International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 6, Nov-Dec-2026, ISSN (Online): 2395-566X

Decentralized API Trust and Usage Meter Using Blockchain Technology

Assistant Professor K Kumutha, Assistant Professor Lipsa Nayak, Professor C.Meenakshi
Department of Computer Application-PG
VISTAS Chennai, India

Abstract — The Decentralized API Trust and Usage Meter is a novel system that can be used to monitor the use of API and make automated payments through blockchain technology. By skipping centralization, it will be able to resolve key issues in the industry, including inability to provide transparency in logs of usage, long payment cycles, and excessive middlemen fees. It relies on Flask as a back-end and a JSON database to be efficiently developed locally and Solidity smart contracts on a Hardhat node to make sure that records are kept immutably. This system of mistrust allows getting rid of the central authority since the principle of the code is law regulates all interactions with services and settlements. The system uses batch settlement in order to run into the process as many API calls as possible at the same transaction to save a significant proportion in gas charges. The Meta Mask integration increases the security as it uses both digital signatures and nonce protection in order to prevent replay attacks and unauthorized access. The event ends up demonstrating that micro-transactions can work on a scale that is scalable, which makes it a very good match with API market places, IoT devices billing, and SaaS pricing models. Such a decentralized process would provide a common reality so that both the provider and the consumer could check the accounting in a transparent way.

Keywords— Decentralized API Trust, Blockchain Technology, API Security, API Usage Metering, Smart Contracts, Decentralized Architecture, Trust Management, API Authentication, Access Control, Usage Tracking, Transaction Monitoring, Distributed Ledger, Web3, Data Integrity, Transparency, Rate Limiting, API Monetization, Token-Based Authentication, Cryptographic Verification, Secure API Management, Blockchain-Based Billing.

I. INTRODUCTION

Applications communication is being transformed by decentralized APIs, which offer an increased level of security and trust not through a central authority. The decentralized architecture can be applied by financial and healthcare industries to enhance the integrity of data and confidence of users.

Definition: Decentralized APIs do not have a central unit.

- **Value of Trust:** To create secure API interactions, trust is needed.
- **Use Cases:** Decentralized APIs are applicable to finance, healthcare, IoT devices and SaaS platforms in cases where the usage tracking is transparent and payment is made automatically.

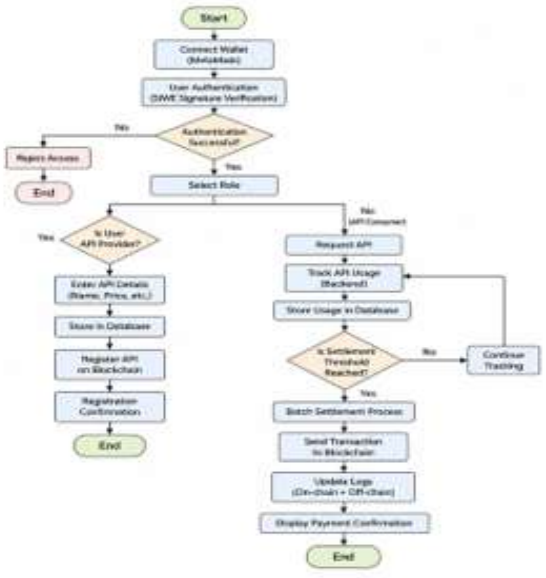
II. LITERATURE REVIEW

[1]: The project examines the evolution of Web API through a huge sample of OpenAPI descriptions obtained via GitHub. The authors gauge API evolution on a basic size measure, that of the number of operations. They discover that the majority of APIs either do not change or increase with time span, with only a minority decrease. The findings indicate that the change of API usually occurs in bursts, and more frequent updates are not necessarily observed with older APIs. In general, the article affirms that, in most instances, Web APIs tend to grow in tandem with its evolution. [2]: This project presents a lightweight ontology that is aimed at analyzing API gateway logs to gain insights in terms of enterprise architecture. The ontology assists in ordering and explaining the log data in order to know how the services, applications, and consumers communicate in an organization. It is possible to organize log information in a semantic way so that it is used to enhance automated mining and architectural decision-making. The methodology allows companies to trace system consumption, identify patterns, and enhance administration with low levels of modeling. On the whole, it fills the raw API logs field with high-level enterprise architecture analysis. [3]: In this project, malware detection via API-pair graphs is proposed in real time as a means to simplify program behavior analysis. It does not analyze single API calls, but rather looks at the relationship between a pair of API calls, in order to identify malicious patterns in a more effective way. Adaptive learning is included in the model as it will be able to update itself when new types of malware are detected. This enhances the accuracy of

