WHITE-LABEL SHOP FOR DIGITAL INTELLIGENT ASSISTANCE AND HUMAN-AI COLLABORATION IN MANUFACTURING

Title	D3.2 – Smart Contract Specification
Document Owners	ICCS
Contributors	ICCS
Dissemination	Sensitive
Date	19/12/2023
Version	0.9

Co-funded by the Horizon Europe programme
of the European Union under Grant Agreement
№ 101092176

VERSION HISTORY

Nr.	Date	Author (Organization)	Description
0.1	02/10/2023	Giorgos Lagos ICCS	Deliverable structure
0.2	11/10/2023	Giorgos Lagos ICCS	Initial Draft
0.3	20/10/2023	Fotis Paraskevopoulos (ICCS)	Input to general specification of the NFT contract
0.4	24/10/2023	Giorgos Lagos (ICCS)	Addition to general specification of the NFT contract. Outline the technical implementation of the contract
0.5	05/11/2023	Fotis Paraskevopoulos (ICCS)	Review technical implementation
0.6	11/11/2023	Giorgos Lagos (ICCS)	1st Draft of final deliverable
0.7	15/11/2023	Fotis Paraskevopoulos (ICCS)	Review of 1st Draft
0.8	18/12/2023	Alessandro Canepa Lagos (IDEAL)	Peer review of 1st Draft
0.9	19/12/2023	Giorgos Lagos(ICCS)	Final version

DISCLAIMER

ACKNOWLEDGEMENT

This document is a deliverable of the WASABI project. This project has received funding from the European Union's Horizon Europe programme under grant agreement N° 101092176

CONTENT

FIGURES

TABLES

1. EXECUTIVE SUMMARY

Objective

The primary goal of Deliverable 3.2, titled “Smart contract specification,” is to design and implement a software component, named the "Royalty Distribution Module" (RDM), for effective management of developers' royalties. This module will be a part of a decentralized application on the Ethereum blockchain and will comply with the ERC-721 standard and possibly with the ERC-2981 standard as well. These standards are crucial for transforming traditional marketplace items (web2) into Non-Fungible Tokens (NFTs, web3) and managing royalties transparently and effectively across the network.

Integration and User Interaction

The RDM will be an integral component of the PrestaShop WASABI White Label Shop, functioning as a standalone unit for seamless interaction between marketplace users and the smart contract. The module will primarily serve two types of users: the Instance Administrator and the Skill Developer. The Instance Manager, as the contract owner, will oversee the distribution and monitoring of royalty payments. Simultaneously, the Skill Creator (Author) will have exclusive rights to their royalty earnings, with capabilities to manage and monitor distributed payments on the blockchain network.

Key Features

RDM will introduce several features:

Tokenization and Royalty Management: This includes the conversion of skills into unique NFTs representing transferable digital assets and a system for real-time royalty calculation and distribution.

Ownership and Transactions: Ensures secure and transparent ownership transfer through blockchain transactions.

Outcomes and Innovation

The decentralized framework of RDM aims to revolutionize the compensation model for open-source developers. This innovation is not only expected to promote more fair royalty distribution but also to encourage further innovation in the field. By leveraging blockchain technology, RDM provides a transparent, secure, and efficient mechanism for skill tokenization and royalty management.

Conclusion and Recommendations

In conclusion, the RDM module stands as a significant advancement in the realm of digital asset management and developer compensation. It is recommended to continually evaluate and update the module to keep pace with evolving blockchain technologies and marketplace demands. Further research and development may be directed towards enhancing user interface and transactional efficiency to maximize the benefits of this innovative approach.

2. INTRODUCTION

The goal of Task 3.2 “NFT-based royalty management for skill developers” is to create a Smart Contract on the Ethereum network that will allow the conversion of conventional marketplace products, into Non-Fungible Tokens (NFT), allowing the fractionization of the contributions of kill developers in the form or accrued royalties. The contract will generate unique, transferable digital assets for each skill, capturing royalties in an immutable manner adhering to the ERC-2891 specification. Royalties will be accrued to the developer each time their skill is used (via an eshop checkout / purchase), either directly or as a dependency in another skill. The module, called the Royalty Distribution Module (RDM), aims to manage these royalties and provide a seamless gateway to bind products in the Web2 realm, through a PrestaShop marketplace, to web3.

RDM has two main users: the Instance Administrator, who owns the Smart Contract, and the Skill Developer, who will receive the royalties. Each entity will have unique contract functions according to their needs. Finally, this Smart Contract will be incorporated into a PrestaShop module, allowing easy interaction within the shop environment.

Purpose and scope of the deliverable

The purpose of the work reported here is to develop a Smart Contract aligned with the ERC721 specification to tokenize open-source developers' skills as Non-Fungible Tokens (NFTs). Named the "Royalty Distribution module," this system aims to offer a transparent, secure, and automated way to capture royalties for the usage of these skills.

(a) Scope

- Skill Tokenization: The contract will mint unique digital assets or NFTs for each skill or set of skills provided by developers.
- Royalty Management: The NFTs minted will serve as vehicles to capture and distribute royalties to the respective developers upon the usage of a skill.
- Ownership and Transactions: Enables secure and transparent transactions, through interaction with the smart contract on the Blockchain.
- Administration and User Interaction: Two types of entities will interact with the smart contract—Instance Administrators and Skill Developers (authors), each with unique functionalities.

(b) Relation to other WPs and tasks

This task is designed to integrate with the outcomes of Task 3.1 to achieve the following:

1. Skill Upload and Provisioning: Provides an approach to uploading skills and minting them as NFTs.
2. Royalty Computation: Uses dependency discovery methods to accurately compute and allocate royalties.
3. Ownership Transfer: Facilitates easy ownership transfers through Blockchain transactions.

The Royalty Distribution Module (RDM) will serve as a critical component of a larger system, offering functionality and the ability to scale and adapt to other Work Packages (WPs).

3. STRUCTURE OF THE DELIVERABLE

(c) Components

- Web3 Transition: Enables the shift of web2 products into the decentralized web3 domain by issuing NFTs with unique identification codes and metadata.
- Decentralized Application (Smart Contract): Deployed on the Ethereum network, offering persistence and enhanced security.
- PrestaShop Integration: The smart contract functionalities will be accessible via a specialized PrestaShop module, making it user-friendly for both Instance Administrators and Skill Developers.

(d) User Roles

- Instance Administrator: Will act as the owner of the Smart Contract, having special administrative privileges.
- Skill Creators or Developers: They will be registered as eligible developers on-chain, earning royalties. Plus, they can interact with the contract to manage their royalty rights.

By structuring the deliverable in this manner, it will enable easy navigation and understanding.

4. BLOCKCHAIN AND THE ETHEREUM NETWORK

This section offers a fundamental overview of key technologies shaping royalty management of digital assets in the online systems landscape. It begins by examining Blockchain Technology, the foundational architecture for numerous online platforms. This is followed by a look at the Ethereum Network, notable for its capacity to execute smart contracts. The section also highlights EBSI, a European initiative focused on standardizing Blockchain services. It then clarifies the differences between coins and tokens, two forms of digital currency that are often confused. An introduction to Non-Fungible Tokens (NFTs), unique digital assets, is also provided. To conclude, the section discusses the integration of Blockchain Technology and Smart Contracts in WASABI, elaborating on its significance.

Blockchain Technology

Blockchain technology, often simply referred to as "Blockchain," was implemented as a groundbreaking force in the late 2000s, initially conceptualized to back Bitcoin. However, its applications have since expanded, now spanning numerous sectors beyond just finance.

At its essence, a Blockchain is a decentralized distributed ledger that exists across various locations or participants. This decentralization ensures no single entity has overriding control, contrasting with traditional centralized databases. [1]

Key Components of a Blockchain

Blocks: Digital 'containers' that house data, capturing transactions and events at specific moments.

Chain: Blocks are chronologically linked. Each one carries a cryptographic hash of its predecessor, ensuring a sequential and interconnected structure. [2]
Nodes: Computers in the Blockchain network validating and relaying transactions, upholding data integrity and consensus. [3]
Consensus Mechanisms: Protocols used to gain network agreement on transaction validity. Examples include Proof-of-Work (PoW) and Proof-of-Stake (PoS). [2]

Fundamental Principles

Decentralization: The distributed nature of Blockchain guards against systemic failures. [4]
Immutability: Information on a Blockchain remains unchangeable, vouching for the authenticity of records. [3]

Table 1 below outlines the key advantages and disadvantages of Blockchain technology.

Advantages	Challenges
Security: Cryptographic methods secure data, making unauthorized alterations challenging. [5]	Scalability: Some Blockchains, especially those employing PoW, can struggle with swiftly processing vast data amounts. [2]
The omission of middlemen can lead to swifter, cost-effective transactions. [2]	Regulation and Control: With various governments still discerning how to approach Blockchain and associated cryptocurrencies, a fluctuating landscape emerges for users and enterprises. [6]
Particularly in supply chains, a clear lineage of products is established. [2]	

Table 1: Advantages and disadvantages of Blockchain technology.

