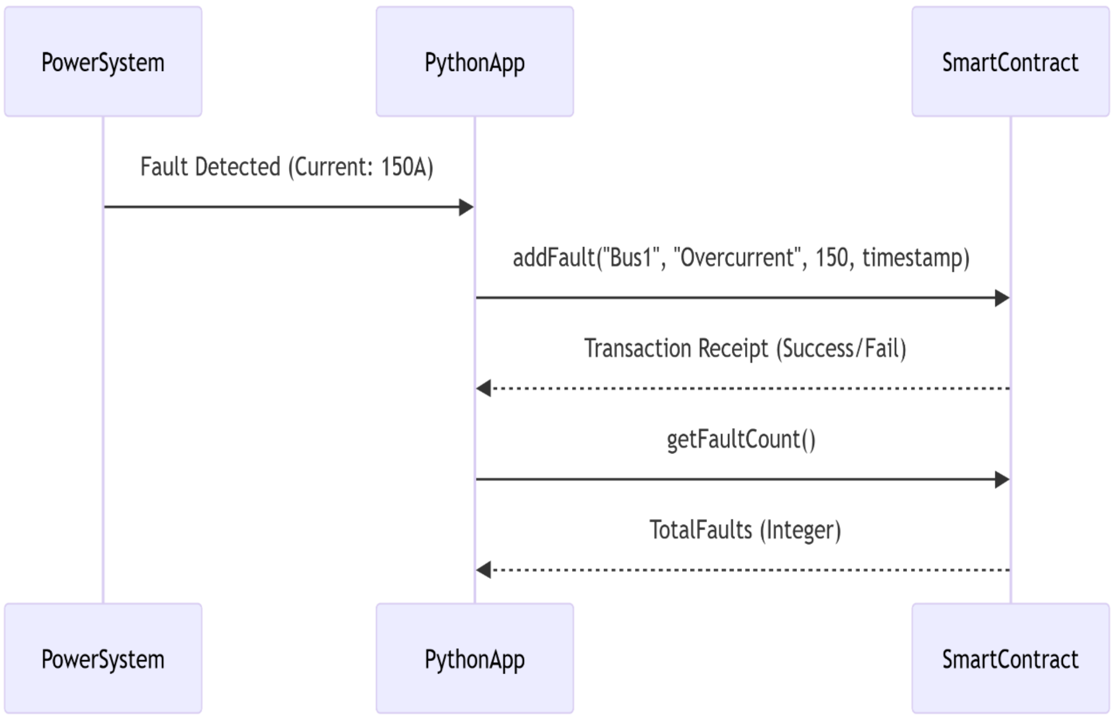
2.3 System Testing and Validation

A comprehensive testing framework was implemented across three layers: unit, integration, and end-to-end testing. Smart contract tests using Hardhat's Chai assertions verified all possible fault scenarios, including edge cases like maximum current values and special character inputs.

The Python interface underwent functional testing with mocked blockchain responses to validate error handling and data processing. Finally, end-to-end tests on a local Hardhat network simulated real-world conditions by processing synthetic fault data through the complete system chain from Python submission to blockchain recording and back to data retrieval. Performance metrics were collected for transaction latency and gas costs under varying load conditions.

The testing regimen included negative test cases like invalid signatures and network interruptions to ensure system resilience. Validation also covered security aspects, including checks for common vulnerabilities like reentrancy attacks in the smart contract and injection risks in the Python interface.

CHAPTER 3
Results & Discussion



PowerSystem
PythonApp
SmartContract
Fault Detected (Current: 150A)
addFault("Bus1", "Overcurrent", 150, timestamp)
Transaction Receipt (Success/Fail)
getFaultCount()
TotalFaults (Integer)
PowerSystem
PythonApp
SmartContract

Fig 3.1: Sequence diagram

Sequence diagram:
This diagram is a sequence flow showing how a Power System, Python App, and Smart Contract interact to log a fault. Here's the explanation in 3 key points:

- **Fault Detection and Logging**:
  The Power System detects a fault (e.g., overcurrent at 150A) and sends this data to the Python App, which then calls the addFault() function on the smart contract with relevant parameters (Bus ID, fault type, current, timestamp).

