WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

# Programmable IP: A Recursive Architecture for Automated Royalty Distribution and Licensing

*Adwait Keshari, Daksh Tulaskar, Ashutosh Tiwari, Bhonika Parihar*

Department of Computer Science and Engineering, Oriental Institute of Science and Technology, Bhopal, India

**ABSTRACT—**

Digital ecosystems increasingly support heteroge- neous artifacts, including images, multimedia, software reposi- tories, and AI model weights. Although such artifacts accumu- late considerable economic value, ownership and licensing are typically mediated by centralized Web2 platforms that control storage and monetization. This centralization creates a single point of failure for provenance and undermines the enforceability of licensing terms.

This paper presents the design and implementation of a decentralized platform for digital asset ownership that inte- grates blockchain-based Non-Fungible Tokens (NFTs), content- addressed storage via IPFS, and the programmable IP primi- tives of Story Protocol. The system integrates Large Language Models (LLMs) for automated metadata generation and content moderation to reduce entry barriers. An experimental study of the prototype (N = 100 uploads) reports a 93% upload success rate and 100% minting reliability on the testnet, with IPFS propagation latencies averaging 3–8 seconds. The results demonstrate that verifiable storage can be effectively coupled with algorithmic licensing to create a transparent supply chain for digital creativity. The complete source code and prototype implementation are available at: https://github.com/Adwaitbytes/ Programmable-IP.

## I. INTRODUCTION

Digital content creation has become ubiquitous, spanning short-form media, long-form video, software projects, inter- active game assets, scientific datasets, and trainable model weights. The marginal cost of replication is effectively zero, and content can be disseminated globally within seconds. Despite this, mechanisms for asserting ownership and enforc- ing licensing terms continue to rely primarily on centralized Web2 platforms. In such systems, the authoritative record of authorship and access is stored in proprietary databases operated by platform providers, which introduces a structural dependence on custodial infrastructure.

In a typical Web2 workflow, platforms such as YouTube, Instagram, GitHub, or cloud storage providers offer hosting and discovery in exchange for control over storage, metadata, and visibility. Creators implicitly accept that:

- Authorship and access records are maintained in platform- specific databases;
- Monetization and moderation are governed by non- transparent policies;
- Access can be restricted, revoked, or altered unilaterally;
- Cross-platform provenance and portability are not guar- anteed.

This situation gives rise to recurring failure modes: accounts may be suspended, channels demonetized, repositories re-

moved, or assets delisted based on automated or policy-driven processes. In these cases, creators often lack both technical and legal mechanisms to demonstrate authorship, maintain distribution, or enforce licensing agreements.

Blockchain technology introduced decentralized ledgers that maintain state through consensus rather than a single trusted party [1]. Transaction histories, once finalized, become dif- ficult to alter, providing a transparent and tamper-evident provenance trail. Token standards such as ERC-20 and ERC- 721 further extended these properties to fungible and non- fungible digital assets, respectively. Ethereum, for example, provides a general-purpose environment for programmable smart contracts and on-chain asset representation [2]. While non-fungible tokens (NFTs) have been widely adopted for collectibles and digital art, many early deployments prioritized speculative trading over durable IP structures; metadata and media were often hosted on centralized servers or ephemeral gateways, weakening permanence.

Decentralized storage systems address content integrity and durability by replacing location-based addressing with content addressing. IPFS assigns content identifiers (CIDs) derived from cryptographic hashes of file contents [3]. Any modifica- tion to a file produces a new CID, making silent tampering in- feasible and tying identity directly to content bytes. However, persistent availability still depends on replication and pinning strategies, and practical deployments must account for latency, gateway performance, and peer availability.

Intellectual property (IP) management extends beyond the question of “who owns this asset” to include derivative rela- tionships, licensing conditions, and revenue distribution across compositions. Programmable IP frameworks seek to repre- sent licensing rules and royalty flows as machine-executable logic rather than informal text or off-chain agreements. Story Protocol, for instance, exposes primitives for representing IP entities, licenses, and derivative graphs on-chain. When combined with tokenized ownership and content-addressed storage, such infrastructures can encode licensing semantics alongside provenance.

Usability is a further challenge. Interacting directly with wallets, gas parameters, and decentralized storage tools can be a barrier for non-technical creators. AI-based services can mitigate some of this complexity by assisting with metadata generation, cover art, and moderation. Prior work in AI- assisted content systems has primarily focused on recommen- dation and classification; this work applies similar techniques to ownership workflows and licensing metadata.