Ethereum Network

Ethereum, in the expansive world of Blockchain, is distinct not just as a digital currency but more as a comprehensive platform for decentralized applications (DApps). Initiated by Vitalik Buterin in 2013, Ethereum has grown into an invaluable asset for developers and enterprises. [1]

Ethereum is an open-source platform on Blockchain that allows developers to create and deploy DApps. Its native cryptocurrency is Ether (ETH), but Ethereum goes beyond just being a digital currency. [1]

Key Components of the Ethereum Network

Ether (ETH): Ethereum's native cryptocurrency, primarily utilized for transaction fees and computational services. [7]
Smart Contracts: Self-executing contracts whose terms are directly encoded into lines of code. They ensure credible transactions without intermediaries. [7]
Ethereum Virtual Machine (EVM): A runtime compiler capable to facilitate smart contracts on the Ethereum platform. [7]
Gas: Denotes the computational expense of operations, crucial for transactions or running DApps. [1]

Distinctive Features

Decentralization: Ethereum runs on a decentralized cluster of computers. [7]
Permissionless: Open to all, Ethereum allows anyone to participate, manage accounts, or develop DApps. [7]
Turing Complete: Ethereum's computational prowess is vast, capable of handling any computation given ample resources. [1]

Advantages	Challenges
DApps: Beyond just finance tools, Ethereum's DApps range from gaming applications to extensive financial tools.	Scalability: Historically, Ethereum's reliance on Proof-of-Work made it somewhat sluggish, causing occasional congestion and elevated transaction fees. However, the transition to Ethereum 2.0 and its Proof-of-Stake mechanism seeks to mitigate this.
Flexibility: Ethereum's design emphasizes adaptability. Its smart contracts can support a wide spectrum of applications.	Complexity: Creating smart contracts requires proficiency in the Solidity language, where errors can be significantly expensive.
Decentralized Finance (DeFi): Ethereum's platform has become foundational for the DeFi domain, which includes decentralized lending, borrowing, and beyond.	Interoperability: Ethereum's vastness still requires bridging solutions for seamless interactions with other Blockchains, which can be complex.

Table 2: Advantages and disadvantages of Ethereum

European Blockchain Services Infrastructure (EBSI)

The European Blockchain Services Infrastructure (EBSI) represents a significant effort by the European Union (EU) to create a pan-European Blockchain platform. Its primary aim is the provision of cross-border digital public services, ensuring superior security and privacy standards. [8]

Background and Origins

Prompted by the global surge in Blockchain adoption, the European Union recognized the potential impact of this technology on public services, innovation, and transparency. EBSI is a testament to this recognition, crafted through the joint efforts of the European Commission and the European Blockchain Partnership, encompassing 29 European nations.

Key Objectives of EBSI

Interoperability: A uniform infrastructure promoting smooth operations across member states, integrating diverse Blockchain systems.
Trust and Security: Tapping into Blockchain's security attributes, EBSI strives for trustworthiness in digital trails, emphasizing data authenticity and fraud reduction.
Digital Sovereignty: Upholding European principles and regulatory frameworks, ensuring digital offerings align with European standards, particularly in data protection and privacy. [8]

Key Services and Use Cases

EBSI stands as a platform addressing a wide variety of sectors:

Notarization: Employing Blockchain's unchangeability for efficient and reliable digital notary services.
Diplomas: Guaranteeing the genuineness and verifiability of educational qualifications throughout the EU.
Identity: A robust and interoperable eID mechanism, assisting EU residents in establishing their identity, enhancing trust in public offerings.
Audit Documents: Offering clear, unalterable audit trails for official documentation.
Customs and Taxation: Streamlining customs and tax procedures by leveraging the transparency and traceability inherent in Blockchain. [8]

Coins vs Tokens

In the realm of cryptocurrency, "coin" and "token" often get used interchangeably by many. However, understanding the distinction is important for both developers and investors.

Coins: Independent digital currencies built on their own Blockchains, considered as digital money. Examples include Bitcoin (BTC), Ethereum (ETH), and Litecoin (LTC). [9]

Tokens: Digital assets constructed and maintained on pre-existing Blockchains via smart contracts. Ranging from value units to tangible assets, they commonly originate from Initial Coin Offerings (ICO) or Token Generation Events (TGE). [5]

Key Differences

Infrastructure:
Coins: Function on their genesis Blockchain. For example, Bitcoin relies on the Bitcoin Blockchain.
Tokens: Crafted atop existing Blockchains, such as Ethereum. ERC-20 tokens on Ethereum exemplify this.

Utility and Purpose:
Coins: Mainly serve as exchange mediums, value reserves, or accounting units.
Tokens: Offer diverse utilities, from representing tangible assets like property to project stakes, voting rights, or even granting specific project access. [2]

Origination:
Coins: Generated via mining (in Proof-of-Work systems) or staking (for Proof-of-Stake systems).
Tokens: Usually generated from ICOs or TGEs, where they're exchanged for well-known cryptocurrencies, mostly Ethereum. [5]

Valuation:
Coins: Value arises from elements like demand, supply, utility, and general market consensus.
Tokens: Their worth is often connected to the utility they offer, the potential of their associated projects, or the tangible assets they symbolize. Some might even be valueless.

Non-Fungible Tokens

NFTs, standing as a breakthrough in digital ownership, represent the long-standing digital achievement of tangibility and ownership. Recently gaining prominence in sectors like art, collectibles, and real estate, they offer verifiable digital asset authenticity and ownership rights.

Distinct from typical cryptocurrencies, a Non-Fungible Token (NFT) is a unique digital certificate on a Blockchain guarantying ownership of a singular item or content. In contrast to fungible assets like Bitcoin, NFTs are distinct and cannot be swapped on an equal basis. [9]

Key Components of an NFT

Uniqueness: Each NFT has unique attributes, ensuring its authenticity. [10]

Indivisibility: NFTs stand whole: Therefore, one can cannot purchase them in fractions. [10]

Provenance: Blockchain traceability guarantees the NFT's authenticity. [10]

Interoperability: NFTs have the flexibility of use across multiple applications and platforms. [10]

Core Principles

Digital Ownership: NFTs challenge the traditional licensing model, offering genuine digital item ownership. [10]

Authenticity and Rarity: With Blockchain's assistance, NFTs guarantee authenticity, and their metadata can guarantee for their rarity. [10]

Smart Contracts: Embedded within NFTs, these contracts can confirm usage rights or secure creator royalties upon resales. [10]

Benefits

Digital Art Monetization: Artists can trade their digital creations as NFTs, affirming their rights and establishing origin.

Gaming and Collectibles: Gamers can now trade in-game assets as NFTs, adding tangible value to their virtual engagements.

Royalties: Through NFTs, creators can ensure continuous compensation from subsequent sales by embedding royalties.

Decentralization: Operating within decentralized spaces, NFTs can often bypass censorship, offering creators increased autonomy.

Addressing Privacy Preservation and Privacy Concerns with NFTs

In the modern digital era, there's a growing realization of the importance of protecting identities and privacy online. Frequent data breaches and privacy scandals have undermined trust in the security of personal information. Blockchain technology, with its decentralized and immutable nature, offers a promising solution to enhance security and transparency.

NFTs hold potential in digitizing personal identities, offering individuals complete control over their personal data and its sharing. This approach mitigates dependence on centralized authorities and enhances defenses against data misuse. NFTs enable users to interact online using pseudonyms, providing verified credentials without revealing sensitive personal information. This pseudonymity, along with blockchain's inherent security, fosters an environment conducive to privacy.

While NFTs alone don't inherently ensure privacy, they enhance it through blockchain's features like immutability, transparency, and decentralized control. NFTs enable pseudonymous interactions, selective data sharing, and self-sovereign identity, where individuals own and control their digital identities.

In the context of royalty distribution, the level of privacy required by NFTs is not as extreme. Pseudonymity suffices in this application, as the primary focus is on authenticating ownership and tracking the distribution of royalties rather than protecting deeply personal information. This use case benefits from the transparency and traceability of blockchain technology, ensuring fair and verifiable distribution of royalties without necessitating high-level privacy protections typically associated with personal data.

Integrating Blockchain Technology and Smart Contracts in WASABI: The Rationale

The decision to integrate Blockchain technology into the WASABI project, that will be integrated as shown in Figure 1, aims to enhance both transparency and efficiency in the way authors interact with the system.

