



**D7.5 IPR Authoring tool & Transaction Management
Strategies for iPRODUCE Social Manufacturing Missions 2**

CERTH

June 2023

DELIVERABLE INFORMATION	
Author(s)/ Organisation(s)	C. Tsotakis (CERTH), T. Kontodina (CERTH), T. Restas (CERTH), Dr. E.M Pechlivani (CERTH), Dr. D. Ioannidis (CERTH)
Document type	Report
Document code	D7.5
Document name	IPR Authoring tool & Transaction Management Strategies for iPRODUCE Social Manufacturing Missions
Status	EU
Work Package / Task	WP7, T7.3
Delivery Date (DoA)	M42
Actual Delivery Date	June 2023
Abstract	This document reports the results of the activities carried out by M42 within the context of WP7 (Sharing Economy Business Models and Execution Tools), particularly in the Task 7.3 "IPR & Transaction Management Strategies & Automation". IPR Management strategies, Blockchain technologies and frameworks have been investigated in order to reinforce the Design Thinking processes of product co-creation. In addition, the architecture and functional overview of the IPR Authoring Tool has been defined and presented.

DELIVERABLE HISTORY			
Date	Version	Author (Partner)	Summary of main changes
20/04/2023	V0.1	C. Tsotakis (CERTH), T. Kontodina (CERTH), T. Restas (CERTH).	Initial version (ToC)
21/06/2023	V0.2	C. Tsotakis (CERTH), T. Kontodina (CERTH), T. Restas (CERTH).	Draft version
23/06/2023	V0.2.1	T. Kolyvas (EDLUX)	1 st Peer Review
26/06/2023	V0.3	C. Tsotakis (CERTH), T. Kontodina (CERTH), T. Restas (CERTH).	Final Version with integrated comments and suggestions from the peer review

DISSEMINATION LEVEL		
PU	Public	
PP	Restricted to other programme participants (including the EC services)	
RE	Restricted to a group specified by the consortium (including the EC services)	
CO	Confidential, only for the members of the consortium (including the EC)	x



DISCLAIMER

This document contains information and material that is the copyright of iPRODUCE consortium parties and may not be reproduced or copied without consent.

© The information and material included in this document are the responsibility of the authors and do not necessarily reflect the opinion of the European Union. Neither the European Union institutions and bodies nor any person acting on behalf may be held responsible for the use that may be made of the information and material contained herein.

© iPRODUCE Consortium, 2020-2023.

Reproduction is authorized provided the present document and authors are acknowledged

iPRODUCE • Grant Agreement: 870037 • Innovation Action • 2020 – 2023 | Duration: 42 months

Topic: DT-FOF-05-2019: Open Innovation for collaborative production engineering (IA)

Executive Summary

This document is a deliverable of the iPRODUCE project, funded by the European Commission's Directorate-General for Research and Innovation (DG RTD), under its Horizon 2020 Research and innovation programme (H2020). The document reports the results of the activities carried out by M42 within the context of WP7 (iPRODUCE Sharing Economy Business Models and Execution Tools), particularly in the Task 7.3 "IPR & Transaction Management Strategies & Automation".

The objectives of the WP7 are:

- To devise novel business models and IPR management strategies and tools to simplify and automate multi-stakeholder interactions. (Business models for shorter time-to-market product engineering. Build trust through smart contracts)

This document represents the final updated rendition of the D7.4, with completion scheduled for M42. The previous work conducted by M18, introduces a functional prototype of the iPRODUCE, IPR Authoring Tool. Within the components of the IPR Authoring Tool, a local Blockchain framework accompanied by a comprehensive and intuitive user interface has been developed, facilitating the reinforcement of the Design Thinking processes of product co-creation.

Initially, in order to define and justify the component's architectural and functional specifications, extensive research about existing Blockchain technologies, frameworks and verifications tools related to the Ricardian Contracts has been carried out. The benefits of the Ricardian Contracts emerged as results.