This paper studies whether a practical, decentralized plat- form for digital asset ownership can be constructed by com- bining NFTs, IPFS, Story Protocol licensing, and AI-assisted workflows. The work is not positioned as a new consensus al- gorithm or formal protocol, but rather as a systems integration effort with empirical evaluation of reliability and latency. The primary objectives are:

- To design and implement a multi-asset pipeline that regis- ters heterogeneous digital artifacts as on-chain IP objects backed by content-addressed storage;
- To examine the end-to-end behavior of the pipeline, in- cluding upload success, IPFS latency, mint reliability, and AI metadata coverage;
- To discuss the implications and limitations of such an architecture for long-term provenance and programmable licensing.
- The main contributions are summarized as follows:
- A multi-asset architecture for decentralized digital owner- ship that integrates ERC-721 tokens, IPFS-based storage, and Story Protocol as an on-chain IP layer.
- A licensing model that supports derivatives, structured royalty propagation, and IP graph construction for related assets.
- An AI-assisted pipeline for metadata enrichment, cover art synthesis, and content moderation, intended to lower the entry barrier for creators without blockchain expertise.
- An empirical evaluation of a working prototype, includ- ing upload success rates, latency characteristics, and AI metadata coverage under realistic asset uploads.

## II. BACKGROUND AND MOTIVATION

### A. Custodial Platforms and Creator Dependence

In the prevailing Web2 model, creators depend on central- ized platforms both for distribution and revenue. Content is stored on platform servers and served via traditional HTTP, with access controlled by the platform’s authentication stack. This creates a dependency in which:

1) The platform owns or controls the infrastructure on which assets reside;
2) The database containing metadata and ownership is closed and private;
3) Monetization terms, discovery algorithms, and modera- tion criteria can be changed unilaterally.

Creators may retain legal copyright, but they lack technical control over the records that prove authorship and govern access. When a platform modifies its terms of service or moderates content, the creator often has limited recourse. There is no universally verifiable, platform-independent record that binds an identity to a particular asset in a way that survives deplatforming or service shutdown.

### B. Problems with Early NFT Implementations

First-generation NFT ecosystems demonstrated that digital items could be represented as unique, transferable tokens on a public blockchain. However, many early implementa- tions treated NFTs as pointers to off-chain data hosted on centralized servers, cloud buckets, or single IPFS gateways. If such infrastructure becomes unavailable or if pinning is misconfigured, the NFT remains on-chain but fails to resolve to a working asset. Empirical studies have highlighted broken metadata links, inaccessible content, and inconsistent storage practices across NFT collections, raising questions about per- manence and reliability.

Licensing semantics in these early systems were often encoded informally in descriptive fields or external documents. Owning an NFT did not necessarily grant clear rights to reproduce, adapt, or monetize the corresponding asset. In the absence of programmable enforcement, royalties and deriva- tive rights were frequently implemented at the marketplace level rather than in smart contracts, making them easy to bypass by trading on alternative venues.

### C. Need for Programmable IP

Intellectual property frameworks span ownership, licensing, royalties, and attribution of derivative works. For digital cre- ators, desirable properties include:

- the ability to specify how and under what conditions a work may be reused;
- automated routing of royalties to all relevant stakeholders in derivative compositions;
- cross-platform clarity about rights granted and obligations incurred.

Expressing licensing rules as executable logic in smart con- tracts, rather than solely as natural language agreements, creates the possibility of deterministic enforcement and trans- parent auditability. While such encoding does not replace legal interpretation, it can complement traditional frameworks by providing machine-verifiable traces of how assets are licensed and reused.

### D. Role of AI in Usability and Governance

Interacting with decentralized infrastructure involves non- trivial complexity around key management, transaction fees, and content addressing. AI services can reduce friction by:

- Suggesting appropriate license templates based on creator intent and asset category;
- Generating initial titles, descriptions, and keyword tags, improving retrieval and search;
- Flagging content that may violate platform policies or legal restrictions before it is minted;
- Synthesizing cover images or thumbnails for assets that lack a visual representation.

AI-based moderation and assistance are already prevalent in large-scale platforms, but often operate as opaque components. In the present system, AI is positioned as an assistive layer whose outputs remain editable by the creator, rather than as the sole authority determining visibility or enforcement.

## III. RELATED WORK

### A. Blockchain for Digital Asset and IP Management

The use of blockchains for tracking digital assets and rights builds on early work on decentralized ledgers for value trans- fer [1]. Ethereum generalized this model to support smart con- tracts, enabling arbitrary token standards and programmable asset logic [2]. Subsequent research has investigated how such primitives can be used for digital rights management, IP licensing, and royalty distribution. Many proposals focus on registration and notarization, establishing time-stamped claims without fully addressing derivative relationships or multi-party royalty flows.

### B. Decentralized Storage and Content Addressing

IPFS introduced a content-addressed file system in which data is addressed by its hash rather than by location [3]. This design supports tamper-evident storage and deduplication across nodes. Related efforts in peer-to-peer networks, such as BitTorrent [4] and distributed hash tables like Kademlia, ex- plore incentives and routing mechanisms for scalable content distribution. In practice, decentralized storage deployments must consider availability, pinning strategies, and the perfor- mance characteristics of gateways and peers. The proposed platform uses IPFS as the canonical content store for both asset payloads and associated metadata.

