International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 10 | Oct - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Decentralised Advertisement Marketplace

Asst. Prof. Priya Kamble[1], Hardik Wankhede[2], Atharva Pandhare[3], Ketan Chourasia[4], Swarup Bobde[5]

[1]Asst. Prof. Priya Kamble, Department of Industrial IoT, Priyadarshini College of Engineering
[2]Hardik Wankhede, Department of Industrial IoT, Priyadarshini College of Engineering
[3]Atharva Pandhare, Department of Industrial IoT, Priyadarshini College of Engineering
[4]Ketan Chourasia, Department of Industrial IoT, Priyadarshini College of Engineering
[5]Swarup Bobde, Department of Industrial IoT, Priyadarshini College of Engineering

---

Abstract - This paper presents the design of Decentralised Advertisement Marketplace, a web-based advertising marketplace that uses blockchain infrastructure and smart contracts to coordinate advertisers and publishers. Conventional advertising services commonly place a central platform between these parties, which can make it difficult to independently inspect transaction handling, fees, and payment status. The proposed model allows advertisers to publish campaigns and publishers to offer advertising inventory, with bidding and settlement rules represented through smart contracts. A hybrid architecture keeps essential transaction evidence on-chain while storing application records in a conventional database. The design aims to improve auditability and reduce reliance on a single intermediary. Practical deployment would still need to address transaction costs, scalability, privacy, and reliable verification of ad delivery.

Key Words: blockchain, smart contracts, digital advertising, Solana, Web3, decentralised marketplace.

1. INTRODUCTION

Online advertising is an important channel through which businesses reach potential customers. In many existing arrangements, a central service coordinates advertisers, website owners, ad placement, reporting, and payment. This model is convenient at scale, but participants may have limited independent visibility into how campaign activity is recorded, how fees are calculated, or how disputes are resolved.

Decentralised Advertisement Marketplace explores a different arrangement: advertisers and publishers interact through a web application, while selected agreements and financial events are handled by blockchain programs. An advertiser can define campaign requirements and a budget; a publisher can make advertising space available and consider relevant offers. Smart contracts can enforce predefined conditions and produce records that participating parties can verify.

The proposal combines a web interface with backend services, off-chain data storage, caching, and Solana-based smart-contract functionality. Blockchain records do not, by themselves, prove that a human genuinely viewed or clicked an advertisement. Therefore, any production system would also require a carefully designed ad-delivery and fraud-detection process.

2. RELATED WORK

Centralised advertising networks offer tools for campaign management, audience reach, and performance reporting, but their intermediary role can create concerns about transparency, platform dependence, privacy, and fees. Blockchain-based approaches have been explored as a way to make selected transactions inspectable and to reduce the need for a single organisation to control every stage of a commercial interaction.

A blockchain maintains a shared record of submitted transactions, while a smart contract executes program logic according to its coded rules. These capabilities may support campaign registration, bidding, agreement tracking, and payment settlement. Decentralisation does not automatically solve all advertising problems: transaction throughput, network fees, user privacy, malicious activity, and adoption remain important design constraints. This project applies these general ideas to an advertiser–publisher marketplace and relies on established platform documentation for its technical foundation.

3. PROBLEM STATEMENT

In a conventional advertising workflow, a central provider often controls the systems used to match campaigns with publishers, report activity, and process payments. Advertisers and publishers may consequently depend on the provider's records when checking delivery or resolving disagreements. The proposed system investigates whether blockchain-based transaction records and rule-driven settlement can make this relationship more transparent. Key issues addressed:

- *Limited visibility:* participants may not have a shared, independently verifiable record of important campaign and payment events.
- *Intermediary dependence:* a central platform controls access to the marketplace and may impose service charges or operating rules.
- *Settlement disputes:* parties need a consistent way to determine whether agreed payment conditions have been met.
- *Data governance:* advertising services handle user and campaign information that should not be exposed unnecessarily.

The proposed approach records selected transaction evidence on-chain and keeps larger or more frequently changing application records off-chain. This split is intended to support auditability without placing all marketplace data on a public ledger.

4. SYSTEM OBJECTIVES

The main goal is to define a marketplace where advertisers and publishers can manage advertising arrangements with clearer transaction records and less dependence on a single intermediary.

- Provide a common interface for advertisers to create campaigns and publishers to list available advertising spaces.

- Support bidding or offer selection using rules that are clear to both parties.
- Use smart contracts to execute supported agreement and settlement steps.
- Keep verifiable references to important campaign, bid, and payment events.
- Separate public transaction evidence from private or operational application data.
- Provide a structure that can be extended with ad verification, analytics, and fraud controls.