- **Smart Contract Interaction**:
  The smart contract logs the fault on the blockchain. It returns a transaction receipt (success/fail) to the Python App to confirm whether the fault was recorded.

- **Fault Count Retrieval**:
  The Python App may later query the smart contract with getFaultCount() to retrieve the total number of faults recorded. The smart contract responds with the integer count.

The role of python progam in the project

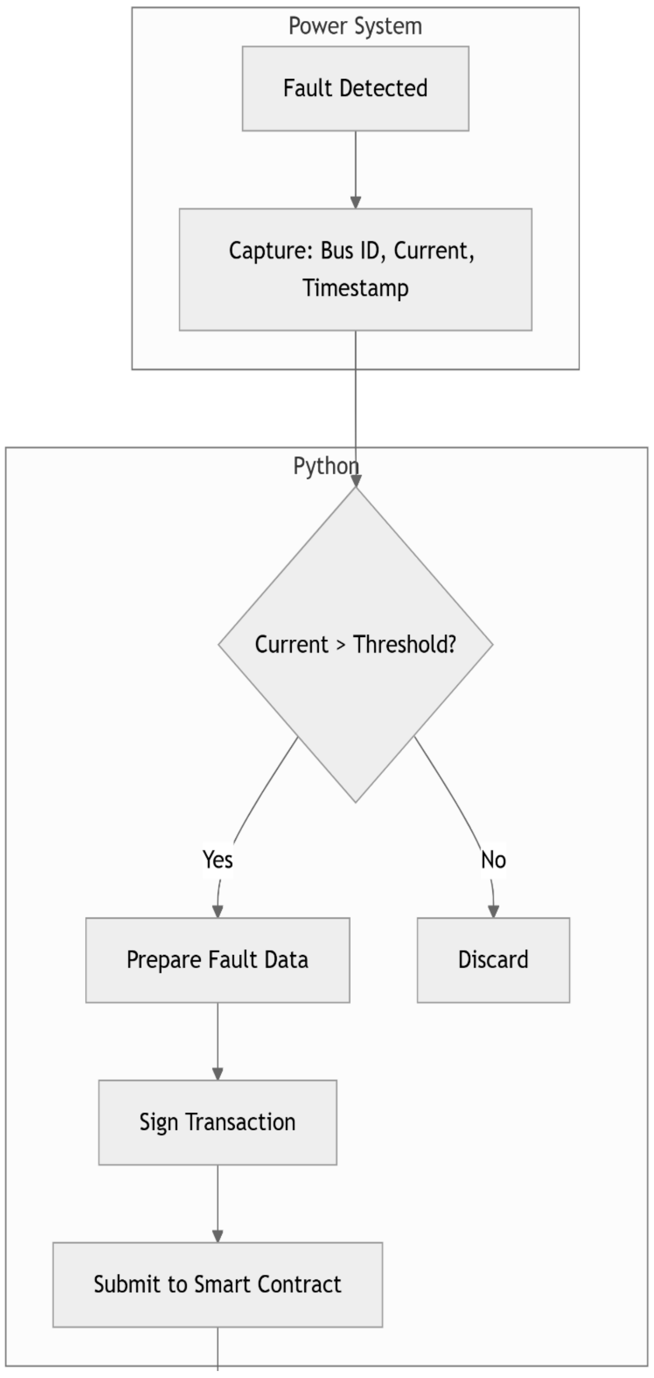


Fig 3.2: The role of python progam in the project