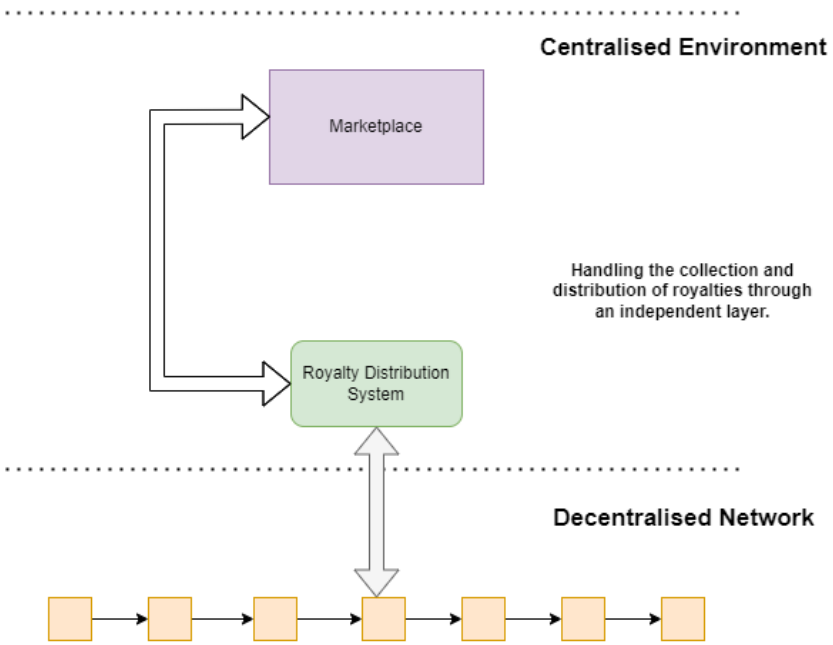

Figure 1 – Royalties will be processed in a decentralized Network.

Transparency and Trust in Donations

In the WASABI marketplace, authors are entitled to receive donations upon the sale of their products. Utilizing Blockchain technology to handle these transactions is a strategic choice that aligns perfectly with the need for transparency. On a Blockchain, all transactions are recorded on a public ledger that is immutable, meaning that once data is entered, it can neither be altered nor deleted. This creates an environment of trust, as authors can be confident that their donations are being handled in a fair and transparent manner. They can verify each transaction themselves, reducing the likelihood of disputes over donation amounts or their distribution.

Real-Time Access to Earnings

Blockchain technology not only provides a high level of transparency but also enables real-time tracking of transactions. Authors can, at any moment, check their earnings in a transparent manner without having to go through bureaucratic hoops. This immediate access to financial data is empowering for authors, who no longer have to wait for periodic reports or go through third parties to understand their earnings. The Blockchain acts as an open book, readily available for examination, further increasing trust in the system.

System Integrity Over Traditional Platforms

When compared to traditional forms of product uploading and royalty distribution, Blockchain comes out as a more reliable system. Traditional systems often involve layers of intermediaries, each taking a cut from the author's earnings adding a layer of opacity to the transaction. Blockchain eliminates these intermediaries, ensuring that more of the revenue goes directly to the authors.

Decentralization and Empowerment

Additionally, the decentralized nature of Blockchain allows for a democratization of access and control. Authors are not just passive participants but can actively engage with the system, making choices about their products and earnings (royalty rights allocation) without the need for an intermediary or the risk of centralized control affecting their benefits.

In conclusion, integrating Blockchain technology into WASABI serves to amplify the core values of the platform: transparency, trust, and efficiency. This makes it a compelling alternative to traditional systems, drawing in authors who seek a more open, reliable, and equitable marketplace for their products.

5. SKILL UPLOAD, TOKENIZATION AND PROVISIONING AS AN NFT ON THE SELECTED BLOCKCHAIN NETWORK

As we delve into the era of Web3 and the evolution of digital capabilities, the intersection of traditional web2 products (skills in our case), their tokenization, and their representation on Blockchain networks becomes an interesting point for exploration.

Through this work we unpack the innovative processes of transitioning skills from a Web2-based Marketplace to the Web3 ecosystem, ensuring seamless integration and offering significant benefits for both developers and consumers.

We define how skills can be defined, via specific metadata, and subsequently, their representation within an Ethereum smart contract.

The primary emphasis is laid on the precision of skill encapsulation, on the algorithm capable of handling decentralized royalty management, and the overall functionality within the chosen Blockchain infrastructure.

Skill Packaging & Metadata Specification

In the Web2-based Marketplace, skills are uploaded and need to be seamlessly transitioned into the Web3 ecosystem. Within this Web3 environment, skills are documented in a specialized format within a Smart Contract. This Smart Contract operates autonomously, fulfilling its designated purpose, which will be elaborated upon subsequently. To successfully mint a Skill on the Blockchain, all parameters must be provided. These parameters are essential for implementing accurate royalty logic on-chain. Each Skill is attributed to one or more authors, and every author has a proportional contribution to the Skill. Additionally, some skills may depend on others, and this dependency information must also be provided when minting a Skill on-chain.

An appropriate file format for storing products (Skills) in the Web2 marketplace would be JSON. In this format, each product would encompass all the necessary information to facilitate the processes. Transitioning these Skills into the Web3 space would then become a straightforward task, as all relevant metadata would be readily available within the JSON file, as shown in Figure 2.

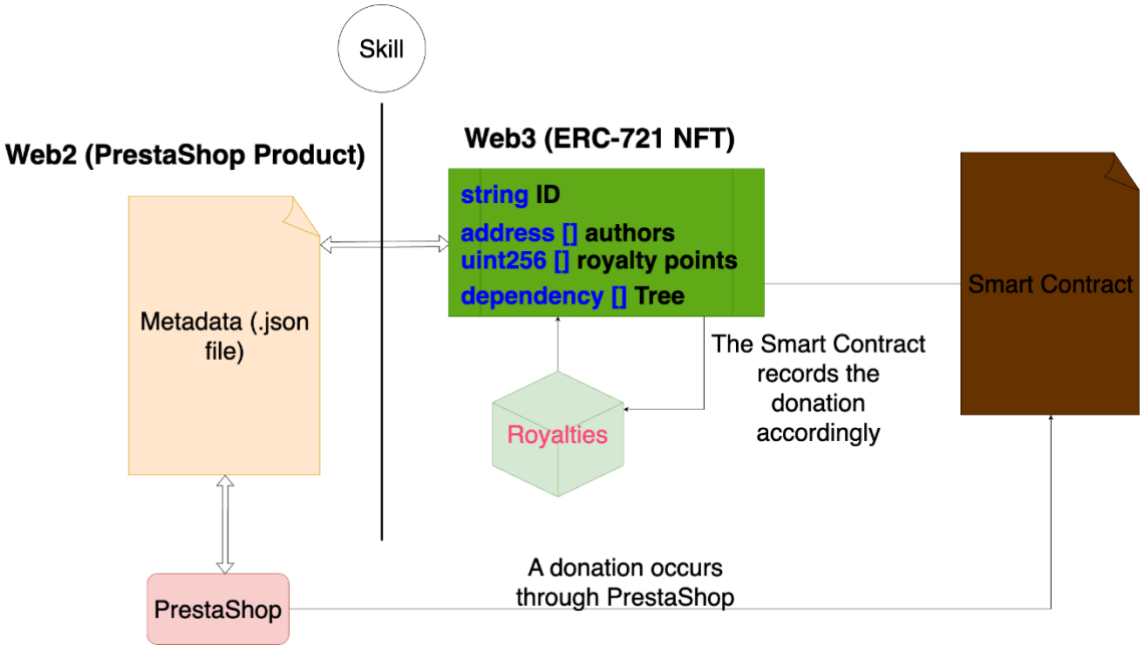

Figure 2 - Bridging products from Web2 to Web3.

Skills encapsulated in a JSON file would contain all the essential information needed for both the Web3 module and other components of the system. An example format that sufficiently supplies data for the decentralized royalty module is depicted in Figure 3.

```
{
    "name": "SkillA",
    "type": "package",
    "type_meta": {
        "mime_type": "application/x-zip"
    },
     “royalties":[
          {
          "type": “author",
          “id": “0x6227d76b055235e003e8F119DA6fFfB9c984187f”
          "contribution": 20
          },
          {
          "type": “author",
          “id": “0x07D7DBAE2a0203280c0783b0a2d1F2a8E09dC312”
          "contribution": 60
          },
          {
          "type": “skill",
          “id": “SkillB”
          "contribution": 20
          }
     ]
}
```

Figure 3 - Example format of a bridgeable to Blockchain product.

We take a closer look at the important details in the .json file shown above, explained in Table 3:

Key	Description
name	This is the name of the skill, as is considered a unique key across the marketplace. Through this key author can specify skill dependencies.
type	The type of skill based on the form into which the skill is delivered. Possible values are: - *package:* A binary package containing the either the executables of the source code needed to compile the and use the skill. - *service:* A service provided in the form of a SaaS endpoint - *source:* An SCM link (Git / SVN) to a particular commit, where the skill can be downloaded and used In the future other values can be integrated, such as *executable*, *PaaS*, etc.
type_meta	The contents of this key is specific to the *type* of skill, and it provides the Skill consumer with the necessary information in order to use the skill. The contents are arbitrary and the contract of the contents are determined by the Skill developers. An example of the *type_meta* key can be the compression used for a binary packge, or the access token for a SaaS service, etc.
royalties	The royalties field in the JSON structure is an array containing the details of royalty distributions associated with a particular digital asset or skill. Each item within this array is an object representing a specific beneficiary's entitlement or contribution to the royalties. Here's a breakdown of what each key represents within the royalties objects: type: This key specifies the role of the entity associated with the royalty. In the given example, two types are shown: "author" and "skill". "author" refers to individuals who have contributed to the creation or development of the digital asset or skill, whereas "skill" represents a dependency to an existing skill. id: This is a unique identifier for the entity described by the type. For "author" types, this is a blockchain address that uniquely identifies the author's account on the Ethereum network. This would be where the royalties will be sent. For the "skill" type, the ID is a string that references the dependent skill. contribution: This key represents the percentage or portion of the royalties that the specified entity is entitled to receive. The numbers are percentage points, summing up to 100 across all contributors within a single royalties array. This allows for an explicit declaration of how the total royalties should be divided among the various stakeholders.