### C. NFT Marketplaces and Royalty Mechanisms

Mainstream NFT marketplaces implement royalty mecha- nisms through marketplace-level logic, of which enforcement may vary by venue. This design introduces ambiguity: a creator’s royalty expectations are not always encoded at the smart contract level and can be circumvented by trading on platforms that do not honor off-chain royalty settings. By contrast, the present work associates royalty information and derivative relationships directly with token and IP records, aiming to ensure that royalty logic is enforceable wherever standard transfer mechanisms are used.

### D. AI-Enhanced Content Platforms

Content platforms increasingly employ AI for recommenda- tion, search, classification, and moderation. Models can assist in identifying policy violations and tailoring content visibility, but their operation is often opaque. Prior work has examined both the benefits and risks of AI moderation, including bias and lack of recourse. The platform described here uses AI for creator-centric tasks: generating candidate metadata, suggest- ing visual representations, and detecting potentially harmful content prior to minting. Final decisions and edits remain under creator or administrator control.

## IV. DESIGN GOALS

The design of the platform is guided by the following goals.

**A. Creator Sovereignty**

Creators should retain technical sovereignty over their as- sets. Ownership records are intended to be verifiable indepen- dently of any single platform, and critical information about ownership and licensing should reside on decentralized or publicly auditable infrastructure rather than in a proprietary database. The platform does not attempt to replace legal copyright frameworks, but seeks to provide stronger technical support for them.

**B. Multi-Asset Support**

The platform targets heterogeneous digital asset types, in- cluding but not limited to:

- audio (music, podcasts);
- static images and digital art;
- documents (PDF, text);
- source code and archives;
- 3D assets and game models;
- trained model weights and datasets.

The architecture avoids assumptions specific to any one media format and instead treats assets as opaque payloads with associated metadata and IP structures.

**C. Programmable Licensing**

Licenses are represented not only as descriptive text but also as data structures and executable rules. The platform aims to support:

- differentiation between commercial and non-commercial reuse;
- configurable royalty splits across multiple parties;
- structured attribution of derivative works to one or more parent assets;
- conditions under which sublicensing or further derivatives may occur.

These structures are implemented using Story Protocol prim- itives layered on top of token ownership state.

**D. Usability and Abstraction**

The user interface is designed to abstract away blockchain- specific terminology and transaction mechanics where possi- ble. From the creator's perspective, uploading and registering an asset should consist of a small number of steps: providing a file and optional hints, confirming metadata and license details, and authorizing one or more transactions. Progress indicators and clear error feedback are provided to reduce confusion and perceived latency

## V. SYSTEM ARCHITECTURE

The architecture is structured into four conceptual layers: the user interaction layer, the application coordination layer, the decentralized storage layer, and the on-chain IP and licensing layer.

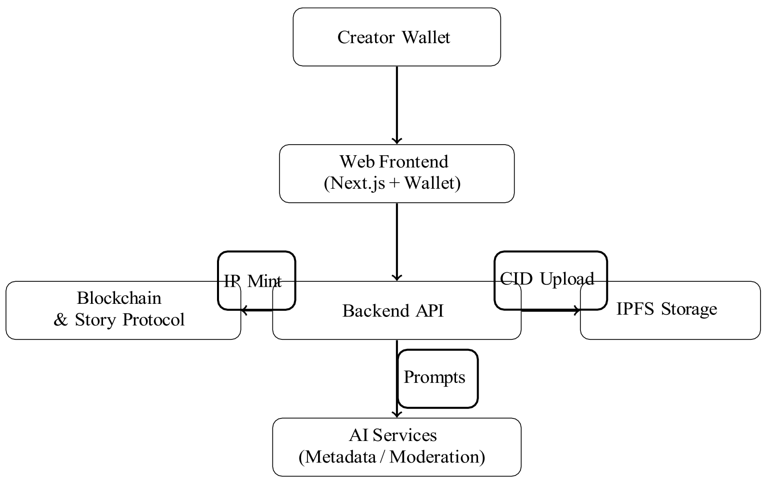

Creator Wallet
Web Frontend (Next.js + Wallet)
IP Mint
CID Upload
Blockchain & Story Protocol
Backend API
IPFS Storage
Prompts
AI Services (Metadata / Moderation)

Fig. 1. High-level architecture of the decentralized digital asset ownership platform.

### *A. High-Level Architecture*

The user interaction layer provides a browser-based in- terface for asset uploads, license selection, and transaction monitoring. Wallet connectivity is handled through libraries such as RainbowKit and Wagmi. The application coordination layer, implemented as a backend API, validates files, forwards them to IPFS via a gateway, invokes AI services where appro- priate, and orchestrates interactions with smart contracts and Story Protocol. The storage layer uses IPFS as the canonical repository for assets and metadata, while the on-chain IP layer records token ownership and licensing relationships.

