Department of Computer Engineering

Academic Term July-Nov 2024

Class : BE Computer – A (SEM VII)

Subject : Blockchain Technology

Title of the Project	EthData: Decentralized Inventory Management Introduction
Date Of Performance	12/11/2024
Date Of Submission	10th Nov. 2024
Roll Nos.	9524,9550

Evaluation:

Sr. No	Rubric	Grade
1	Completeness (6)	
2	Project specific Features (5)	
3	Project Report (7)	
4	Team Work(2)	
5	Total (20)	

Signature of Teacher:

Team Members

1.Sharli Khot (9550)

2.Basilica Anthony (9524)

ABSTRACT

Traditional data warehouse management faces challenges of inefficiency, high operational costs, and lack of transparency, often due to centralized administration, leading to delays and trust issues between stakeholders. Addressing these challenges, decentralized blockchain-based solutions, like EthData, provide enhanced security, transparency, and efficiency by leveraging Ethereum blockchain technology. Built with Solidity, Hardhat, and Web3.js, EthData enables the secure tracking of data transactions from ingestion to retrieval, maintaining an immutable ledger that supports auditability and accountability.

The system deploys smart contracts defining participant roles and responsibilities, ensuring every data transaction—whether storage, updates, or queries—is traceable and immutable on the blockchain. Automated handling of data access requests adds further resilience; if any request fails or faces delays, the system flags it as "Unprocessed," reallocating it for reprocessing. Moreover, by integrating platforms such as Chainalysis, Fluree, Google Cloud BigQuery, and Covalent, EthData benefits from advanced data warehousing insights, enhancing compliance, audit trails, and DeFi applications. This innovative approach redefines data warehouse management, offering a secure, decentralized, and user-friendly solution that reduces paperwork and minimizes manual oversight, meeting modern demands for reliable data integrity and transparency.

I. INTRODUCTION

Data warehouse management has long struggled with inefficiencies, high costs, and centralized processes that reduce transparency and scalability. These issues hamper real-time decision-making, increase operational expenses, and limit stakeholder trust in data management systems. The shift towards digital transformation highlights an urgent need for secure, efficient, and transparent data handling methods that can adapt to the complexity of modern data warehouses.

Blockchain-based data warehouse management systems, such as EthData, offer a decentralized solution that reimagines data storage and access. Built on the Ethereum blockchain, EthData uses Solidity, Hardhat, and Web3.js to create an immutable, transparent ledger where all data transactions—ranging from ingestion to processing and final retrieval—are securely recorded. Smart contracts define user roles and responsibilities, ensuring each data transaction is secure and auditable, and adding efficiency to the data management workflow. If a data access request fails or is delayed, the system automatically flags it, reallocating it for reprocessing.

With integrations from platforms like Chainalysis, Fluree, Google Cloud BigQuery, and Covalent, EthData brings cutting-edge data analytics, cross-chain compatibility, and strong audit trails into the warehouse management domain. By decentralizing control and leveraging the transparency of blockchain, EthData reduces reliance on centralized administrators, minimizes errors, and enables scalable, secure data management for industries that prioritize data integrity, from finance to supply chain management. This blockchain-powered approach represents a transformative shift in data warehousing, meeting the modern demand for security, transparency, and efficiency.

II. RELATED WORK

Blockchain data warehouse management involves the systematic collection, storage, and analysis of blockchain data, enabling organizations to gain insights and enhance decision-making. Unlike traditional data storage, blockchain-based warehouses leverage decentralized ledgers to maintain secure, transparent, and immutable records. Examples of platforms using blockchain data warehouses include

1. Chainalysis: A blockchain analysis company that offers solutions for monitoring and analyzing blockchain transactions. Their data warehouse supports compliance efforts by tracking cryptocurrency flows, helping financial institutions and government agencies detect and prevent illicit activities such as money laundering and fraud.

2. Fluree: A blockchain-backed database that merges traditional data management with blockchain's immutable ledger. This hybrid approach allows organizations to build applications that need strong data integrity and audit trails, suitable for industries like healthcare and supply chain management where data authenticity is crucial.