Table 3: Further analysis of the skill packaging field that defines royalties.

Contract

The royalty management system operates as a decentralized application (dApp) built on the Ethereum Blockchain, specifically in the form of a smart contract named 'RoyaltiesSystem.' This smart contract is an ERC721 token, a standard commonly employed for Non-Fungible Tokens (NFTs). It is engineered to handle royalties for developers within a given ecosystem, enabling donations to be directed to developers' wallets. Furthermore, developers can always review their balances. The contract also provides features allowing the contract owner to mint new tokens (Skills) and temporarily suspend the contract's operations.

To enable this functionality, the contract incorporates two external libraries from OpenZeppelin: ERC721Enumerable and Ownable. ERC721Enumerable extends the basic ERC721 standard to make tokens enumerable, thus facilitating the enumeration of the total token supply and the examination of token balances. The Ownable library establishes contract ownership and introduces modifiers to certain functions, limiting their execution to the contract owner.

Within the smart contract, Skills are initially defined by the owner and subsequently stored in an array within the contract itself. To successfully create a Skill on-chain, the owner is required to invoke the `MintSkill()` function from the contract and supply all the necessary information.

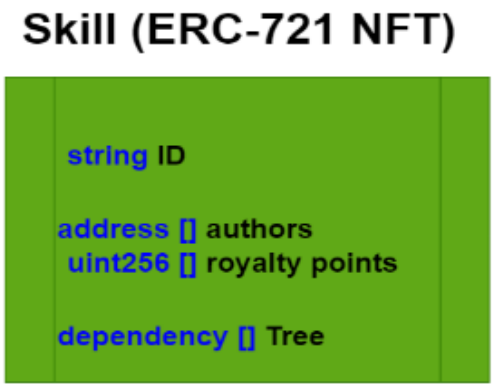

Figure 4: Definition of products within the Smart Contract.

As depicted in Figure 4, each Skill is characterized by several parameters. The 'ID' serves as the unique identifier for each Skill. The 'authors' array lists all developers who have contributed to the Skill and are thus entitled to claim royalties from donations made to that Skill. The 'royaltyPoints' array specifies the proportionate allocation each developer has in the Skill. Finally, the 'dependency' array may either be empty or include all the Skills upon which a particular Skill is dependent. The format for these dependencies, represented as a new struct, is 'Dependency: {ID, Allocation}'. In this context, a Dependency is considered valid if it includes both the ID of the dependent Skill and a percentage that is deducted and transferred to that dependent Skill whenever a donation is made. An example of a Dependency Tree for a particular skill can be illustrated as follows in Figure 5:

Dependency Tree

Skill ID	Distribution
0	20%
5	30%
...	...

Figure 5 - Example table showcasing the dependencies of a skill.

The smart contract employs a specific donation logic characterized by the following key elements:

- The received donation is logged for subsequent processing by both the Contract and Marketplace Owner.
- The total donation amount is proportionally distributed among all affiliated skill developers based on their percentage allocation.
- The royalty distribution system prioritizes a "dependency-first" approach. This means that upon logging a donation, the balances of dependent skills and their respective developers are increased first. Only then are the balances of the developers associated with the skill that directly received the donation incremented.

We will also consider the importance of complying with the ERC-2981 standard to ensure consistency across various marketplaces.

In such a case the key components of this integration will be as follows:

1. When a donation is made, the transaction will be recorded in a manner that aligns with the ERC-2981 standard.
2. The total sum of donations received will be allocated among skill developers in direct proportion to their designated royalty shares as outlined in the ERC-2981 standard

By following the ERC-2981 standard, the system guarantees a consistent and transparent method for royalty payments, ensuring creators are compensated fairly for their work, irrespective of where or how their skills are sold or utilized within the NFT marketplace.

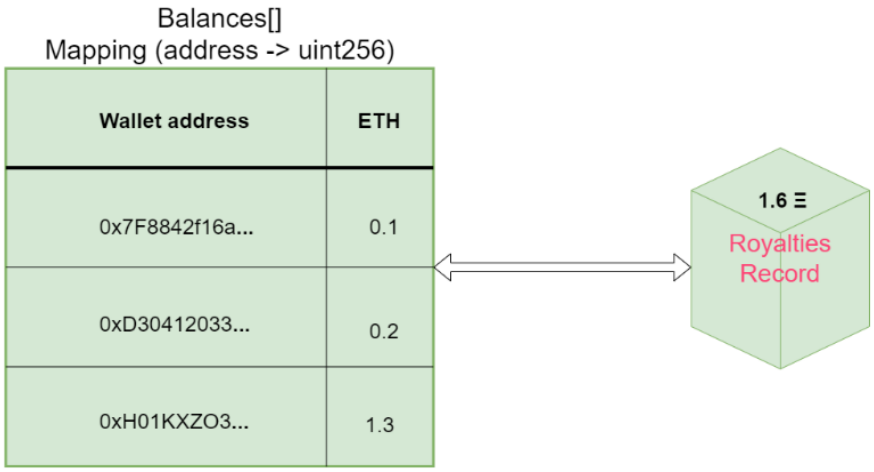

Figure 6 - An example of a potential author balances record.

Considering the components detailed above in Figure 6, a real-time snapshot of the contract would appear as depicted in Figure 7 below.

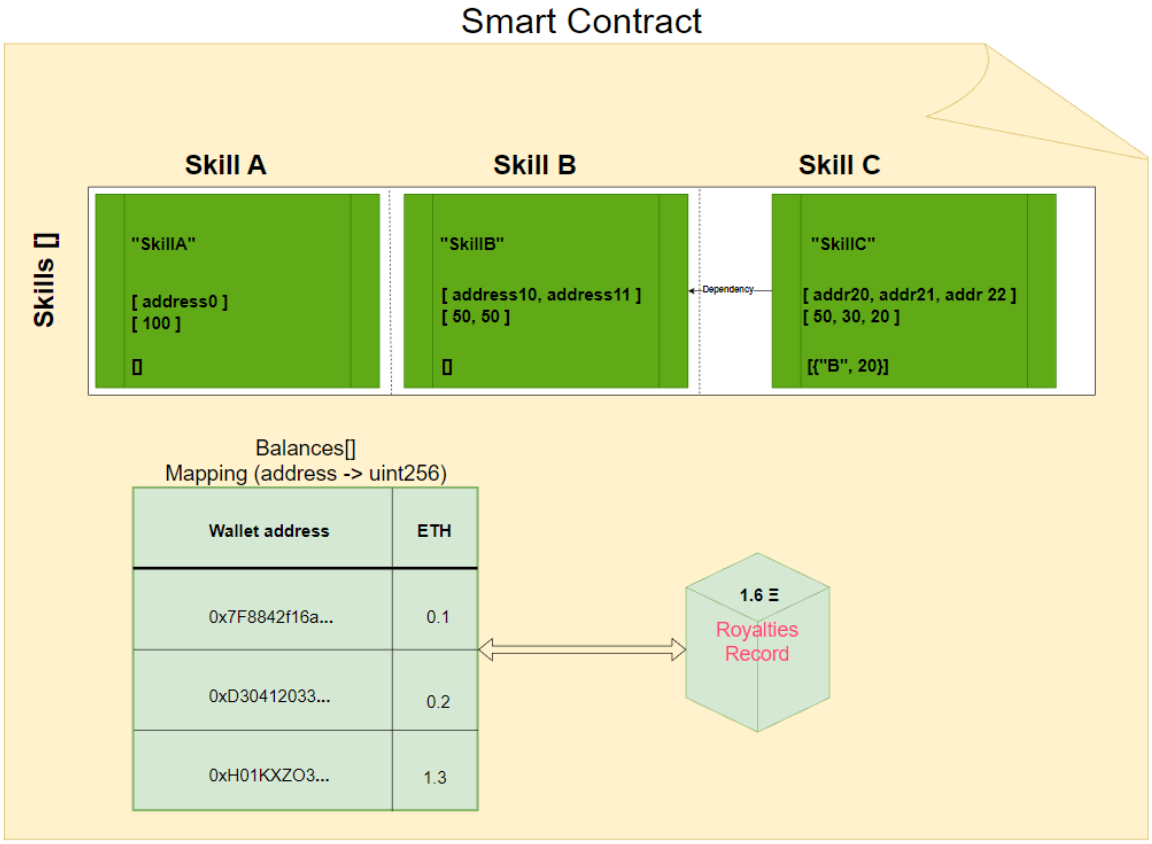

Figure 7 - An example of a possible instance of a smart contract during operation.