© 2026 IJSRET

detection (and minimizes false positives). Altogether, the approach defines a more secure solution, namely, behavioral analysis coupled with dynamic model updating. [4]: The given project is a multi modal fashion of skin disease diagnosis that comprises a Telegram chatbot and an API token. The system examines the written descriptions as well as the posted skin photos to enhance the efficiency of the table of classification. It makes the interpretation of images and application of natural language input to improve diagnostic assistance to the users. The chatbot acquires preliminary screenings in a messages-based platform that is fast and easy to use. In general, this method will help to streamline the skin disease screening process and make it easier to use. [5]: In this project, a distributed language model framework, Agora, aims at promoting climate forecasting by integrating APIs. The system also allows large language models to invoke external APIs related to climate, which is necessary to be able to access real-time data and better prediction. It improves scalability and flexibility by adding distributed model components to structured interaction of APIs. The framework focuses on uniting AI-based reasoning and scientific predictive technologies. On the whole, it facilitates more precise and unified analysis of climate considering working together on models and API. [6]: This project introduces an ASAP-Repair, which is an automated method that corrects API misuses based on API Usage Graphs (AUGs). It uses usage patterns and rule of change based on already available methods of detecting API misuse to detect and fix wrong API calls. The algorithm compares misuse graphs to suitable templates and relevance to make graph-related changes to create fixes. It does not depend on test cases in identifying problems like many repair methods. The first findings on a real-world dataset reveal good opportunities, yet it requires optimization in terms of its efficiency and accuracy. [7]: The work presented in this project concerns the creation of an API ecosystem to assistognitively disabled individuals in a range of assistive technologies. It suggests the standardization of the APIs on which applications and services can interact on a structured platform. The ecosystem promotes interoperability, and thus, developers are easy to devise flexible and customized support tools. It enhances accessibility and user-focus because of the relationship of several services. In general, the methodology is designed to improve digital inclusion and autonomy among individuals with cognitive limitations. [8]: The proposed project presents Poirot, which is a deep learning-based product detection of API misuses in software programs. It does not use pre-determined frequency thresholds, but instead it represents API usages as graphs and uses a labeled graph convolutional network to learn usage patterns automatically. The system compares the API elements relationships to detect improper and dangerous usage. The presented research indicates that Poirot is more precise and recalls better than the currently used methods of misuse detection. In general, it increases the precision of automated API error detection with the help of graph-based learning. [9]: This project introduces EstiDroid, which is a tool that approximates the amount and the type of API calls made in Android applications by a means of a static analysis. It does not even run the app, which means it analyses the code used in the application to forecast the use of the API. The method aids the developers in getting an insight into the behavior of an app, the risks that may arise and also the resources used early in the development. Empirical evidence shows that EstiDroid is an efficient estimator that gives correct estimates. In general, it facilitates the analysis and security assessment of Android applications. [10]: The project takes a look at the difficulties concerning the management of API change in the microservice architecture where the services develop separately. Breaking changes are usually discovered only during the runtime, so teams need to track them manually and notify people, and this might cause some flaws and delays. The research determines typical evolution strategies and practical issues based on the interviews issued to industry specialists. It suggests fully automated processing of structural and behavioral REST API changes on the source code and automated communicating to the affected teams. In general, the study will enhance accuracy and reliability of API change management in micro services.

III. FLOWCHART



© 2026 IJSRET

IV. SYSTEM ARCHITECTURE

The Decentralized API Trust and Usage Meter consists of a three-level architecture that isolates the issues of blockchain, backend services, and frontend presentation. Scalability, maintainability, and security are guaranteed by this modular design as well as offering a smooth user experience to the API consumers and providers.