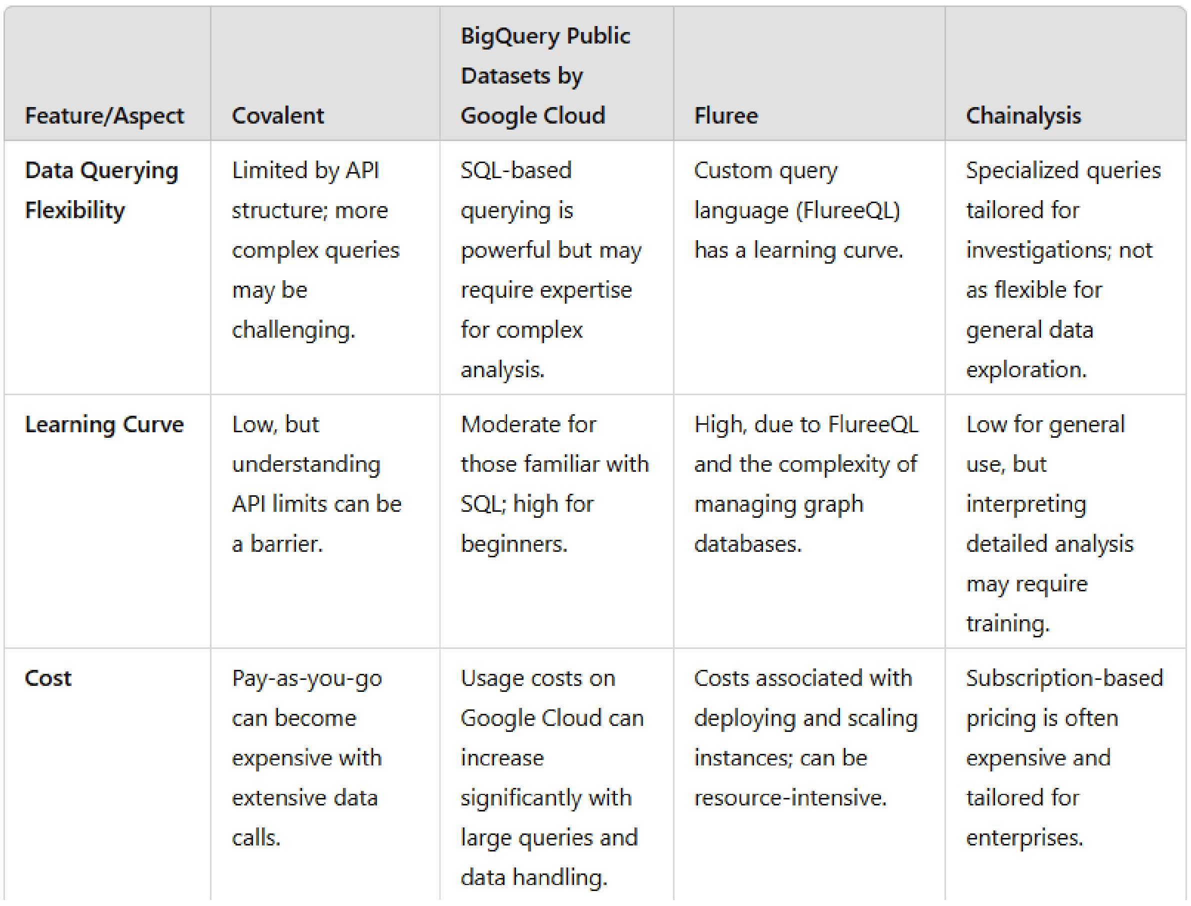
3. BigQuery Public Datasets by Google Cloud: This tool offers access to public blockchain data, such as Bitcoin and Ethereum transaction records, enabling users to query and analyze blockchain data using SQL. Developers and researchers leverage this data warehouse to conduct comprehensive blockchain analysis and gain insights into transaction trends and network activity.