The outlined components sufficiently address the requirements for on-chain skill encapsulation, logging donations for later processing, and facilitating the royalty fund distribution at a time chosen by the marketplace

administrator. Additionally, while preserving its core functionality, the system can handle auxiliary requests. These requests include:

- Verify whether an Ethereum wallet address is associated with any skill and qualifies for royalties (True/False).
- Retrieve the total balance owed to an author (provided they are registered as royalty-eligible) (numeric value).
- Search for skills on-chain using a specific ID and retrieve all related skill data (only if registered on-chain).
- Transfer skill royalty rights to another entity (assuming you possess the royalty rights to begin with).

For all the secondary functionalities mentioned and of course the core functionalities a suitable dashboard will be created within PrestaShop for both the marketplace administrator to interact with the smart contract and the skill authors.

PrestaShop Integration

The core engine of the RDM system is the smart contract, which operates continuously on the Ethereum Blockchain. The contract's functions, however, are invoked through the PrestaShop module. Specifically, administrative tasks in the smart contract are accessed via the admin dashboard, while functions tailored for authors are triggered within their own dashboard. Additionally, certain functions are automatically activated through specific PrestaShop hooks during processes like product upload and checkout.

The above information is encapsulated in Figure 8 below.

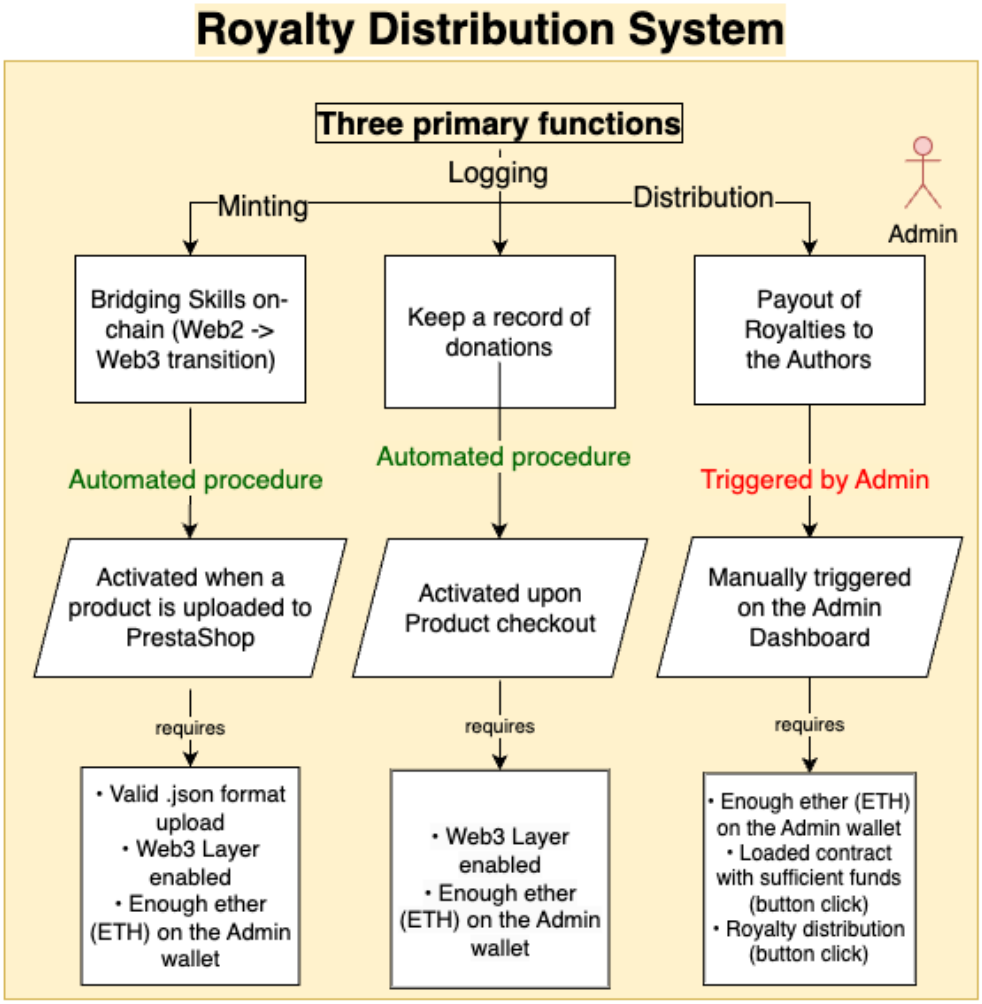

Figure 8 - An overview of the logic behind the PrestaShop integration.

(e) The Web3 Admin Dashboard

The PrestaShop Marketplace Administrator (hereafter referred to as "Admin") is tasked with supplying a mnemonic seed phrase for an Ethereum wallet. Once this mnemonic phrase is provided, the entire Web3 system will be activated.

This designated wallet will serve dual purposes: it will be employed to deploy the smart contract and facilitate interactions with this contract. Such interactions will occur automatically during product uploads and checkouts on PrestaShop while some will be manually initiated by the Admin.

An overview of the admin dashboard can be seen in Figure 9:

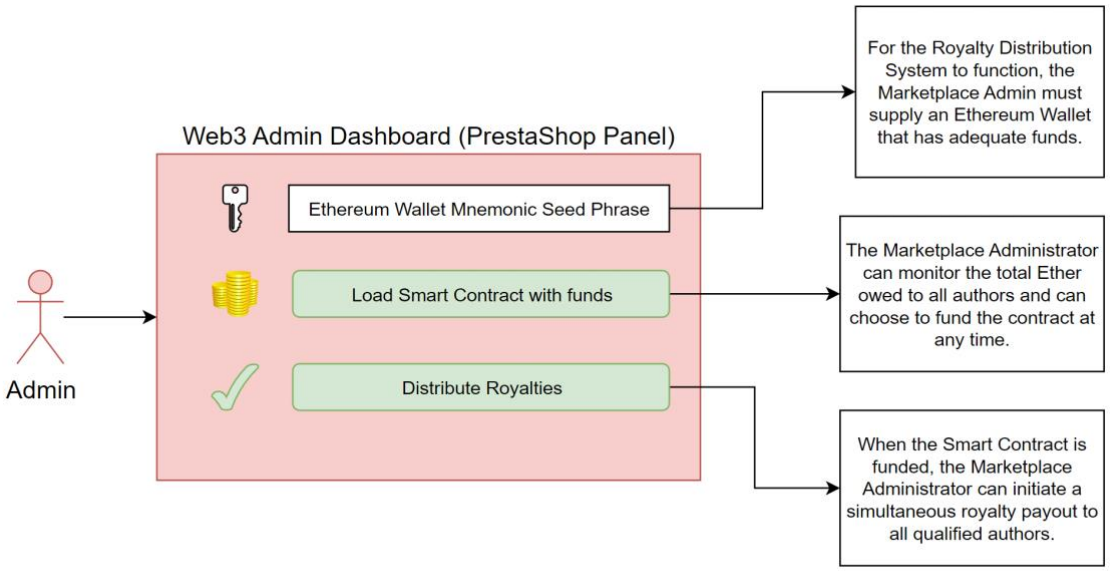

Figure 9 - An overview of the Administrator's dashboard.

(f) The Author Dashboard

Authors can link their Ethereum wallets within the dashboard to view all skills that have been successfully attributed to them on-chain. In addition, they can monitor their royalty balances, though withdrawal capabilities are not provided through this interface. They can also assign royalty points (and consequently upcoming royalties) from skills they own to other individuals.

An overview of the author dashboard can be seen in Figure 10:

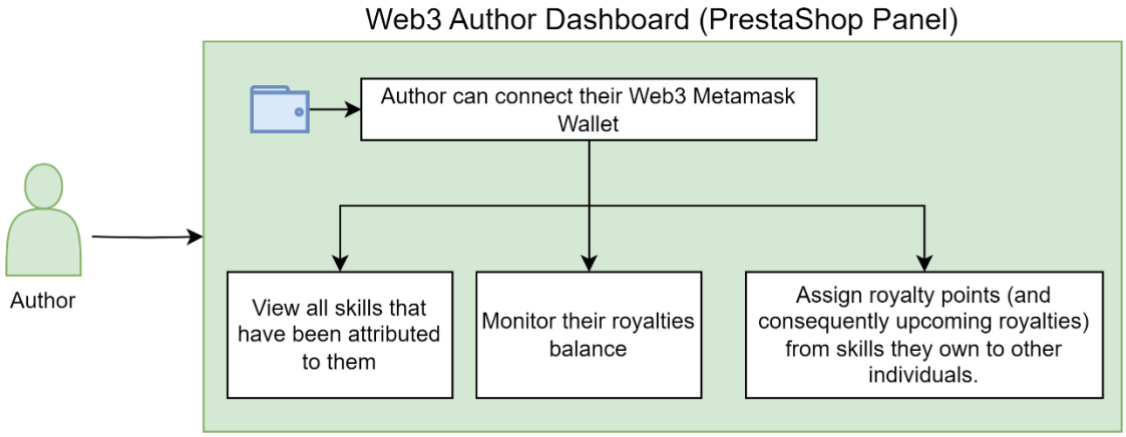

Figure 10 - An overview of the author's dashboard.