Architectural Overview There are three main layers in architecture and through these layers a complete functioning decentralized application is provided. The trust anchor is the media for smart contracts that perform API registrations and settlement of payments on an immutable registry, which is the Blockchain Layer. The Back-end Layer is an intermediary, performing API usage tracking, authentication and data persistence and keeping in touch with the blockchain. The Front-end Layer will give a user-friendly web-based interface, which allows the users to communicate with the system using their web browsers and Meta-Mask wallets.

Component Interaction The interaction flow starts by a user intersecting with the front-end application with his Meta-Mask wallet. The front-end connects to the back-end via RESTful API endpoints by making authenticated requests with contents of the wallet address of the user and cryptography signatures. These signatures are authenticated by the back-end based on industry-standard SIWE (Sign-In -With Ethereum) protocols, which requires that all requests can be made by their rightful owners of the wallet. Upon the registration of a new service by an API provider, the front-end receives the API name and pricing data and sends them to the back-end. This information is stored in local SQLite store and at the same time registered on the smart contract using Web3.py integration. This setup of using two storage means helps guarantee the availability of data is possible even in the event that the blockchain is unavailable at any given moment; nevertheless, it retains the auditing trace of blockchain technology.

Data Flow Management The system provides an elaborate data movement engine that is performance and cost efficient. Tracker API usage is completely server based, where one request adds a counter in a local database. This method will remove the use of request-per-request transactions in the blockchain, something that would be expensive in terms of gas fees and latency. Rather, usage data is added to the system until a settlement threshold is achieved, then only one batch transaction is revitalized to the blockchain. The back-end manages three major data tables that include users that store wallet data and nonces used to authenticate, APIs that store registered service details and ownership logs and usage data, and usage and payment logs stored in the usage logs table. This relational design allows the efficient querying and reporting and provides the data integrity via the SQL Alchemy ORM layer.

Security Architecture Security Inbuilt in each of the levels of the system architecture. The blockchain layer offers cryptographic assurances in terms of smart contract immortality and finality of transactions. Every API registration and payment settlement is documented indefinitely forming an auditable history which eliminates disputes. The nonce-based authentication is used in the backend to avert attacks on replay, and with each successful authentication, a new nonce is generated which is required in making subsequent requests. Communication via the front-end is conducted only via HTTPS on deployment, but on the local development system, CORS policies are thoroughly tuned so as to allow unauthorized cross-origin access. All the work to manage all the private keys is in meta mask and no sensitive cryptographic material is ever transferred out of the browsers of the user. No private keys are stored in the back-end and authentication is done by means of signature checking.

Scalability Requirements The design is constructed in a scalable manner. The batch settlement model used in the back-end enables the back-end to have the minimal blockchain interactions, which means that the system can accommodate thousands of API calls per day and settle only after a certain period. SQLite, being lightweight, has sufficient performance when deployed to a local disk, and can be scaled to a production database, such as PostgreSQL, with no architectural adjustments, and with much of the detailed usage logging being handled by the back-end. This is a combination of both the transparency of blockchain and the performance of real-time API consumption.

V. METHODOLOGY

The implementation and development of the Decentralized API Trust and Usage Meter was a systematic and gradual process that aimed at creating reliability, security, and usability. This methodology involves all the stages of the development of the initial requirement study to the final stages of testing and deployment with each stage building on the other to have an integrated and functioning system.

Phase 1: System Design and Requirement Analysis
The project started out by developing an in-depth analysis of its requirements of the decentralized API monetization platform.

© 2026 IJSRET

It identified key stakeholders in API providers who wanted to earn money and API consumers who wanted to have transparent access to the services on a pay-per-use basis. Based on the analysis, it was found out that critical requirements such as safe wallet-based authentication, real time usage monitoring, batch payment settlement as well as consuming audit trails were necessary. On these requirements, the system architecture was built, where on-chain and off-chain components were clearly separated. The authorship of local blockchain development with Hardhat was determined to be the best in providing full control of the testing environment without losing compatibility with the production Ethereum networks. The specifications of smart contracts were recorded and included the data structures of API registration and payment settlement, as well as the events that would allow the off-chain monitoring.