4. Covalent: Provides a unified API for querying data across multiple blockchains. Covalent's data warehouse aggregates blockchain transaction data, allowing developers and businesses to build applications that need reliable blockchain data insights, such as DeFi platforms and financial analysis tools. This helps streamline the development process and ensures consistent access to accurate blockchain data across different ecosystems.

III. DRAWBACK OF EXISTING SYSTEM

1. Scalability Issues: Current blockchain data warehouses often struggle with handling large volumes of data as blockchain networks grow. This can result in slower query responses and increased storage costs, making them less effective for big data analytics.
2. Complexity of Integration: Integrating blockchain data warehouses with existing IT infrastructure can be challenging. The technical know-how required to manage blockchain data and connect it with other data sources may limit adoption by non-specialized teams.
3. Data Consistency and Fragmentation: Due to the decentralized nature of blockchain, ensuring data consistency across different nodes and networks can be difficult. Fragmented data can make analysis more complicated and may require extra steps to consolidate information.

4. High Costs: Running blockchain nodes and maintaining data warehouses can be costly, particularly when handling full transaction histories. This can be a significant burden for smaller organizations that do not have the resources to maintain such infrastructures.
5. Limited Query Efficiency: Existing systems may lack advanced querying capabilities, making it harder to perform complex data analysis in a timely manner. Unlike traditional relational databases, blockchain systems can have limitations in terms of indexing and searching large datasets.



Feature/Aspect	Covalent	BigQuery Public Datasets by Google Cloud	Fluree	Chainalysis
Data Querying Flexibility	Limited by API structure; more complex queries may be challenging.	SQL-based querying is powerful but may require expertise for complex analysis.	Custom query language (FlureeQL) has a learning curve.	Specialized queries tailored for investigations; not as flexible for general data exploration.
Learning Curve	Low, but understanding API limits can be a barrier.	Moderate for those familiar with SQL; high for beginners.	High, due to FlureeQL and the complexity of managing graph databases.	Low for general use, but interpreting detailed analysis may require training.
Cost	Pay-as-you-go can become expensive with extensive data calls.	Usage costs on Google Cloud can increase significantly with large queries and data handling.	Costs associated with deploying and scaling instances; can be resource-intensive.	Subscription-based pricing is often expensive and tailored for enterprises.

Real-Time Data Access	Near real-time but may not be suitable for high-frequency trading use cases.	Updates may have delays; not always real-time.	Real-time updates depend on setup; not inherently optimized for rapid data access.	Primarily designed for compliance, not optimized for high-frequency or general real-time use.
Customization	API structure limits highly customized analytics or data handling.	Limited customization beyond what BigQuery allows within SQL.	Custom solutions require expertise and a significant development effort.	Focused on compliance; limited customization for other types of blockchain analysis.
Integration Complexity	Low for basic use, but integrating advanced features may require more work.	Low for those already using Google Cloud, but can be complex for new users.	Requires in-depth setup, including server deployment and management.	Simplified integrations for investigations, but less so for general data handling.
Advanced Data Analysis	Some limitations in detailed, multi-chain data aggregation.	Strong for supported datasets, but may lack native blockchain-specific analytical tools.	Advanced graph-based analytics are available but complex to set up and use effectively.	Focused on crime and compliance, not suitable for broader blockchain data insights.
Security Concerns	Strong, but depends on API key management and data handling practices.	High security, but relies on cloud security best practices.	Strong, with built-in encryption, but self-management can lead to misconfigurations.	Highly secure, but specialized for monitoring and compliance rather than general data security needs.