(g) Automated minting of skills upon upload on PrestaShop

Once a product is successfully uploaded to PrestaShop in the form of a valid .json file, an automated process will initiate to mint this product (Skill) on-chain. This automation is facilitated using the relevant PrestaShop hook. This action triggers an on-chain transaction, which incurs Ethereum gas fees, to the smart contract using the admin wallet specified in the PrestaShop Administrator panel.

(h) Automated recording of donations upon checkout on PrestaShop

Every time a product checkout happens, an automated process is triggered to maintain an on-chain record of the royalties. These royalties will be allocated later to the authors of the skills (or products) that were part of the checkout.
The royalty amount is predetermined and stands at 5% of the total product price.

6. ROYALTY COMPUTATION THROUGH DEPENDENCY DISCOVERY

To accurately compute the royalties that skill authors are entitled to, accounting for potential skill dependencies, the system requires comprehensive information. This ensures that the algorithm can effectively process and analyze the data on-chain.

Due to the supported skill upload format (.json), all dependency-related information has been seamlessly transitioned from the web2 to the web3 domain, enabling immediate discovery.

Given that all dependencies for each skill are identified, a "first-process-dependencies" approach for royalty donations has been implemented. This implies that whenever a donation is logged on-chain, the system initially determines the portion to be deducted from the total donation and allocated to the dependencies.

After addressing the royalty allocations for all dependent skills, the remaining donated amount is divided among the skill's authors based on their recorded contributions within the smart contract, verifiable on-chain. It's important to note that the actual distribution does not occur at this stage. Instead, this phase involves logging the donated royalties and updating the balances of the respective authors.

A summary of the donation logic employed by the system is outlined below in Figure 11:

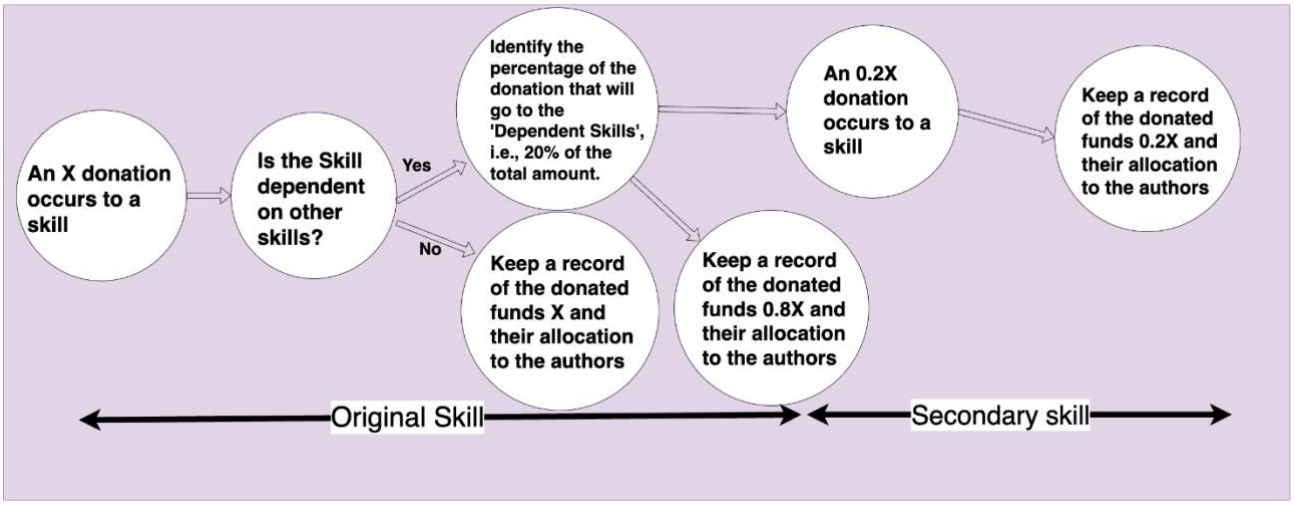

Figure 11 - The algorithm behind the royalty allocation to skills with dependencies.

Figure 12 below presents a concrete example with numerical data following the algorithm outlined in Figure 11.

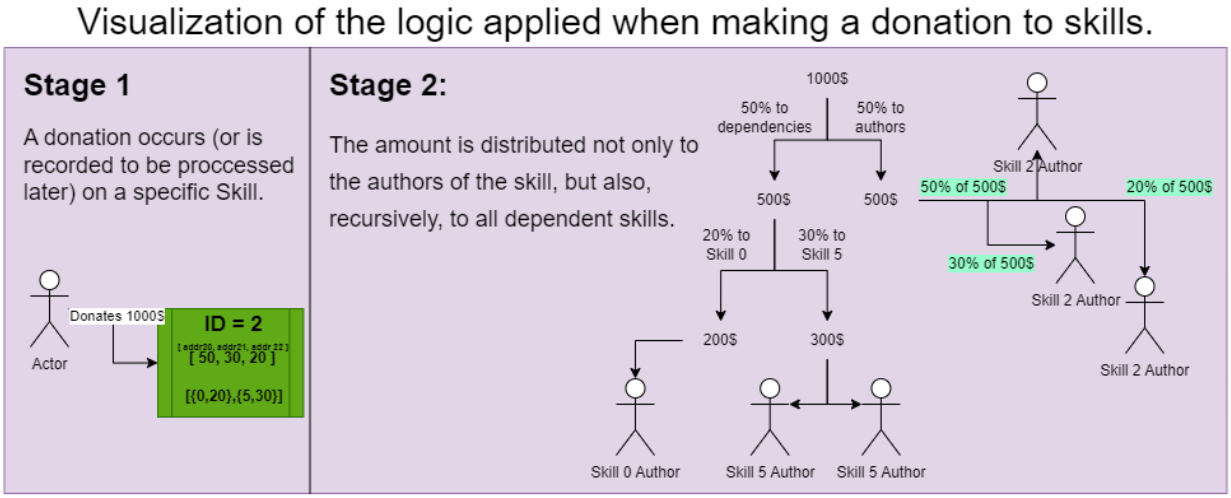

Figure 12 - A numerical example of a donation happening to a skill with dependencies.

7. OWNERSHIP TRANSFER THROUGH BLOCKCHAIN TRANSACTIONS

Every skill author is permitted to transfer either a portion or the entirety of their royalty rights (the rights entitling them to receive skill royalties) through Blockchain transactions.
In on-chain terms, these royalty rights are represented as "Royalty Points" and are embedded within Skills as a component of their structural definition, as shown in Figure 13. Solely the designated author of a skill, who inherently possesses royalty points for that specific skill, can apportion a segment of their points to another party.

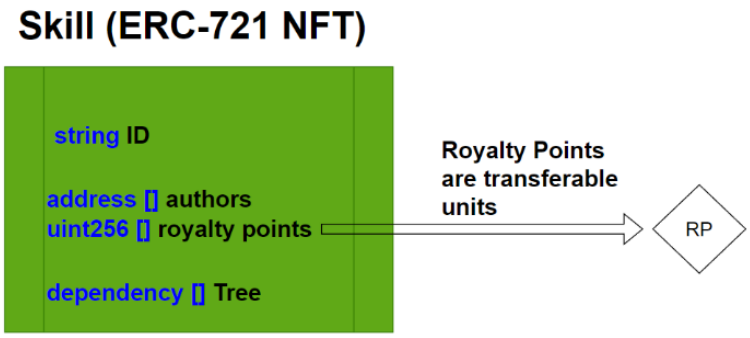

Figure 13 - Defining royalty allocation on the Blockchain.

Consequently, one might view royalty points as on-chain transferable tokens that confirm royalty entitlements for a particular skill, as shown in Figure 14.

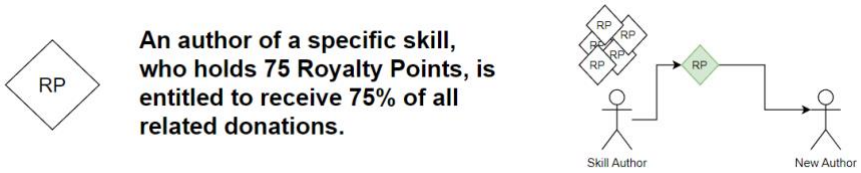

Figure 14 - The mechanism allowing royalty transfer from authors.

As previously stated, an author can allocate a portion (or all) of their royalty points to another entity. By design, this must be a legitimate Ethereum wallet address, given that the entire process unfolds on-chain. An author also has the option to completely relinquish their royalty points. This action can be performed through the Author dashboard, where they can connect their wallet and specify the quantity of royalty points, they wish to transfer and to which recipient.

In scenario A, as shown in Figure 15, imagine an author entirely relinquishing their royalty rights. In this situation, these rights are evenly distributed among the other eligible skill authors. If the renouncing author is the sole contributor to that skill, then the royalty rights automatically transfer to the Marketplace Administrator and the designated wallet they've supplied (or, in Blockchain terms, the smart contract owner).