5. SYSTEM ARCHITECTURE

The proposed platform is organised into layers. Each layer has a defined responsibility, allowing the user interface, application logic, blockchain program, and database to evolve without requiring every item of data to be stored on-chain.

The overall architecture is shown in Fig. 1.

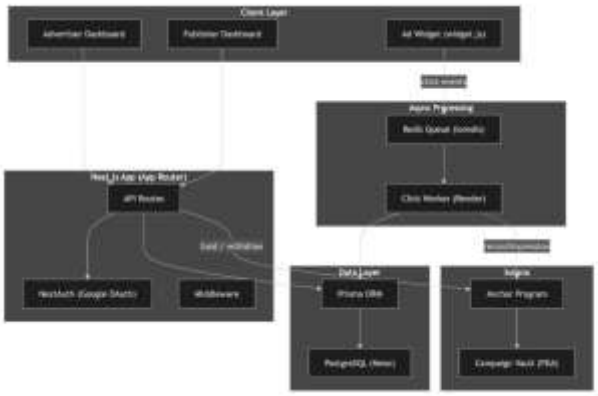


Fig -1: System architecture of Decentralised Advertisement Marketplace, showing the client, application, asynchronous-processing, data and Solana layers.

5.1 User Interface Layer

The frontend can be implemented with Next.js and TypeScript [7]. It provides separate workflows for advertisers, publishers, and administrators. Advertisers manage campaign details, budget limits, and offers; publishers maintain their inventory and review campaigns; administrators can perform permitted oversight tasks. A wallet connection, supported by the Solana JavaScript client library [11], enables users to approve blockchain transactions when required.

5.2 Application and Backend Layer

Backend services validate requests and coordinate the frontend, database, and blockchain. Their responsibilities include account and campaign management, publisher-space records, bid handling, application programming interface (API) responses, access control, and communication with smart contracts. Application-level validation remains necessary even when a blockchain program is used, because not every operation belongs on-chain.

5.3 Blockchain and Smart-Contract Layer

The design uses Solana with the Anchor framework for program development [1], [2], [3], [4]. A smart contract can encode supported campaign registration, bid or agreement operations, payment conditions, and transaction events. The program should verify authority and input constraints before accepting an operation. Once a transaction is confirmed, its on-chain record can be inspected independently, although off-chain claims about ad impressions still require separate evidence.

Potential contract functions include campaign registration, offer submission, agreement confirmation, settlement, and recording of relevant events. The exact functions and safeguards depend on the final implementation and must be tested before deployment.

5.4 Database Layer

A relational database such as PostgreSQL [8] can store information that is not suitable for direct blockchain storage: user profiles, campaign descriptions, publisher details, application settings, and marketplace metadata. Prisma [5], [6] can provide a structured interface between the backend and the database. On-chain transaction signatures or identifiers can be stored alongside the related application records so users can cross-check important events.

5.5 Caching and Supporting Services

Redis [9] can cache frequently requested marketplace information and hold short-lived operational data. Caching may reduce repeated database reads and improve responsiveness, but cached values should not be treated as authoritative proof of a blockchain transaction. The backend should reconcile critical state with the relevant database or blockchain source.

5.6 High-Level Transaction Flow

Advertiser creates campaign → Publisher lists ad space → Offers or bids are considered → Agreement is confirmed → Smart contract processes the permitted transaction → Transaction reference is stored → Both parties review campaign status.

In a typical workflow, a user signs in and connects a wallet. The advertiser submits campaign requirements, while publishers describe available inventory. The marketplace presents suitable opportunities and supports an offer or bidding process. After the parties accept an arrangement, the smart contract checks the required conditions and processes the relevant transaction. The backend stores supplementary details and the transaction reference, which can later be displayed in the application.

The advertiser and publisher workflows are illustrated in Fig. 2.



Fig -2: Advertiser workflow (left) and publisher workflow (right).

5.7 On-Chain and Off-Chain Data

The system uses a hybrid storage model. On-chain records may include transaction identifiers, contract events, settlement references, and the minimum data needed to verify an agreement. Off-chain storage can retain detailed profiles, campaign descriptions, publisher information, and interface-related metadata. Keeping large and potentially sensitive records off-chain reduces unnecessary public exposure and avoids placing bulky data directly on the blockchain.