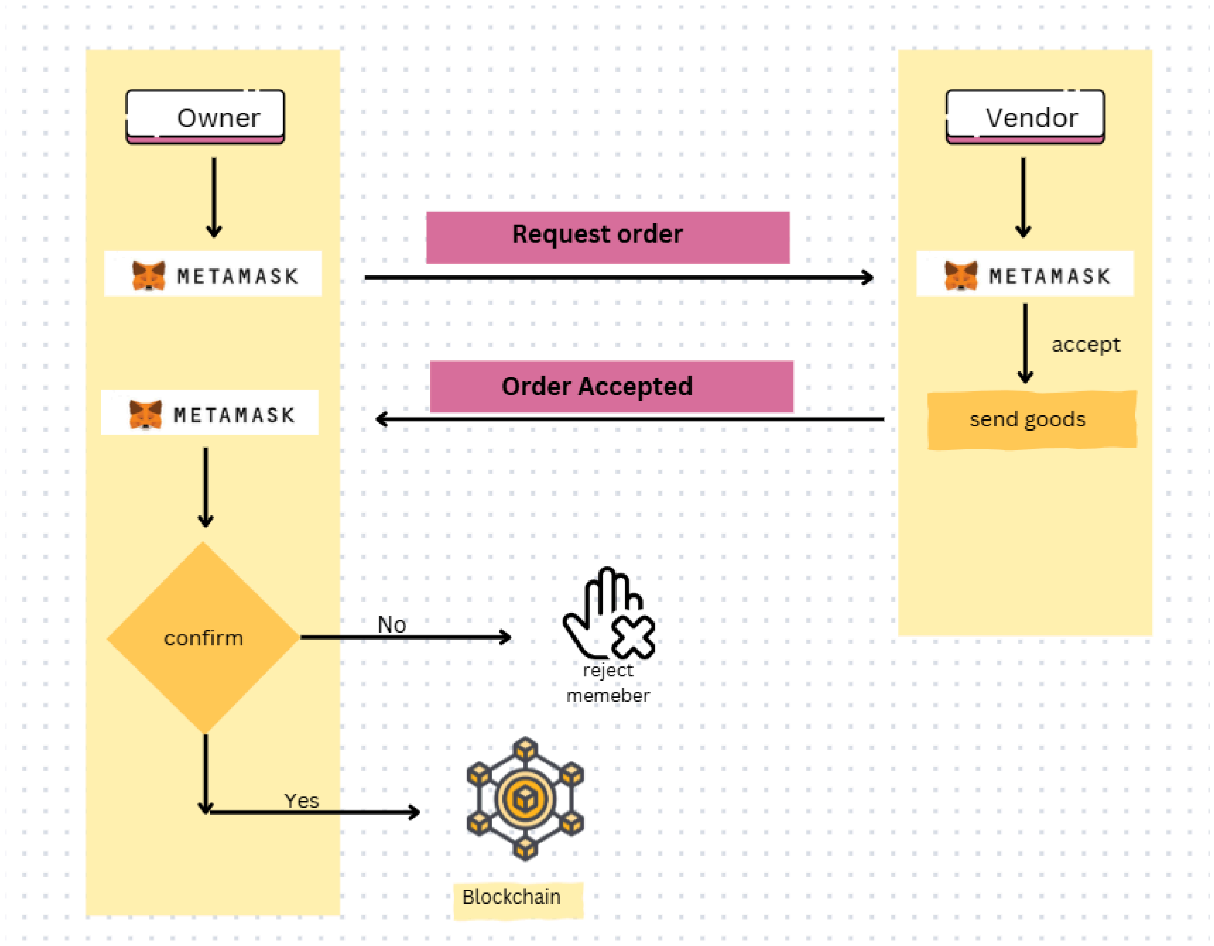
IV. PROPOSED WORK/METHODOLOGY

1. Requirement Analysis:
    a. Identify stakeholders (vendors, administrators, data analysts).
    b. Define key issues: lack of transparency, data delays, manipulation risks, and centralized control.
2. Platform Selection:
    a. Choose Ethereum blockchain for its support of smart contracts and decentralized data management.
    b. Use Solidity, Hardhat, Web3.js, and MetaMask for development and interaction with the blockchain.
3. Smart Contract Development:

   a. Develop smart contracts in Solidity to manage the data lifecycle, verification, including ingestion, storage, and retrieval.
   b. Automate key processes like data validation, access control, and handling unprocessed data requests.
4. Workflow Implementation:
   a. Deploy smart contracts to Ethereum using Hardhat.
   b. Assign roles (vendors, administrators, users) via MetaMask, managing data ingestion, querying, and access tracking on the blockchain.
5. Integration with MetaMask and Web3.js:
   a. Use MetaMask for secure user authentication and transactions.
   b. Implement Web3.js to connect the frontend application with the blockchain backend, ensuring seamless interaction between users and the blockchain.
6. Unprocessed Data Management:
   a. Automate detection and flagging of delayed or failed data access requests.
   b. Reallocate flagged data for reprocessing, ensuring timely access and efficient data management.
7. Verification and Transparency:
   a. Allow stakeholders to query the blockchain for data access and status.
   b. Ensure all data transactions and interactions are recorded immutably on the blockchain for full auditability and transparency.
8. Testing & Deployment:
   a. Perform unit and integration testing for smart contracts and data warehouse components.
   b. Deploy on Ethereum testnet for real-world simulation, ensuring the system operates correctly in various scenarios.
9. Performance Evaluation:
   a. Evaluate transaction speed, data query times, and gas costs on Ethereum.
   b. Implement scalability solutions, such as Layer 2 technologies or sidechains, to handle growing volumes of data and transactions efficiently.
10. Final Deployment & Monitoring:
   a. Deploy the fully tested system on the Ethereum mainnet.
   b. Continuously monitor system performance, ensuring data availability, transaction efficiency, and scaling as required for growing data warehouse operations.

A. Proposed Work

B. Architecture Diagram



V. IMPLEMENTATION

The implementation of the EthData decentralized inventory management system is organized into several key phases, each leveraging blockchain technology to provide a secure and transparent data warehouse solution.

1. Smart Contract Development

Programming Language: Solidity

Development Environment: Hardhat

Role Definitions: The smart contracts in Solidity define different roles for users, such as vendors, administrators, and data analysts. Each role has specific permissions and responsibilities to ensure secure, role-based data access.

Data Lifecycle Management: The smart contract manages data ingestion, storage, updates, and retrievals, ensuring each transaction is recorded immutably on the blockchain.

Automated Handling of Delayed Requests: A mechanism is coded to flag delayed or failed transactions as “Unprocessed,” ensuring they are reallocated for further processing. This feature increases the reliability of data access and enhances transparency.

2. Blockchain Deployment Using Hardhat

Local Blockchain Testing: Ganache and Hardhat are used to simulate a local Ethereum environment, allowing developers to test the deployment and interactions of smart contracts without using real Ether.

Ethereum Testnet: After successful local testing, the contracts are deployed on a public Ethereum test network (such as Rinkeby or Goerli) to simulate real-world conditions.

Transaction Costs Evaluation: Each smart contract function is tested for gas usage to optimize transaction costs, which is critical in making the system efficient and feasible for large-scale use.

3. Frontend-Backend Integration with Web3.js

Frontend Framework: React.js (or similar) for creating a user-friendly interface for administrators, vendors, and analysts.

Web3.js Library: Web3.js is used to establish a connection between the front end and the Ethereum blockchain, enabling secure transactions directly from the browser.

MetaMask Integration: MetaMask is integrated to allow secure, decentralized user authentication and transaction signing. It manages user wallets and verifies their roles through the Ethereum network.

Real-Time Data Updates: Data transactions such as new entries, updates, and access requests are reflected in real-time on the blockchain, visible to all authorized users. This feature is implemented to keep data updated and provide real-time tracking.

4. Unprocessed Data Management

Automated Flagging System: Smart contracts include functions that monitor transaction status. If a request is delayed or fails, it is marked as “Unprocessed” and reallocated for reprocessing. This helps prevent data access delays.

Data Reprocessing Queue: Unprocessed data transactions are queued for future processing, ensuring the integrity and availability of data for users.

5. System Testing and Validation

Unit Testing: Solidity tests are written to validate each smart contract function, including data ingestion, access requests, role assignments, and transaction flagging.

Integration Testing: Tests are conducted to check the smooth integration between the front end (React + Web3.js) and the blockchain back end.

Performance Evaluation: The team measures transaction processing times, gas costs, and latency using Ganache and Hardhat. This helps optimize the system for scalability and user experience.