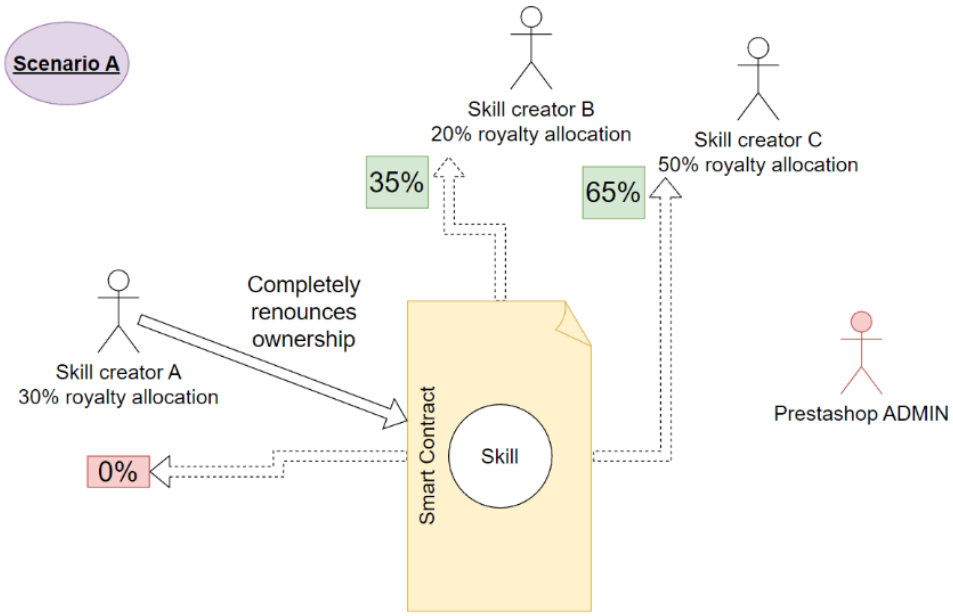

Figure 15 - Showcasing an example of an author totally renouncing their royalty allocation.

In scenario B, as shown in Figure 16, an author chooses to transfer a portion (or the entirety) of their royalty points to a different wallet address. They have the liberty to execute this transfer, and the smart contract doesn't restrict or filter any addresses, if they are legitimate Ethereum wallet addresses.

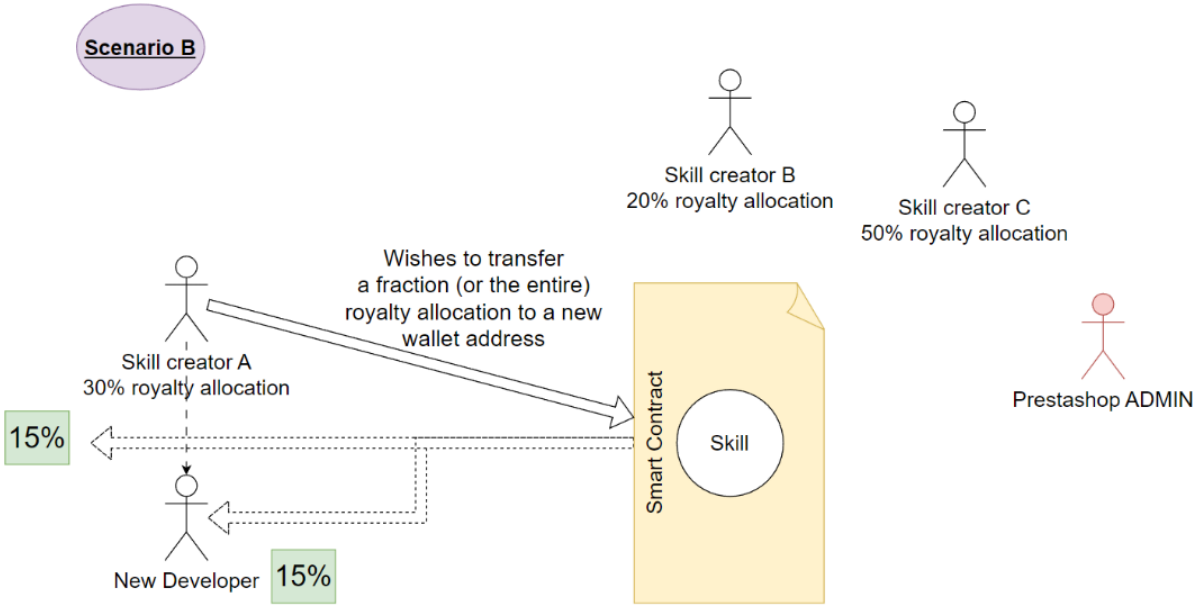

Figure 16 - Showcasing an example of an author transferring royalty allocation to a new author.

8. IMPLEMENTATION

An overview of the system, detailing exactly how, by whom, and when the smart contract will be invoked is shown in Figure 17.

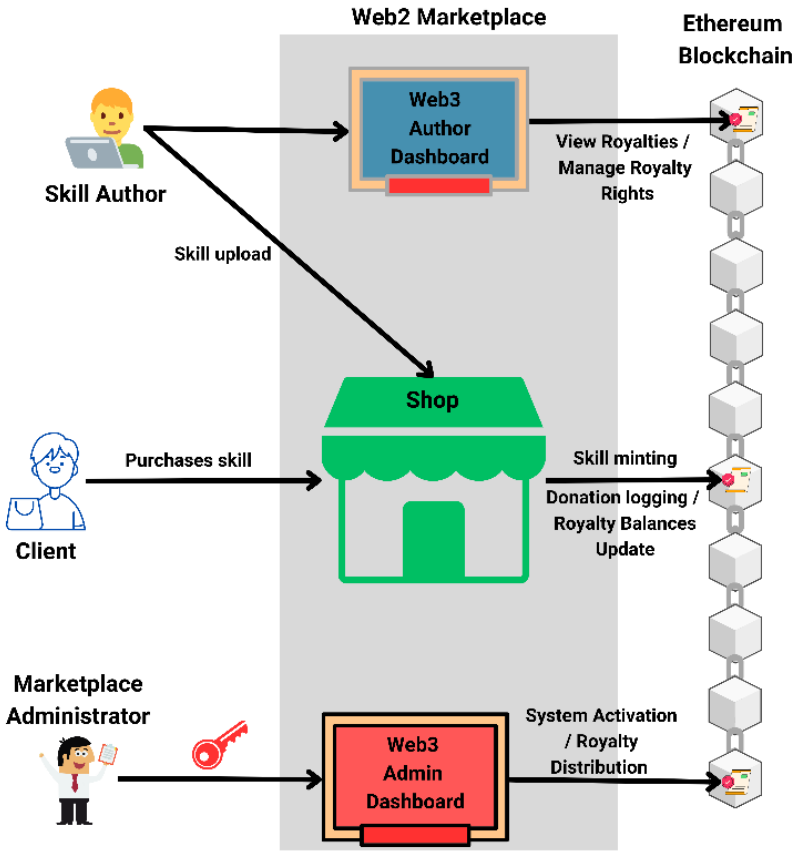

Figure 17 - Showcasing the interaction with the system by all involved parties.

The UML diagram provided below in Figure 18 outlines all the Smart Contract functions and endpoints. These are intended for use by:
A) The contract itself (for secondary functions)
B) The system administrator
C) Skill authors via their dashboard.

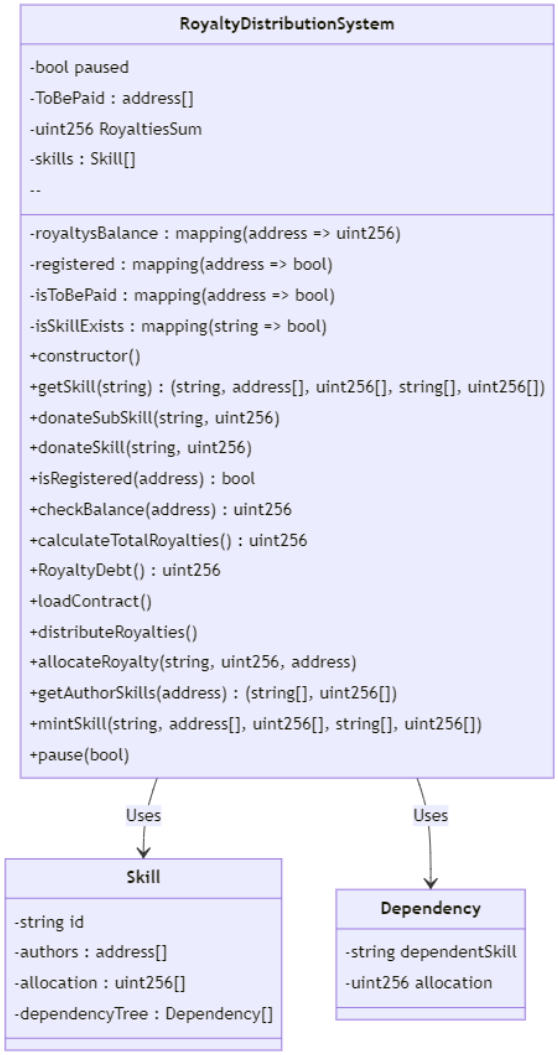

Figure 18 - An approach of the Smart Contract as a UML Class diagram

The UML class diagram shows the structure of the RoyaltyDistributionSystem and its associated classes, Skill and Dependency. Each class encapsulates specific attributes and operations associated to the royalty distribution mechanism in a Blockchain-based environment.