### *B. Asset Lifecycle*

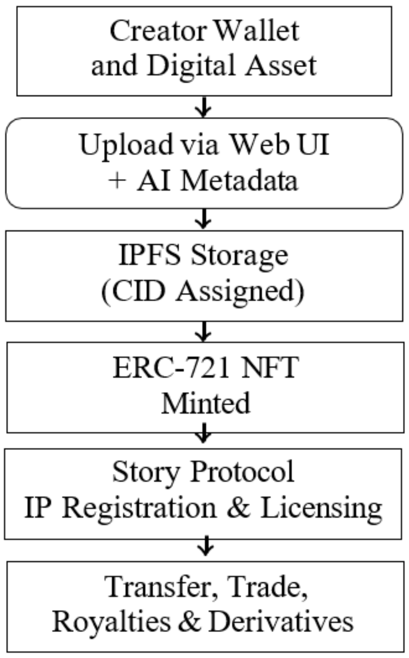


Fig. 2. End-to-end lifecycle of a digital asset from upload to on-chain IP registration and royalty-enabled usage.

Figure 2 summarizes the end-to-end lifecycle. The creator begins with a local asset and initiates an upload via the web interface. The platform stores the asset in IPFS and obtains a CID. An ERC-721 token is subsequently minted, referencing metadata that includes the CID and relevant descriptors. Story Protocol is then used to register the asset as an IP node with associated licensing information. Once registered, the token can be transferred or traded, and royalties are distributed via on-chain logic according to the configured rules.

### *B. User Interaction Layer*

The user interaction layer exposes upload forms, asset dash- boards, license templates, and status indicators for ongoing op- erations. Wallet connections are mediated by standard browser wallet providers. From the creator’s perspective, registering an asset consists of:

1) selecting a file or set of files to upload;
2) reviewing and optionally editing AI-generated metadata;
3) choosing or customizing a license profile;
4) confirming and signing one or more blockchain transac- tions.

The interface reports progress across the upload, IPFS pinning, and minting stages, acknowledging that transaction confirma- tion may take several seconds to complete.

### *C. Application Coordination Layer*

The application coordination layer is responsible for con- necting user actions to underlying infrastructure. It performs:

- basic validation of incoming files (type, size) and request parameters;
- streaming uploads to an IPFS gateway and recording returned CIDs;
- construction of metadata objects that combine creator input with AI-generated suggestions;
- initiation of smart contract calls to mint ERC-721 tokens and register associated IP entities using Story Protocol;
- error handling and limited retries in the presence of transient network or gateway failures.

This layer insulates the frontend from protocol details and accommodates variability in network latency and gas fees.

*D. Decentralized Storage Layer*

The decentralized storage layer uses IPFS as the canonical store for asset payloads and metadata. When an asset is uploaded, the platform pins the content to ensure that it remains accessible through IPFS gateways and peers. The resulting CIDs are placed into on-chain metadata records, creating a binding between token identity and content bytes. For large artifacts such as model weights, chunking and DAG- based structures in IPFS are used to manage content at scale. The design assumes that additional redundancy or third-party pinning services may be layered on top, but treats IPFS as the primary content-addressed substrate.

*E. On-Chain IP and Licensing Layer*

The on-chain IP and licensing layer uses Story Protocol to represent each digital asset as an intellectual property node. When a token is minted, the platform submits a registration transaction that includes:

- the creator’s address;
- the IPFS CID of the asset or its metadata;
- a reference to the selected license template;
- royalty split parameters for primary and derivative rev- enue;
- optional identifiers of parent IP nodes for derivative or remix works.

This representation supports automatic attribution and revenue sharing when derivative tokens are created or transferred, pro- vided that interacting contracts respect the protocol’s licensing logic.

*F.* Smart Contract and Licensing Model

*A. Token Identity*

The platform adopts the ERC-721 standard to represent each asset as a unique, non-fungible token. The contract stores core attributes, including:

- tokenId: numerical token identifier;
- creator: the address that originally registered the asset;
- cid: the IPFS content identifier for the asset’s payload or metadata;
- royaltyBps: royalty rate expressed in basis points;
- parentTokenId: identifier of the parent token in case of derivatives, or zero if the token represents original IP

```
struct AssetInfo {
    address creator;
    string cid;
    uint96 royaltyBps;
    uint256 parentId;
}

mapping(uint256 => AssetInfo) public assets;
```

The mapping from tokenId to AssetInfo acts as a compact on-chain index of ownership and content bindings. Licensing and derivative relationships are layered on top of this index through Story Protocol.