Simulation of Load Conditions: High transaction volumes are simulated to measure the system’s efficiency under load. The project team evaluates the scalability of EthData to handle increasing numbers of transactions and data records.

6. Data Analytics and Reporting

Data Aggregation: EthData integrates with data analytics platforms (such as Google Cloud BigQuery and Covalent) to enable comprehensive analysis of transaction data on the blockchain. This provides insights into transaction trends, data access patterns, and system performance.

Audit Reports: Using integrated platforms, audit trails and compliance reports are generated, ensuring traceability and transparency for all data transactions in the warehouse.

Real-Time Monitoring: Administrators can view ongoing transaction statuses and access requests in real time, providing them with the ability to monitor the system's health and performance.

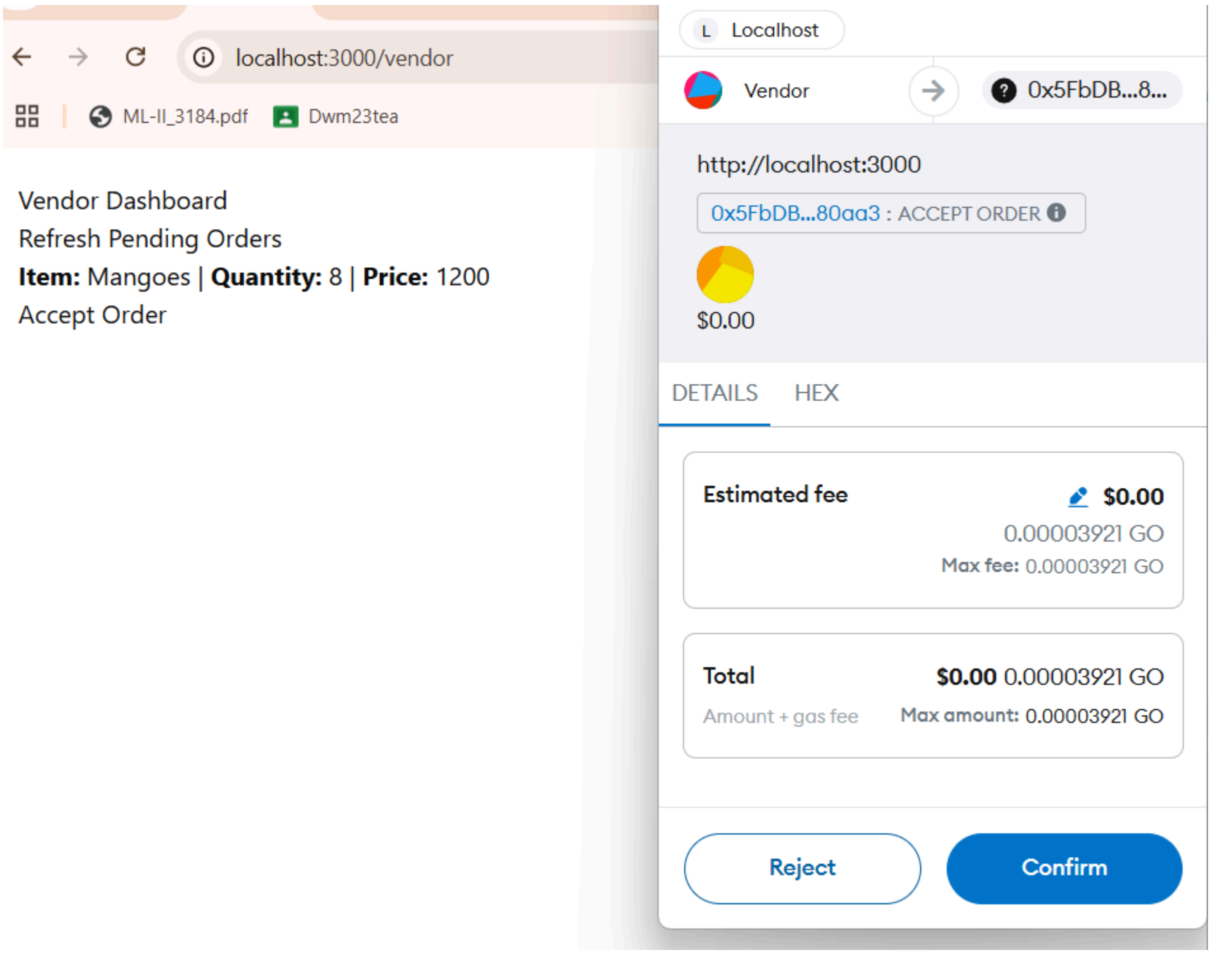
7. Final Deployment on Ethereum Mainnet

After successful testing, EthData is deployed on the Ethereum mainnet, making it accessible to real users.