RoyaltyDistributionSystem Class: This class serves as the primary orchestrator for the royalty distribution mechanism.

Attributes:
paused: A boolean flag indicating the operational state of the system.
royaltysBalance: A mapping that correlates an address to its respective royalty balance.
registered: A mapping to verify if an address is officially registered within the system.
ToBePaid: An array storing addresses queued for royalty payment.
isToBePaid: A mapping to ascertain if an address is marked for payment.
RoyaltiesSum: A counter maintaining the cumulative sum of royalties.
isSkillExists: A mapping to validate the existence of a skill based on its unique identifier.
skills: An array storing instances of the Skill class.

Note: The names mentioned are provisional and may be subject to change for alignment with the standard ERC-2981 guidelines, if compliance is necessary.

Operations:
Methods like constructor(), getSkill(), and donateSubSkill() facilitate the initialization, retrieval, and donation processes respectively. Other methods like isRegistered(), checkBalance(), and calculateTotalRoyalties() provide utility functions to check registration status, balance, and total royalties. Administrative functions such as loadContract(), distributeRoyalties(), and pause() enable contract management.

Dependency Class: This class encapsulates the dependencies that a particular skill might have on other skills.

Attributes:
dependentSkill: A string denoting the skill on which there's a dependency.
allocation: A numerical representation (uint256) indicating the percentage allocation of the dependency.

Skill Class: Represents individual skills within the system.

Attributes:
id: A unique string identifier for the skill.
authors: An array of addresses representing the creators or contributors of the skill.
allocation: An array indicating the percentage allocation for each author.
dependencyTree: An array of Dependency objects, illustrating the skill's dependencies on other skills.

Relationships:
The RoyaltyDistributionSystem class has a unidirectional association with the Skill class, denoted by the "Uses" relationship. This indicates that the RoyaltyDistributionSystem class utilizes the Skill class for its operations but not vice versa.

Similarly, the RoyaltyDistributionSystem class also has a unidirectional association with the Dependency class, again denoted by the "Uses" relationship. This signifies the system's reliance on the Dependency class to manage and understand skill dependencies.

The final version of the system will consist of multiple independent royalty distribution platforms, each corresponding to a unique marketplace instance, as shown in Figure 19. Every platform will operate its own smart contract on the Ethereum Blockchain, managed by the administrator of that marketplace instance. These platforms will serve various shops and fulfill the royalty entitlements of multiple authors.

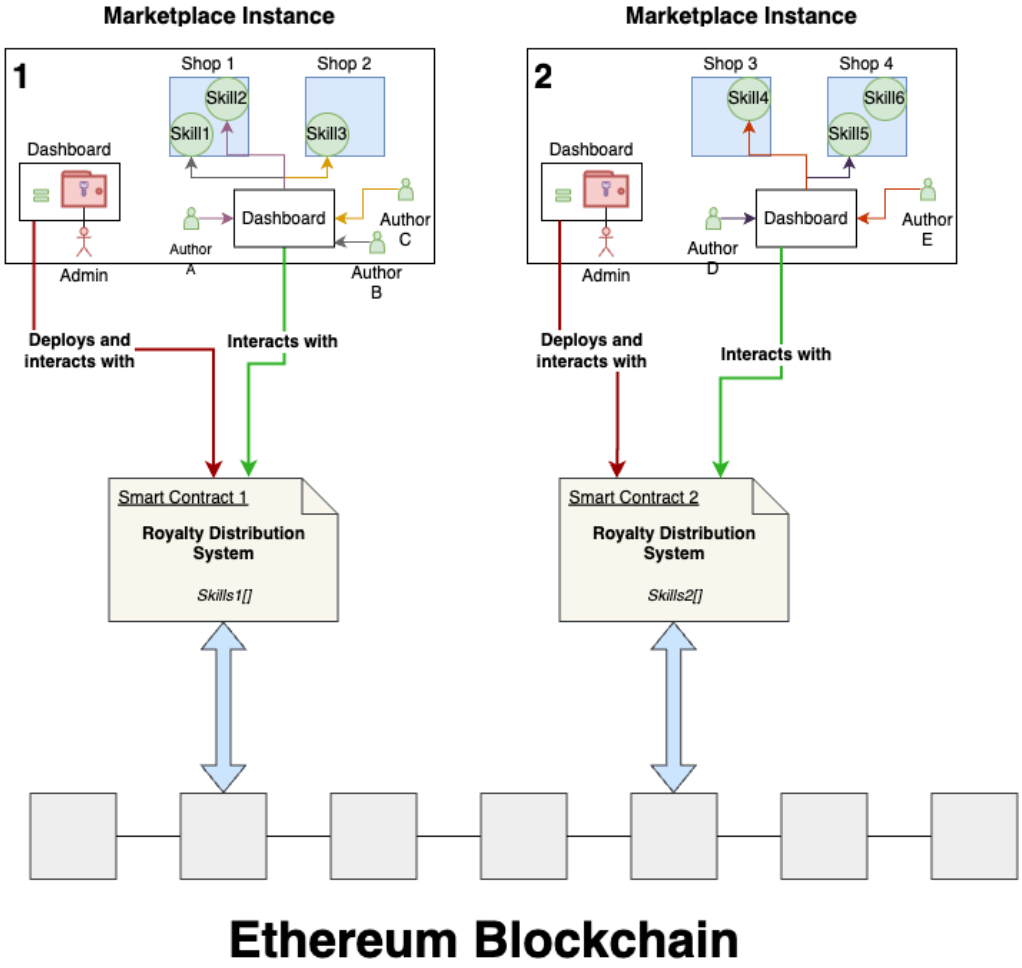

Figure 19 - Multiple systems running simultaneously capturing royalties from all Marketplaces.

9. CONCLUSION

In Conclusion the Royalty Distribution Module (RDM) aims to effectively bridge traditional marketplace products with the dynamic Web3 ecosystem. As a decentralized application on the Ethereum blockchain, it marks a pivotal shift in converting conventional marketplace skills into Non-Fungible Tokens, in line with ERC-721 and potentially ERC-2981 standards. These frameworks guarantee a system that is transparent, verifiable, and universally compatible for managing and distributing royalties to skill developers.

Key features of the RDM, including tokenization and royalty management, and secure, transparent ownership transactions, are set to revolutionize the valuation and transfer of digital assets. The capability for skill upload and tokenization, along with real-time royalty calculations, will enhance the user-friendliness and efficiency of decentralized royalty management.

Lessons Learned

Throughout the design of RDM, significant insights were gained. One of the primary lessons is the importance of aligning blockchain technology with user-centric design to ensure accessibility and ease of use. Additionally, the integration of RDM with traditional marketplaces highlighted the need for adaptable and scalable solutions in the rapidly evolving digital asset space. These lessons have been instrumental in refining our approach and will continue to guide future enhancements.

Next Steps

As we move forward, the strategy for deploying and integrating RDM with PrestaShop involves several key phases:

Smart Contract Deployment and Testing: The initial deployment of the RDM smart contract on the Ethereum network will be followed by thorough testing to ensure optimal security and functionality.

Integrating with PrestaShop: Utilizing PrestaShop hooks, we plan to embed RDM's functionality seamlessly into the marketplace, facilitating effortless interactions between PrestaShop and the RDM.

User-Friendly Interface Development: A major focus will be on developing an intuitive interface for both the Instance Administrator and the Skill Developer, aiming to streamline their interaction with the platform.

In the immediate future, our team will concentrate on fine-tuning these aspects, ensuring that RDM not only meets but exceeds the expectations of its users.

10. BIBLIOGRAPHY

[1] V. Buterin, 2013. [Online]. Available: https://ethereum.org/en/whitepaper/.

[2] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access,* 2016.

[3] R. Böhme, N. Christin, B. Edelman and T. Moore, "Bitcoin: Economics, Technology, and Governance," *Journal of Economic Perspectives,* 2015.

[4] M. Risius and K. Spohrer, "A Blockchain Research Framework," *Business & Information Systems Engineering,* 2017.

[5] C. Catalini and J. S. Gans, "Some simple economics of the Blockchain," 2016.

[6] V. Gatteschi, F. Lamberti, C. Demartini, C. Pranteda and V. Santamaría, "Blockchain and smart contracts for insurance: Is the technology mature enough?," 2018.

[7] G. Wood. [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf.

[8] E. Commission. [Online]. Available: https://ec.europa.eu/digital-building-blocks/wikis/display/EBSI/What+is+EBSI.

[9] W. Entriken, D. Shirley, J. Evans and N. Sachs, 2018. [Online]. Available: https://eips.ethereum.org/EIPS/eip-721.

[10] M. Madine, K. Salah, R. Jayaraman and J. Zemerly, "NFTs for Open-Source and Commercial Software Licensing and Royalties," *IEEE Access,* 2023.

[11] S. Nakamoto, 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf.

[12] M. Conoscenti, A. Vetro and J. C. De Martin, "Blockchain for the Internet of Things: a systematic literature review," 2016.