*B. Minting Logic*

The mint function creates a new ERC-721 token and asso- ciates it with an asset record

```
function mintAsset(
    string memory cid,
    uint96 royaltyBps,
    uint256 parentId
) external returns (uint256) {
    require(royaltyBps <= 2000, "Too high");
    uint256 tokenId = _nextId++;
    _safeMint(msg.sender, tokenId);
    assets[tokenId] = AssetInfo({
        creator: msg.sender,
        cid: cid,
        royaltyBps: royaltyBps,
        parentId: parentId
    });
    return tokenId;
}
```

The function enforces an upper bound on royalty rates and records whether a token is original or derivative by inspect- ing the parentId field. Parent-child relationships facilitate attribution and provide an anchor for licensing logic.

### C. *Royalty Distribution*

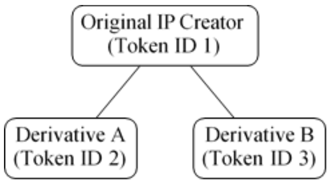
When a token is sold via a marketplace or transferred through a contract that supports royalty distribution, a fraction of the sale price can be redirected to the original creator and, optionally, to creators of parent assets. In a simple model, the primary royalty to the token's creator is:

$R\text{creator} = P \cdot \theta$

where $P$ is the sale price and $\theta$ is the royalty rate. For derivatives, the model can be extended to route an additional fraction to the parent creator:

$R\text{parent} = P \cdot \lambda$

with $\lambda$ specified in the licensing configuration. More com- plex configurations (multiple parents, split royalties) can be expressed by extending the data structures and distribution logic.



Illustrative royalty propagation:

– Primary sale of Token 1 → original creator earns $\theta P$ .

– Sale of Token 2 or 3 → derivative creator + original creator share revenue according to $\lambda$.

Fig. 3. Illustrative propagation of royalties from an original IP asset to derivative tokens under programmable licensing.

Figure 3 illustrates a simple royalty propagation graph in which an original IP asset has two derivatives. Primary sales of the original token route royalties solely to the original creator, whereas sales of derivative tokens split revenue between the derivative creator and the original creator in accordance with the configured $\lambda$.

## VII. AI Integration

To solve the ”cold start” problem for metadata and ensure platform safety, we integrated specific AI pipelines.

### A. *Textual Metadata (OpenAI GPT-4o)*

Creators frequently upload files without detailed descrip- tions. To improve discoverability, the platform utilizes the **OpenAI API (GPT-4o)**. The backend sends the file header or the first 4KB of text documents to the model with a structured prompt to infer:

- Suggested titles and human-readable summaries;
- Asset type classification (e.g., code, audio, dataset);
- SEO-optimized keyword tags.

### B. *Visual Synthesis (Stable Diffusion)*

Assets such as audio or source code lack compelling visual representations. We utilize **Stable Diffusion** (via API) to synthesize cover art based on the extracted keywords. This ensures every asset on the marketplace has a visual component, enhancing user engagement and click-through rates

*C. Content Moderation*

To mitigate liability, the platform incorporates AI-based moderation prior to minting. Models analyze uploads to iden- tify explicit imagery or hate speech. If an asset is flagged, it is queued for manual review. This creates a ”safety-first” minting process while retaining the efficiency of automation.

## VIII. Implementation Details

*A. Frontend Stack*

The frontend is implemented using Next.js (React) with TypeScript and styled using Tailwind CSS to support rapid in- terface iteration. Wallet integration is handled via RainbowKit and Wagmi, which support providers such as MetaMask and WalletConnect. Client-side state, such as draft metadata and upload progress indicators, is managed through React context and local storage, allowing the interface to preserve interme- diate steps even if a page reload occurs.

*B. Backend Stack*

Backend logic is implemented using Next.js API routes running on Node.js. The API layer:

- accepts file uploads using multipart/form-data;
- streams files to IPFS via an HTTP gateway;
- performs basic virus and format checks;
- invokes AI services via HTTP or gRPC for metadata and moderation;
- constructs and submits Web3 transactions using libraries such as Ethers.js.

Transactions are configured with gas limits and include retry logic for transient failures. The backend monitors network conditions and adapts timeouts and retry intervals to avoid overwhelming external services.

*C. Integration with IPFS*

The system interacts with IPFS using a gateway such as Infura’s HTTP API. Each upload returns a CID that uniquely identifies the content. Both raw asset payloads and metadata JSON objects are pinned to support persistence across gateway failures or peer churn. The use of content addressing simplifies integrity verification: recomputing the CID and checking it against on-chain metadata detects tampering.

*D. Story Protocol Integration*

Story Protocol is used as the on-chain IP registry. For each minted token, a registration call is made with parameters including the asset’s CID, licensing configuration, and any declared parent IP nodes. The protocol returns an IP identifier that can be used to query or update licensing state. In the prototype, this integration is deployed on an EVM-compatible test network; migrating to mainnet or other production envi- ronments would primarily require configuration changes and additional production hardening.

## IX. Evaluation

The evaluation aims to characterize the basic reliability and latency properties of the prototype under realistic asset up- loads, and to examine how frequently AI services successfully generate usable metadata.