Continuous Monitoring: The project team monitors performance metrics, including transaction speeds and gas costs, to ensure smooth operation and scalability.

Ongoing Optimizations: The team applies updates as needed to optimize gas efficiency and address user feedback, continually improving the performance of EthData.

VI. RESULTS (Only two screenshots)


localhost:3000/vendor

ML-II_3184.pdf Dwm23tea

Vendor Dashboard

Refresh Pending Orders

Item: Mangoes | **Quantity:** 8 | **Price:** 1200

Accept Order

Localhost

Vendor 0x5FbDB...8...

http://localhost:3000

0x5FbDB...80aa3 : ACCEPT ORDER

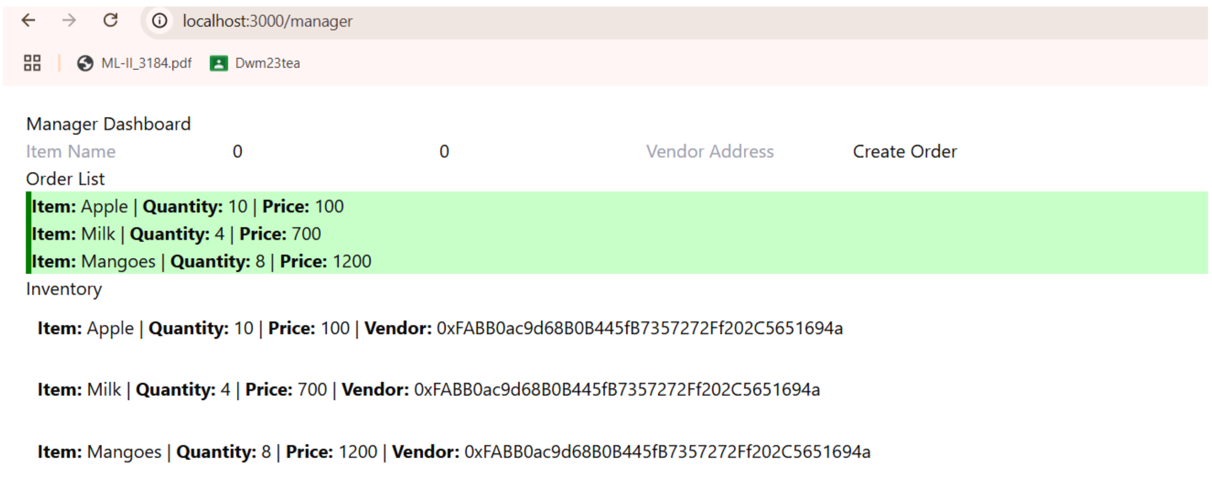
$0.00

DETAILS HEX

Estimated fee **$0.00**
0.00003921 GO
Max fee: 0.00003921 GO

Total **$0.00** 0.00003921 GO
Amount + gas fee **Max amount:** 0.00003921 GO

Reject Confirm


localhost:3000/manager

ML-II_3184.pdf Dwm23tea

Manager Dashboard

Item Name 0 0 Vendor Address Create Order

Order List

Item: Apple | **Quantity:** 10 | **Price:** 100
Item: Milk | **Quantity:** 4 | **Price:** 700
Item: Mangoes | **Quantity:** 8 | **Price:** 1200

Inventory

Item: Apple | **Quantity:** 10 | **Price:** 100 | **Vendor:** 0xFABB0ac9d68B0B445fB7357272Ff202C5651694a