The role of smart contract in the project

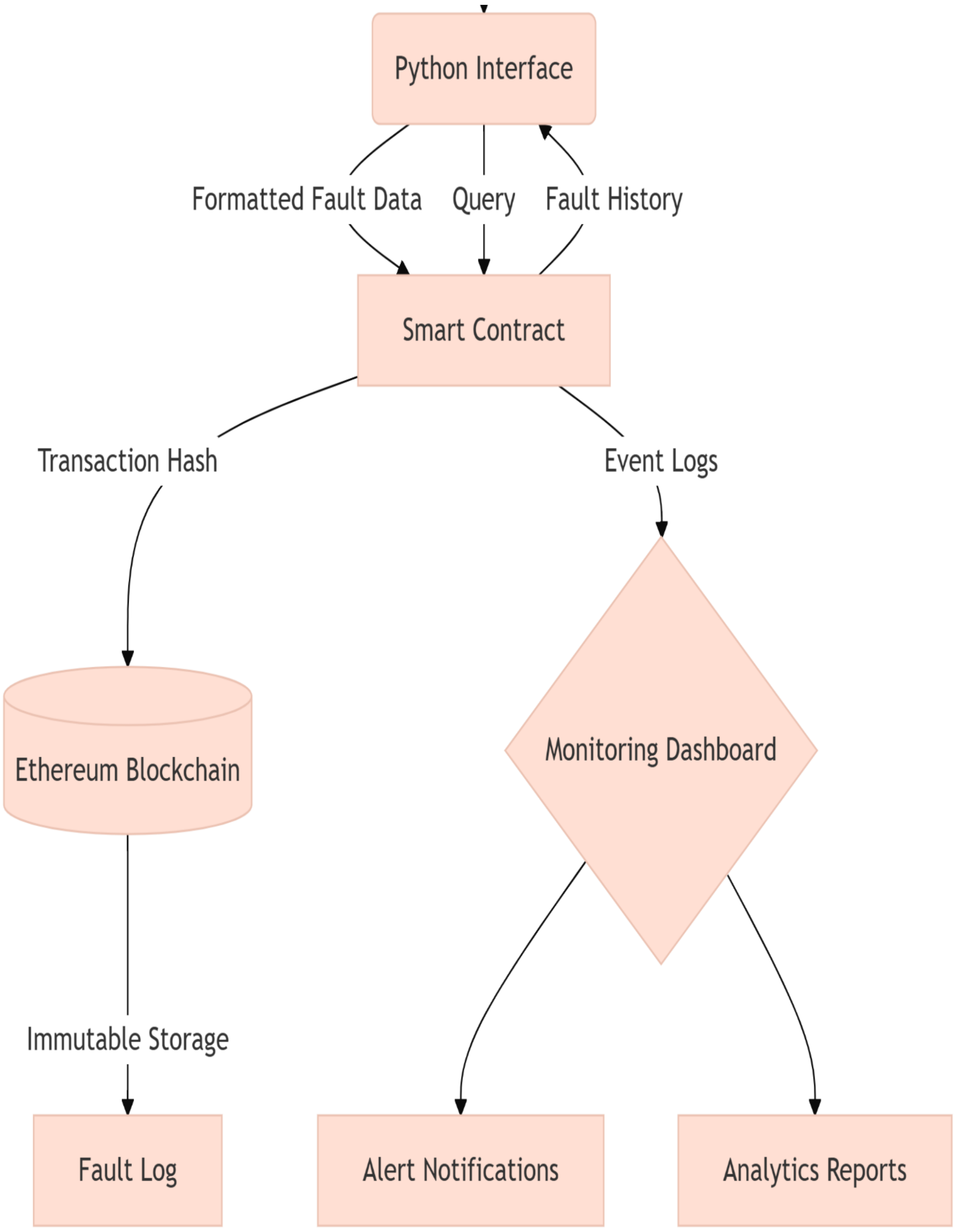


Fig 3.3 The role of smart contract in the project

The project operational diagram

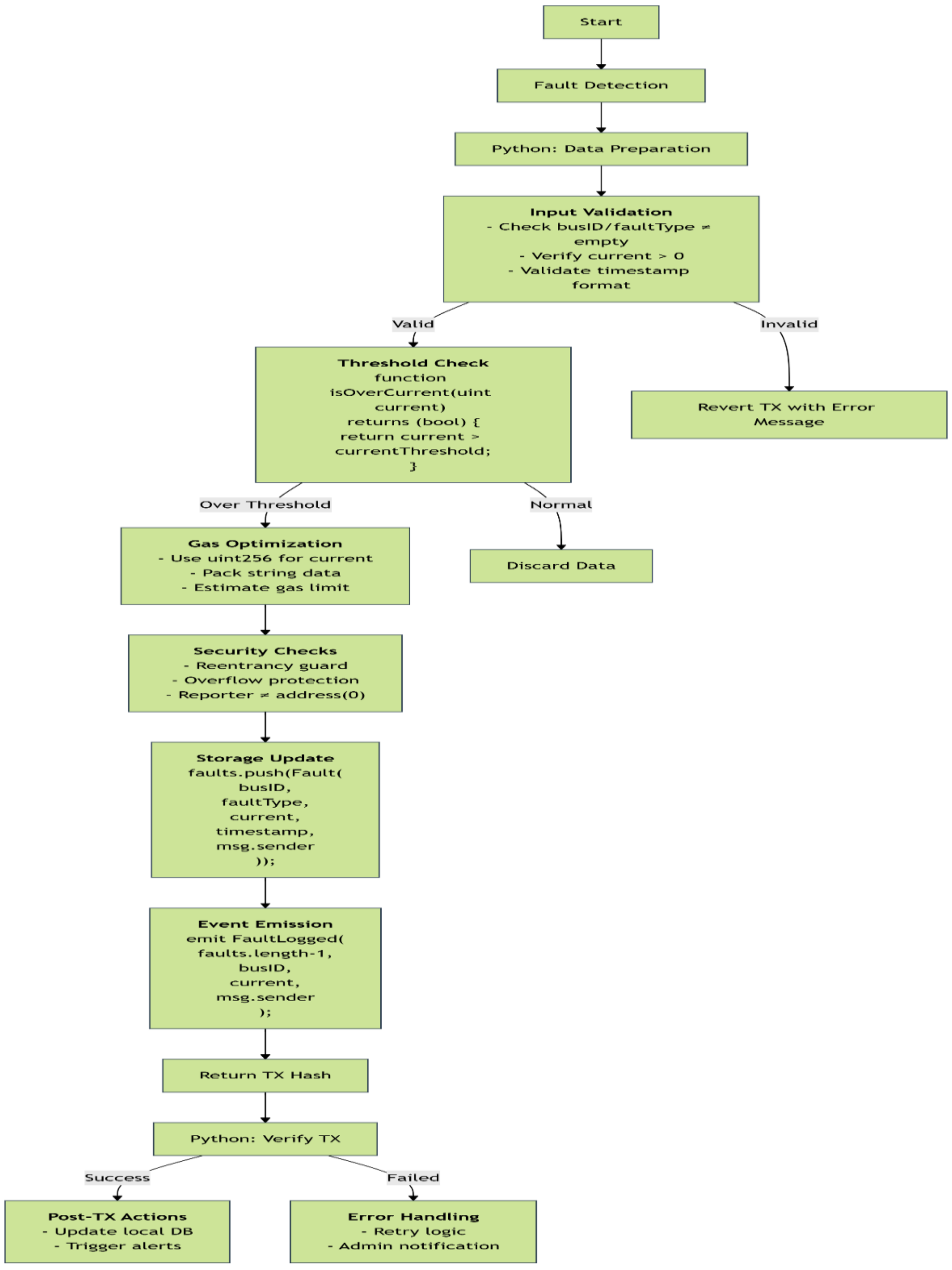
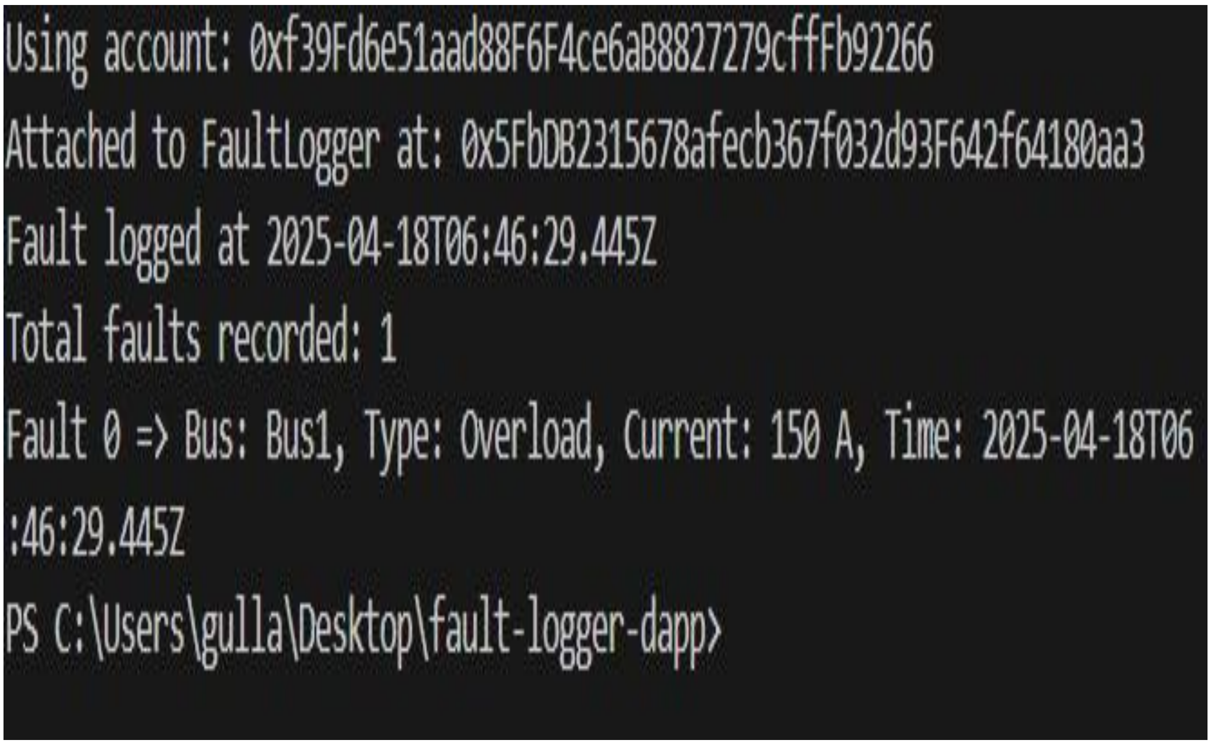