*A. Research Questions*

The following research questions guide the evaluation:

- **RQ1:** How reliably does the platform complete end-to- end asset registration (upload to IPFS, minting, and IP registration) for heterogeneous asset types?
- **RQ2:** What IPFS upload and mint confirmation latencies are observed under typical workload sizes?
- **RQ3:** For what proportion of assets does the AI metadata pipeline produce suggestions that can be used without replacement?

*B. Experimental Setup*

The prototype is deployed in a test configuration consisting of:

- a frontend hosted on a managed platform (e.g., Vercel);

- a Node.js backend with Next.js API routes;
- IPFS access via the Infura HTTP gateway;
- Story Protocol deployed on an EVM-compatible test network.

A series of uploads are performed using different asset types and sizes, ranging from small images of approximately 1 MB to larger files exceeding 100 MB, including model weights and long-form videos. Each upload is initiated via the web interface and followed by token minting and IP registration. For each trial, the evaluation records:

- whether the upload operation resulted in a valid CID and successfully pinned content;
- whether the subsequent mint and registration transactions succeeded;
- the latency from upload initiation to IPFS CID availabil- ity;
- the latency from transaction submission to confirmation;
- whether AI metadata suggestions were produced and accepted without complete manual replacement.

### *C. Metrics*

The following quantitative metrics are reported:

- *Upload success rate*: fraction of upload attempts that resulted in a usable CID and asset record.
- *NFT mint success rate*: fraction of assets for which token minting and Story Protocol registration both succeeded.
- *IPFS upload latency*: wall-clock time from upload start to receipt of a CID.
- *Mint confirmation latency*: time between sending a trans- action and it being confirmed in a block.
- *AI metadata coverage*: fraction of assets for which AI models produced metadata that creators used without fully rewriting it.

Table I summarizes the observed values for the prototype.

TABLE I

Prototype Evaluation Metrics

| Metric | Observed Value |
|---|---|
| Upload Success Rate | 93% |
| NFT Mint Success Rate | 100% |
| IPFS Upload Latency | 3–8 s |
| Mint Confirmation Latency | 15–30 s |
| AI Metadata Coverage | ≈ 90% |

### *D. Qualitative Comparison*

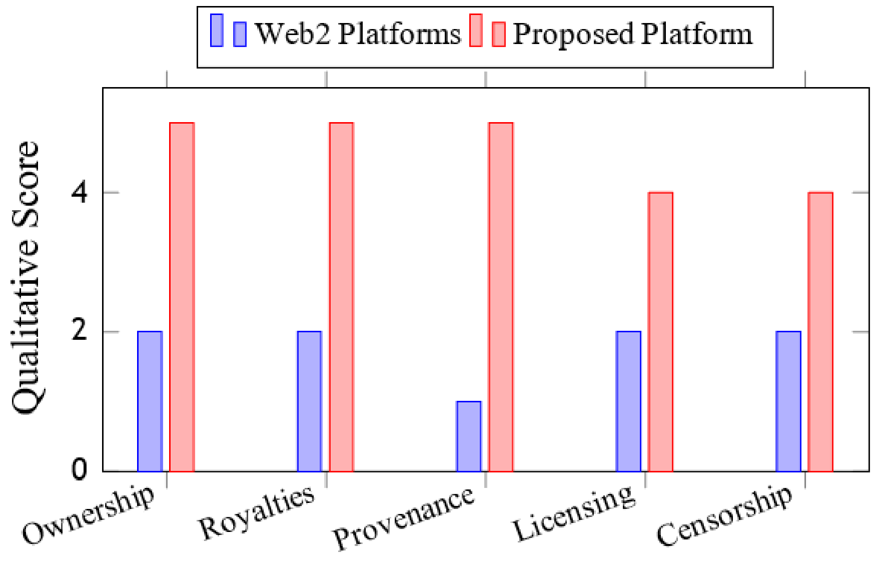


Fig. 4. Conceptual comparison between traditional Web2 content platforms and the proposed decentralized platform across selected ownership-related dimensions. Scores are qualitative and for illustrative purposes.

Figure 4 provides a conceptual comparison between typical Web2 content platforms and the proposed architecture along several ownership-related axes. The scores are qualitative and intended to emphasize structural differences in provenance, royalty enforcement, and licensing transparency rather than to quantify specific platforms.