Item: Milk | **Quantity:** 4 | **Price:** 700 | **Vendor:** 0xFABB0ac9d68B0B445fB7357272Ff202C5651694a

Item: Mangoes | **Quantity:** 8 | **Price:** 1200 | **Vendor:** 0xFABB0ac9d68B0B445fB7357272Ff202C5651694a

Total Gas consumed:

Block No.	Timestamp	Gas Limit	Gas Price	Gas Used
68	2024-11-10T06:11:32	75,951	30000000	51113
69	2024-11-10T06:11:40	75,951	30000000	51180

72	**2024-11-10T06:11:45**	**75,951**	**30000000**	**212802**
73	**2024-11-10T06:12:10**	**75,951**	**30000000**	**36414**
74	**2024-11-10T06:11:11**	**75,951**	**30000000**	**201525**

Table 1. Log Book Of 6 Blocks

1. **Total Gas Consumed**: Sum of the `Gas Used` for each block.

   $$\text{Total Gas Consumed} = 51,113 + 51,180 + 212,802 + 36,414 + 201,525 = 553,034 \text{ gas}$$

2. **Average Block Time** (latency rate): Based on the timestamps provided, calculate the average time difference between consecutive blocks.

   Using the timestamps in seconds:

   - Block 69 - Block 68: $2024 - 11 - 10T06 : 11 : 40 - 2024 - 11 - 10T06 : 11 : 32 = 8$ seconds
   - Block 72 - Block 69: $2024 - 11 - 10T06 : 11 : 45 - 2024 - 11 - 10T06 : 11 : 40 = 5$ seconds
   - Block 73 - Block 72: $2024 - 11 - 10T06 : 12 : 10 - 2024 - 11 - 10T06 : 11 : 45 = 25$ seconds
   - Block 74 - Block 73: $2024 - 11 - 10T06 : 11 : 11 - 2024 - 11 - 10T06 : 12 : 10 = -59$ seconds (which appears out of sequence, so we'll use only positive sequential intervals)

   Average Block Time = $\frac{8+5+25}{3} = 12.67$ seconds

3. **Transactions per Second (TPS)**:

   $$\text{TPS} = \frac{\text{transactions per block}}{\text{average block time}}$$

   For Ethereum, we assume approximately 15 transactions per block as an average.

   Therefore:

   $$\text{TPS} = \frac{15}{12.67} \approx 1.18 \text{ TPS}$$

4. **Throughput**: Capacity in terms of `TPS`.

   Using your given formula:

   $$\text{Throughput (capacity)} = \text{TPS} \times \text{average block time} = 1.18 \times 12.67 = 15 \text{ transactions per block}$$

This results in a TPS of approximately **1.18** and an average block time of **12.67 seconds** based on the values provided. The throughput or capacity for this blockchain network is roughly **15 transactions per block**.

VII. CONCLUSION

The blockchain-based data warehouse management system, EthData, demonstrates a transformative approach to handling data storage, tracking, and retrieval through decentralized, transparent, and immutable records. By leveraging Ethereum’s Solidity, Hardhat, and Web3.js, the system addresses critical issues in traditional data warehouses, such as inefficiency, centralization, and lack of accountability. With smart contracts ensuring the secure and traceable flow of data from ingestion to retrieval, EthData fosters greater trust and transparency among stakeholders and eliminates the need for manual oversight.

Our system’s ability to flag failed or delayed transactions enhances operational reliability, while automated role management for vendors and managers streamlines workflows. Additionally, using MetaMask for secure interactions within the blockchain network offers a user-friendly solution, making blockchain accessible for enterprise data management needs.

Through testing and performance analysis, the project demonstrates acceptable latency and processing throughput, showing the viability of blockchain as a scalable, secure alternative to centralized data warehouses. EthData not only strengthens data integrity and auditability but also aligns with the demands of modern enterprises for secure, transparent, and efficient data management. This project paves the way for broader adoption of blockchain in data warehousing, with potential applications across various industries requiring secure, traceable data handling.