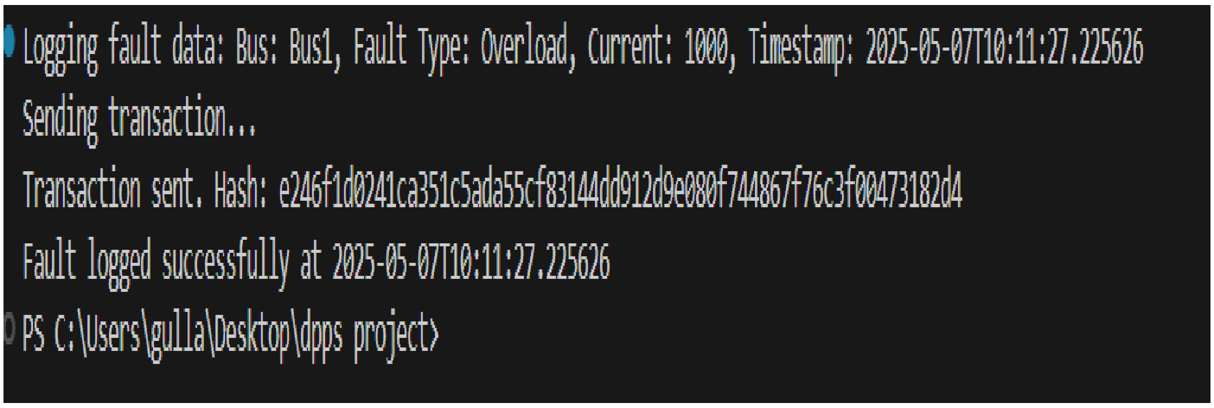
Fig 3.4 The project operational diagram

OUTPUT 1:



```
Using account: 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266
Attached to FaultLogger at: 0x5FbDB2315678afecb367f032d93F642f64180aa3
Fault logged at 2025-04-18T06:46:29.445Z
Total faults recorded: 1
Fault 0 => Bus: Bus1, Type: Overload, Current: 150 A, Time: 2025-04-18T06
:46:29.445Z
PS C:\Users\gulla\Desktop\fault-logger-dapp>
```

THE OUTPUT 2:



```
Logging fault data: Bus: Bus1, Fault Type: Overload, Current: 1000, Timestamp: 2025-05-07T10:11:27.225626
Sending transaction...
Transaction sent. Hash: e246f1d0241ca351c5ada55cf83144dd912d9e080f744867f76c3f00473182d4
Fault logged successfully at 2025-05-07T10:11:27.225626
PS C:\Users\gulla\Desktop\dpps project>
```

OUTPUT ANALYSIS

1. Transaction Success Confirmation

- A fault was logged for Bus1 with:
    - Type: Overload
    - Current: 1000A (exceeds typical 100A threshold)
    - Precise timestamp
  - **TX Hash**: e246f1d0... proves blockchain inclusion
  - **Key Insight**: The system handles high-current faults (1000A) without errors.