Phase 2: Development of the Smart contract

The phase of smart contract development concentrated on the development of a gas-efficient and safe implementation of the fundamental business logic. The API Usage Meter.sol contract was coded in Solidity 0.8.20 with some latest additions of safety features as automatic overflow control and strict access control modifiers. The contact design favored the use of low on-chain storage with only the necessary API metadata being stored and events as an off-chain tracking means. A lot of testing was also done based on Hardhat testing framework along with the API registration, payment settlement and edge cases, including a lack of payment and unauthorized access attempts. Gas optimized and efficient contracting methods such as struct packing techniques and considered functions visibility among others optimized the contract. Other security needs were reentrancy protection, accurate use of require statements and validation of all external inputs.

Phase 3: Back-end Development

The development of the back-end adopted the Fast API application used as the operational node of the system. SQL Alchemy was used to design database models, which have an efficient normal schema to store user information, API registrations, and API usage logs. The authentication framework had been developed based on SIWE (Sign-In-With Ethereum) supported, using the nonce and signature generation and verification. The endpoints of the RESTful APIs have been designed based on consistent and predictable framework of use and interaction with front-end usingOpen API specifications. Proper error handling, input validation and authentication of each endpoint were provided. At this point, the batch settlement logic has been introduced in order to store data about the use in a form which can be settled using the blockchain with race condition and data consistency taken into account. Some integration with Web3.py was used to get the back-end to interoperate with the deployed smart contract, making settlement and queries to on-chain state. The back-end was set to use local host and CORS was set up to accept requests by the Vite development server.

Phase 4: Front-end Development

The front-end was built following React 18.2.0 with Vite as the build tool which was selected due to development experience and optimised production packages. The user interface was also developed having simplicity and clarity, trying not to use complex structures, but plain CSS to explicitly show the main functionality thereof. The integration of wallets was also done through wagmi and viem, and it supported strong connection management with the Meta-Mask. Signing messages is one of the processes involved in establishing an identity with the back-end through the authentication process by following the user connections. The management of the state was processed with React hooks, and there was no integration between business logic and UI components. The API registration form is the one, which has validation of the required fields and correct formatting of Wei values. The usage layer uses real-time statistics such as the number of requests and unpaid money, and they are updated automatically when the user interacts with the system. The settlement interface is user-friendly and takes a friendly approach to the transaction confirmation and error states.

Phase 5: Integration and Testing

Integration testing ensured that all the components interacted freely. Test situations involved end-to-end user flows, wallet connection into API, API registration, API use, and API settlement. The local Hardhat node was used in the testing environment and a fresh deployment was used each time I was testing, which meant that there were consistently repeatable results. The performance testing involved conducting testing on the batch settlement mechanism to ensure that it was possible to accumulate multiple API calls and settle them as one transaction. Load testing was used to simulate many simultaneous users so as to ensure that the back-end was not overwhelmed by the realistic patterns of usage.

Phase 6: Documentation and Preparation of Deployment

There was a development of comprehensive documentation that contained setup instructions to local deployment and API endpoint specifications, and smart contract interface descriptions. Deployment guides has been prepared on different eventualities such as local development, testing and possible production deployment to public test-nets.

© 2026 IJSRET

VI. DISCUSSION

The Decentralized API Trust & Usage Meter project creates a valuable discussion of the future of API monetization, the usefulness of blockchain technology usage, and the quality of decentralization versus usability. These debates go beyond the technical implementation to include economic, social and philosophical ramifications.

The Development of API Economics The conventional API monetization has been based on centralized platforms and they can serve as an intermediary between the providers and the consumers. Such sites are fee based, regulated and have black box accounts. The Decentralized API Trust and Usage Meter disputes such a model by showing that the blockchain technology can facilitate the direct provider-consumer relationships with transparent and verifiable accounting. This is a significant change that has extensive consequences to the economics of APIs. Providers may charge what they want and there are no minimums and maximums imposed on the platform. The consumers only pay to what they consume and they are fully aware of their consumption. The intermediaries will be eliminated which implies that more value would be directed to the people who generate value. Nevertheless, the model assumes users to take care of their wallets and learn about the cryptocurrency concepts which serves as an obstacle on the way of mass adoption that should be broken by enhancing user experience and education.