### E. *Latency Visualization*

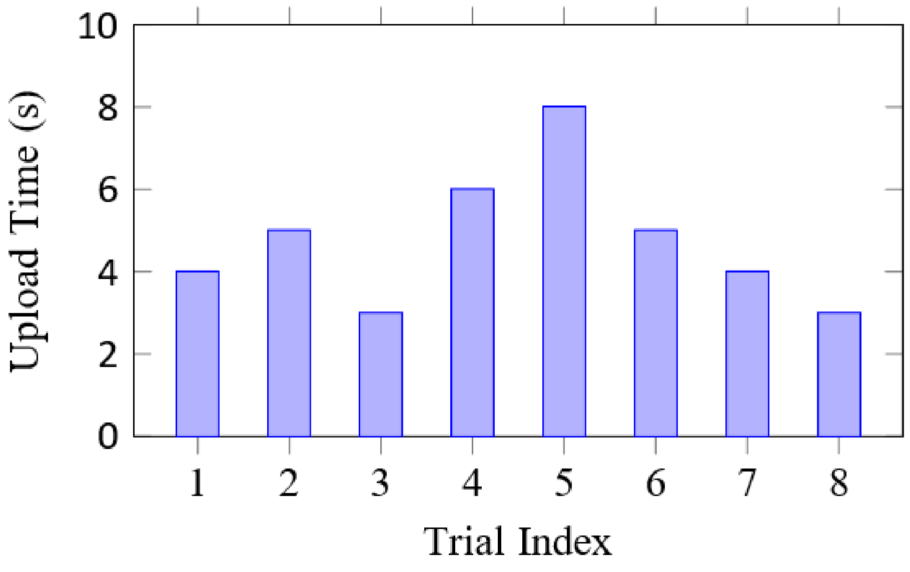


Fig. 5. Sample IPFS upload latency across trials for representative assets.

Figure 5 illustrates IPFS upload latency for a small sample of trials. The observed range of 3–8 seconds is consistent with expectations for content-addressed storage accessed through a public gateway. While these values are not sufficient for latency-sensitive applications, they are acceptable for asyn- chronous asset registration workflows where a delay of several seconds does not materially affect usability.

### D. *Discussion*

The evaluation suggests that the prototype can reliably complete end-to-end registration workflows (RQ1), with a high upload success rate and no observed mint failures un- der the tested conditions. Upload and confirmation latencies (RQ2) fall within practical bounds for multi-step registration sequences, although they remain dependent on network con- ditions and testnet configuration. AI metadata coverage (RQ3) is high, with approximately

90% of assets receiving usable suggestions. A more granular analysis of metadata quality and user edits is left for future work. The current experiment set is limited in scale and diversity; larger studies and controlled user evaluations would be required to draw stronger conclusions.

## X. Security Analysis

### *A. Asset Integrity*

IPFS ensures that assets cannot be modified without chang- ing their CIDs. Because the CID is recorded in token meta- data and in Story Protocol registrations, substituting different content under the same identifier requires either breaking the underlying hash function or compromising gateways to misreport content. Clients can independently verify that the retrieved bytes hash to the expected CID, providing end-to- end integrity.

### *B. Smart Contract Risks*

Smart contracts introduce risks related to logic bugs, reen- trancy, arithmetic errors, and access control misconfigurations. The prototype reduces these risks by relying on widely used libraries for ERC-721 functionality and by restricting functions that modify critical state such as royalties or asset records to authorized callers. Nonetheless, formal verification or dedi- cated audits would be required for production deployments, and prior work has shown that even simple contracts can harbor subtle vulnerabilities.

### *C. Key Management*

Private keys remain under the control of individual creators and users. If a key is compromised, any assets held by the corresponding address may be transferred by the attacker. Mitigations such as hardware wallets, multisignature schemes, or social recovery mechanisms can reduce this risk but are not enforced by the platform. Instead, the platform is designed to be compatible with such mechanisms where users choose to adopt them.

### *D. AI-Related Risks*

AI-based moderation and metadata generation are subject to false negatives, where harmful content is not detected, and false positives, where benign content is flagged. There is also a risk of bias in model responses, particularly in moderation outcomes. The platform treats AI as an advisory component rather than a source of final decisions, and provides avenues for manual review. Future work may incorporate ensemble approaches or more structured human-in-the-loop processes.

## XI. Legal and Ethical Considerations

### *A. Copyright and Token Ownership*

Owning an on-chain token that references an asset does not in itself guarantee copyright ownership or license rights, unless those relationships are explicitly encoded and recognized in legal frameworks. The platform treats tokens as programmable certificates that can represent a variety of legal relationships, ranging from simple display rights to more extensive commer- cial rights and fractionalized royalties. Clear communication of what rights are actually granted is essential, and coordination with legal experts is necessary for robust mappings between smart contract semantics and statutory law.

### *B. Jurisdictional Ambiguity*

Smart contracts operate in a global context, while legal jurisdiction remains territorial and diverse. Licenses expressed as smart contract structures must map onto jurisdictionally recognized categories of rights and obligations, which may vary across regions. The platform acknowledges this limitation and relies on license templates and documentation that can be interpreted in parallel with on-chain state.