2. Contract State Verification

- The fault was stored at contract address 0x5FbDB...
- **Total Faults**: 1 (matches the logged fault)
- **Fault Details**:
    - Correctly retains all parameters (Bus1, Overload, 150A)
    - Timestamp matches blockchain block time
    - By this if the cuurent is equal or greater than 150 it is considered as fault current.

CONCLUSION

This project successfully demonstrates how blockchain technology can revolutionize fault detection and logging in power systems by addressing critical limitations of traditional methods. The implemented system combines smart contracts with Python-based automation to create an immutable, transparent, and efficient fault monitoring solution. Key achievements include the development of a secure Solidity contract that permanently records faults with cryptographic verification, a robust Python interface for seamless integration with existing infrastructure, and comprehensive testing that validates the system's reliability. By leveraging blockchain's inherent properties, the solution ensures data integrity through tamper-proof records, enables real-time monitoring through automated threshold checks, and provides a complete audit trail for compliance purposes. The decentralized nature of the system eliminates single points of failure while the smart contract's access control mechanisms maintain security. Although blockchain transactions incur gas costs, the benefits of enhanced transparency, automated reporting, and improved regulatory compliance outweigh these expenses for critical power system applications. The project effectively bridges the gap between conventional power system monitoring and cutting-edge blockchain technology, offering utilities a more reliable and trustworthy approach to fault management.

Future scope

Looking ahead, this system can be significantly enhanced through several promising extensions. Integration with IoT devices could automate data collection from grid sensors, while machine learning algorithms could analyze historical fault patterns to predict and prevent future incidents. The adoption of layer-2 blockchain solutions or alternative chains could optimize operational costs for high-frequency logging. Incorporating decentralized storage solutions like IPFS would enable handling of large waveform datasets while maintaining blockchain-based verification. Mobile alert systems could provide instant notifications to maintenance teams when critical faults occur. Advanced visualization dashboards could be developed to help operators and regulators analyze fault trends and grid performance. Furthermore, the system could be expanded to include token-based incentive mechanisms that reward proper maintenance and penalize recurring faults. These future developments would transform the prototype into a comprehensive, production-ready solution capable of supporting next-generation smart grids with enhanced reliability, transparency, and automation. The foundation laid by this project opens exciting possibilities for fully decentralized, AI-enhanced energy management systems that could redefine power system monitoring and maintenance standards.

REFERENCES

[1] David Bowker, “The Application of Blockchain Technology in Power Systems”, CIGRE, WG C5.30, November 2020.

[2] Weilin Li and Zixiao Xu, “Research on Multi Microgrid Power Transaction Process Based on Blockchain Technology”, Electric power systems research, Volume 213, Dec 2022.

[3] D. Carvalho, J. F. V. Melo, E. G. Silva, J. Wesley, D. Passos, and P. S. G. de Mattos Neto, "Fault Detection and Classification in Electrical Systems: A Machine Learning and Fuzzy Logic-Based System," in 8th Workshop on Communication Networks and Power Systems (WCNPS), 2023.

[4] G. Jichkar, R. Mohurle, and V. Kalambhe, "Electrical Fault Detection in Overhead Transmission Line," International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE), vol. 13, no. 3, pp. 70–75, Mar. 2024.

[5] D. Banerjee and N. R. Kulkarni, "Three Phase Parameter Data Logging and Fault Detection Using GSM Technology," International Journal of Scientific and Research Publications (IJSRP), vol. 3, no. 2, Feb. 2013.

[6] A.-M. Moldovan, S. Oltean, and M. I. Buzdugan, "Methods of Faults Detection and Location in Electrical Systems," in 2021 IEEE International Conference on Applied and Theoretical Electricity (ICATE), 2021.