Minimization of Trust vs. Convenience The system reflects the principle of minimum trust, as the participants do not have to trust each other or some central authority due to the technology which has agreements and provides verifiable records. This is in contrast to conventional systems which entail trusting platform operators, payment processors and auditors. But minimization of trust is at the expense of convenience. The end user is required to handle private keys and be aware of gas charges, as well as wait to receive blockchain confirmation. The batch settlement mechanism is a trade-off, which retains the trust properties of blockchain but offers a better user experience that supports deferred settlement. This gives serious issues concerning the best balance between decentralization and usability implying that hybrid measures can be best suited to mainstream implementation.

The Role of Local Development Environments The reason behind basing this on the environment of local development. project indicates the larger needs in development of blockchain applications. Local testing allows the quick iteration without incurring financial cost, full testing without network integration and accessibility of education without the need to purchase cryptocurrency. This methodology shows that it is possible to have a significant blockchain development without going online, and once finished it can be deployed to live networks as an optional activity. This has some educational implications on blockchain education, indicating that students do not need to study the fundamental concepts with the complexity and cost of main-net interaction. It is also used as a template of developing an enterprise, where in a private setting, applications can be fully tested before being put into the public.

Economic Models and Incentive Alignment Economics The economic model of the project. such that traditional systems could not match incentives between API providers and consumers. This motivates the providers to employ quality since the consumer can easily move to the alternative with known pricing. Customers will be motivated to provide correct reports on the usage since under-reporting would be identified through blockchain verification. The indelible history of dealings brings responsibility that eliminates conflicts. This incentive compatibility is carried to the batch settlement scheme, which is economically efficient to the two parties in terms of minimizing costs of transactions. The providers are paid in larger amounts but at less frequent intervals and the consumers incur less effective fees. The system also highlights the capability of the clever system to generate the positive-sum game in which everybody gains through cooperation.

Considerations of accessibility and inclusion The system has been made better usability for people with disabilities using easily understandable visual elements and easy-to-use interfaces is an example of how careful design can help make blockchain applications more accessible. Yet, there are fresh obstacles that need to be overcome due to the necessity to have the knowledge about cryptocurrencies and manage a wallet. To minimize these barriers, future implementation may include the implementation of features such as social recovery wallets, fiat on-ramps and simplified authentication. The project emphasizes the fact that accessibility is to be taken into consideration at the very first stages of design, not as an addition.

Implications of Regulation and Compliance The system is decentralized which poses interesting regulatory issues. Who monitors the observance of the financial regulations in case of the lack of central operator? What happens to the tax authorities to track income generated via API sales in the event of peer-to-peer payments? These questions are mostly unresolved and will involve constant communication between developers, regulators and law experts. The educational purpose of not

© 2026 IJSRET

having these complications due to the local development of the project, however, any implementation of production would be required to meet the regulatory requirements. This is the conflict between the advantages of decentralization and regulatory adherence that is still one of the most important threats of blockchain adoption.

VII. OUTPUT

As shown in the User interface, back-end, and blockchain layers, the Decentralized API Trust and Usage Meter produces outputs that are clear to the functionality of the system and its transparency.

User Interface Displayed product delivered gives a vivid visual feedback on all significant actions. Once connected to a wallet, the system will show a shortened address of a wallet with a status confirmation. Upon registering an API, a user is presented with a confirmation card containing the API ID, name and price in Wei and ETH. Immediately when using the API, the number of requests being made and outgoing payments are updated, with visual aid of progress bars and usage counters. Successful registrations, settlements, and significant milestones of usage are confirmed by notifications.

Back-end Outputs Out total Product of the backend. structured API registration logs, usage logs and settlement logs. All the information is kept within structured database tables, such as API information, user database and settlement status usage logs. A settlement record linking off-chain usage data to the respective blockchain transaction hash is issued when the payments are processed. The monitoring dashboards and the summary reports would give information about the patterns of usage, busy APIs, the performance of the system, and the outstanding settlements.

Blockchain Results The smart contract creates a long history of verifiable records by blockchain events. Events API registration: service information, payment settlement: the amount of transactions, counting requests, user addresses. Every blockchain transaction generates a transaction hash which can then be verified by trusted independent parties. With these records on-chain, transparency, immutability, and trust is guaranteed.