5.8 Security and Transparency

Blockchain records can make confirmed transactions more difficult to alter unnoticed, while smart contracts apply programmed rules. Wallet signatures help associate an action with control of a blockchain account, but they do not prove a person's real-world identity. Role-based access control should restrict application features according to user permissions. The implementation should also consider key security, input validation, contract testing, abuse prevention, privacy, and recovery procedures.

6. HARDWARE REQUIREMENTS

The project is primarily a software platform and does not require dedicated Internet of Things (IoT) sensors or microcontrollers for its core functions. Development and testing can be performed on a standard computer with internet access.

- Computer or laptop for coding, deployment, and testing.
- Intel Core i3/i5 or an equivalent processor as a practical baseline.
- At least 8 GB of random-access memory (RAM) recommended for a smoother development workflow.
- Approximately 256 GB solid-state drive (SSD) storage recommended for development tools and project files.
- Internet connectivity for blockchain access, package installation, and deployment.
- A smartphone is optional for testing responsive layouts and compatible wallet interactions.

7. SOFTWARE ARCHITECTURE

The software design can be viewed as four cooperating layers. The presentation layer provides the web interface; the application layer applies business rules and exposes APIs; the blockchain layer executes supported transactions; and the database/support layer stores operational data.

1) Presentation: Next.js and TypeScript provide the screens for campaign creation, publisher inventory, wallet connection, bidding, and transaction status.

2) Application services: backend APIs manage users, campaign data, bidding requests, validation, permissions, and communication with the smart contract.

3) Blockchain programs: Solana programs developed in Rust [10] with Anchor handle supported transaction logic, enforce contract constraints, and emit records for important events.

4) Data and cache services: PostgreSQL stores off-chain application data, Prisma supports database access, and Redis provides temporary caching. Transaction references connect application records to blockchain activity.

Representative screens of the prototype interface are shown in Fig. 3.



Fig -3: Prototype interface screens: landing page, publisher dashboard, analytics view and wallet transaction confirmation.

8. EXPERIMENTAL RESULTS AND EVALUATION

This paper presents the architecture and design of the marketplace; measured performance results are not yet available, so no numerical claim about throughput, latency, cost savings, fraud reduction, or scalability is made. A complete evaluation should measure transaction confirmation time, application response time, database performance, contract execution costs, and behaviour under concurrent users. It should also test invalid inputs, unauthorised actions, failed transactions, and attempts to submit fraudulent impression or click events.

9. CONCLUSION AND FUTURE SCOPE

Decentralised Advertisement Marketplace proposes a blockchain-supported workflow for connecting advertisers with publishers. The design uses smart contracts for selected agreement and settlement operations, while a conventional database stores detailed application information. This combination can make important transaction events easier to verify and may reduce reliance on a central intermediary. However, blockchain alone cannot confirm that an advertisement was genuinely delivered to a real person, and a working deployment must address privacy, transaction costs, contract security, and abuse prevention.

Further development could add machine-learning-assisted campaign matching, auction mechanisms with dynamic pricing, privacy-preserving reporting, support for additional blockchain networks, and a dashboard for impressions, clicks, and revenue. Reliable delivery verification and stronger scalability testing should be treated as core engineering tasks rather than optional additions.

ACKNOWLEDGEMENT

The authors thank the Department of Industrial IoT, Priyadarshini College of Engineering, for its guidance and support during this work.

REFERENCES

1. Solana Foundation: Solana Documentation: Core Concepts and Programs. Online documentation. https://solana.com/docs/core
2. Solana Foundation: Solana Program Examples. Online documentation. https://solana.com/docs/programs/examples
3. Anchor: Anchor Framework Documentation. Online documentation. https://www.anchor-lang.com/docs
4. Anchor: Anchor Framework Quickstart. Online documentation. https://www.anchor-lang.com/docs/quickstart
5. Prisma: Prisma ORM Documentation. Online documentation. https://www.prisma.io/docs

6. Prisma: Prisma ORM with PostgreSQL: Quickstart. Online documentation. https://www.prisma.io/docs/prisma-orm/quickstart/postgresql
7. Next.js: Next.js Documentation. Online documentation. https://nextjs.org/docs
8. PostgreSQL Global Development Group: PostgreSQL Documentation. Online documentation. https://www.postgresql.org/docs/
9. Redis: Redis Documentation. Online documentation. https://redis.io/docs/
10. Rust Foundation: The Rust Programming Language. Online documentation. https://doc.rust-lang.org/book/
11. Solana Foundation: Solana Web3.js: JavaScript Client Documentation. Online documentation. https://solana.com/docs/clients/javascript