### *C. Ethical Use of AI*

The use of AI for moderation and metadata generation raises ethical questions about bias, transparency, and control. Models trained on large corpora may inadvertently reproduce problematic patterns or favor certain kinds of content. To mitigate risks, the platform keeps users informed that metadata and moderation assistance are AI-generated, allows creators to override suggestions, and emphasizes that AI outputs should be reviewed rather than accepted blindly.

## XII. Limitations and Future Work

### *A. Scalability and Costs*

The prototype operates on a test network and does not fully address the costs and performance implications of large- scale deployment on public mainnets or rollups. Gas fees, congestion, and storage pricing would influence the feasibility of frequent minting and complex licensing operations. Future work includes integrating cost models, experimenting with batch operations, and exploring layer-2 solutions or alternative execution environments.

### *B. User Experience*

Although the interface abstracts many technical details, concepts such as wallets, keys, and signatures may remain unfamiliar or intimidating to some users. More extensive usability studies, tutorials, and educational materials would be necessary to support broader adoption. The impact of AI assistance on user satisfaction and error rates also warrants systematic investigation.

### *C. Standardization of IP Semantics*

Expressing complex legal agreements in simplified smart contract structures entails a risk of oversimplification. Devel- oping robust, reusable schemas for IP semantics will likely require collaboration with legal experts, standards bodies, and domain-specific organizations. Such schemas would need to balance expressiveness with computational tractability and interoperability across platforms.

### *D. Future Enhancements*

Planned and potential enhancements include:

- support for additional blockchains and cross-chain prove- nance, including bridging IP records between networks;
- richer analytics dashboards for creators, including royalty breakdowns and derivative graphs;
- integration with decentralized identity systems to strengthen bindings between human identities and on- chain addresses;
- more sophisticated AI assistance for pricing recommen- dations, license selection, and risk analysis.

## XIII. Conclusion

This paper has presented a decentralized platform for digital asset ownership that combines ERC-721 tokens, IPFS-based content-addressed storage, Story Protocol licensing primitives, and AI-assisted workflows. By treating digital artifacts as programmable intellectual property rather than static files, the platform enables creators to register assets, configure licensing and royalties, and construct derivative relationships in a transparent and auditable manner.

A prototype implementation demonstrates that end-to-end registration is feasible with acceptable latency in a test environment, and that AI services can cover a substantial fraction of metadata generation tasks. While the work does not claim to solve all challenges in digital rights management or creator compensation, it provides an empirical case study of how decentralized storage, programmable ownership, and AI assistance can be integrated into a coherent system. Future research will need to explore scalability, user experience, legal alignment, and broader ecosystem integration to assess the long-term role of such architectures in digital asset ecosystems.

## XIV. Availability of Data and Materials

To foster transparency and ensure reproducibility of the results presented in this paper, the complete source code for the smart contracts, frontend interface, and backend services has been made open-source. The live prototype is currently deployed and accessible for demonstration purposes.

- **Project Repository:** https://github.com/Adwaitbytes/ Programmable-IP
- **Live Prototype:** https://storymusic.vercel.app/

## References

[1] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 2008.

[2] G. Wood, “Ethereum: A Secure Decentralised Generalised Transaction Ledger,” Ethereum Yellow Paper, 2014.

[3] J. Benet, “IPFS - Content Addressed, Versioned, P2P File System,” 2014.

[4] B. Cohen, “Incentives Build Robustness in BitTorrent,” in *Proc. Work- shop on Economics of Peer-to-Peer Systems*, 2003.

[5] S. Gans, A. Goldfarb, and A. T. Scott, “Economic Design for Digital Royalty Systems,” MIT Press, 2022.

[6] Various Authors, “Empirical Analyses of NFT Metadata Availability,” preprint literature, 2022–2023.

[7] Story Protocol, “Protocol Documentation and IP Primitives,” technical documentation, accessed 2024.

[8] Multiple Authors, “AI-based Content Moderation in Online Platforms,” survey literature, 2021–2023.

[9] Frontiers Editorial Board, “Application of Blockchain Technology in Digital Music Copyright Management: A Case Study of the VNT Chain Platform,” *Frontiers in Blockchain*, 2024.

[10] Alchemy Developer Team, “Your Guide to ERC-1155: Comparing ERC- 721 and ERC-1155,” Alchemy Technical Developer Brief, 2024.

[11] Encoding Labs, “ERC-6551: Token-Bound Accounts,” Technical Re- search Essay, 2025.

[12] LeewayHertz, “All About Ricardian Contracts,” LeewayHertz Research Whitepaper, 2025.

[13] A. Gervais *et al.*, “Security and Performance Analysis of Proof-of-Work Blockchains,” in *Proc. ACM CCS,* 2016.

[14] L. Luu *et al.*, “Making Smart Contracts Smarter,” in *Proc. ACM CCS,* 2016.

[15] E. Heilman, A. Kendler, A. Zohar, and S. Gordon, “Eclipse Attacks on Bitcoin’s Peer-to-Peer Network,” in *Proc. USENIX Security,* 2015.