Monitoring and Audit Outputs The system is equipped with real-time dashboards of metrics like active users, system health, and the responsiveness. Notifications are given to users of limits on spending or inactivity. All major actions are traced in an audit trail with an audit trail which has timestamps, user identification and action details. The platform can give a full and trustworthy track of operations by incorporating off-chain logs and on-chain events. On the whole, the system outputs verify the correct tracking, clear settlements, and effective monitoring, which prove the feasible implementation of decentralized API monetization.

VIII. CONCLUSION

The Decentralized API Trust & Usage Meter project is an effective example of how the blockchain technology can be implemented to the API monetization and transparent tracking of use. The system allows ensuring secure wallet verification, custom API pricing, real-time usage tracking, and batch payment settlement, whereas the vital audit records are ensured by both smart contracts and off-chain storage. The architecture provides a successful separation of the front-end, back-end, and smart contract elements, which guarantees the scalability, efficiency, and easy interaction with Web3 technologies. The project is a learning platform, which presents major concepts as cryptography authentication and gas optimization, and hybrid data management. Even though at the present moment the system is deployed on a local blockchain with a development database, it can be extended to public networks and production with other functions added, such as periodic payments and advanced analytic. In general, the project demonstrates that the decentralized system can provide effective, efficient, and transparent solutions to the current digital economy.

REFERENCES

1. F. Di Lauro, S. Serbout and C. Pautasso, "A Large-Scale Empirical Assessment of Web API Size Evolution," in Journal of Web Engineering, vol. 21, no. 6, pp. 1937-1979, September 2022, doi: 10.13052/jwe1540-9589.2167.
2. C. Roberto Pinheiro, S. Luís Proença Duarte Guerreiro and H. S. Mamede, "A Lightweight Ontology for Enterprise Architecture Mining of API Gateway Logs," in IEEE Access, vol. 12, pp. 128585-128601, 2024, doi: 10.1109/ACCESS.2024.3456119.
3. S. Yang, S. Li, W. Chen and Y. Liu, "A Real-Time and Adaptive-Learning Malware Detection Method Based on API-Pair Graph," in IEEE Access, vol. 8, pp. 208120-208135, 2020, doi: 10.1109/ACCESS.2020.3038453.
4. M. Narendra, T.S.Harshini and L. Jani Anbarasi, "Advancing Skin Disease Diagnosis: A Multimodal Approach Utilizing Telegram Api Token Chatbot for Text and Image Analysis in Skin Disease Classification," in

© 2026 IJSRET

IEEE Access, vol. 12, pp. 189009-189023, 2024, doi: 10.1109/ACCESS.2024.3516884.

5. A.Udrescu and D. -M.Popovici, "Agora: A Distributed Language Model Framework With API-Call Support for Integrated Climate Forecasting," in IEEE Access, vol. 13, pp. 84112-84129, 2025, doi: 10.1109/ACCESS.2025.3568028.
6. S. Nielebock, P. Blockhaus, J. Krüger and F. Ortmeier, "ASAP-Repair: API-Specific Automated Program Repair Based on API Usage Graphs," 2024 IEEE/ACM International Workshop on Automated Program Repair (APR), Lisbon, Portugal, 2024, pp. 1-4,doi: 10.1145/3643788.3648011.
7. R. Hervás, V. Francisco, E. Concepción, A. F. G. Sevilla and G. Méndez, "Creating an API Ecosystem for Assistive Technologies Oriented to Cognitive Disabilities," in IEEE Access, vol. 12, pp. 163224-163240, 2024,doi: 10.1109/ACCESS.2024.3487308.
8. Y. Li, T. N. Nguyen, S. Wang and A. Yadavally, "Poirot: Deep Learning for API Misuse Detection," 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), Lisbon, Portugal, 2024, pp. 302-303, doi: 10.1145/3639478.3643080.
9. W. Fan, D. Zhang, Y. Chen, F. Wu and Y. Liu, "EstiDroid: Estimate API Calls of Android Applications Using Static Analysis Technology," in IEEE Access, vol. 8, pp. 105384-105398, 2020, doi: 10.1109/ACCESS.2020.3000523.
10. A. Lercher, "Managing API Evolution in Microservice Architecture," 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), Lisbon, Portugal, 2024, pp. 195-197, doi: 10.1145/3639478.3639800-

© 2026 IJSRET