Additionally, the steps of the IPR management strategies, including the Ricardian Contract template functionalities and the IPR Business Analysis and Management Provisions are presented in chapter 2. The interconnection of the IPR Authoring Tool with the rest of the OpIS platform components such as the Marketplace and the OpIS Data Repository as well as the business scenario analysis are introduced in chapter 3.

In the first version of the deliverable, the main focus was to create a Blockchain based smart contract platform and a graphical interface that empowers users to create simple NDA contracts for a team product. Moreover, the connection with the OpIS platform and the OpIS data repository components had been established.

The updated version of the deliverable, D7.5, has successfully implemented the additions, including the full integration of the Blockchain-based smart contract platform and graphical interface with the OpIS platform components. This integration encompasses seamless connectivity with the Matchmaking and Marketplace features. Furthermore, the deliverable provides a comprehensive description of the integration process for the updated IPR Authoring Tool, along with its extensive functional capabilities that effectively address all essential operational needs.

Table of contents

Executive Summary	ii
Table of contents	iii
1. Introduction	1
1.1. Scope of the Deliverable	1
1.2. Structure of the Deliverable	1
1.3. Relation with other Tasks and Deliverables	2
2. IPR Management strategies	3
2.1. Introduction	3
2.2. Template functionalities	3
2.3. IPR Business Analysis and Management Provisions	4
2.3.1. IPR Authoring Tool Business Scenarios Analysis	4
2.3.2. IPR Management Provisions	6
3. Internal and external Architecture	10
3.1. Internal Architecture	10
3.2. External Architecture	11
4. Blockchain technology	13
4.1. Frameworks	13
4.1.1. Ethereum	13
4.1.2. EOS	14
4.1.3. EOS in comparison with Ethereum within iPRODUCE	15
4.2. Blockchain contracts	16
4.2.1. Smart Contracts	20
4.2.2. Ricardian Contracts	22
4.3. Legal Analysis	24
4.3.1. Smart Contracts and Contract Law	25
5. IPR Authoring tool Implementation	26
5.1. Overview	26
5.2. Front-end	27
5.2.1. Welcome Page	27
5.2.2. Dashboard Page	28
5.2.3. Documentation Page	35
5.2.4. Contact Page	36
5.3. Template Creation & Verification	37

5.4. Ricardian Contract Deployment	38
5.5. The Application Layer	39
5.5.1. Component Specifications and Functional View	39
5.5.2. Back-end services	40
5.5.3. Approach and used technologies for IPR	60
6. Conclusions	63
7. References	64
ANNEX	65

List of Tables

Table 3-1: Subcomponents - Ricardian Toolkit	10
Table 3-2: Subcomponent description	10
Table 4-1: Smart versus Ricardian Contracts	24
Table 5-1: Human Readable IP Rules Manager	39
Table 5-2: Machine-Readable Contract Manager	40
Table 5-3: Decentralized IP rules manager	40
Table 5-4: Retrieve Accepted Ricardian Contracts	40
Table 5-5: Retrieve Pending Ricardian Contracts	41
Table 5-6: Retrieve Rejected Ricardian Contracts	42
Table 5-7: Retrieve Deployed Ricardian Contracts	43
Table 5-8: Retrieve transactionID from the blockchain	44
Table 5-9: Reject Ricardian Contract	47
Table 5-10: Add Ricardian Contract to Pending Contracts Page	48
Table 5-11: Deploy Ricardian Contract	49
Table 5-12: Retrieve Users' Notifications	51
Table 5-13: Retrieves the Users' Contracts, Teams, Products, and Team members	53
Table 5-14: Retrieve User's Avatar	56
Table 5-15: Retrieves Block-chain Account Ricardian Contracts	56
Table 5-16: Create Block-chain Account	58
Table 5-17: Import Contract into the Block-chain	59

List of Figures

Figure 2-1: Operational Sequent Steps of IPR Authoring Tool.....	3
Figure 3-1: Component diagram of the IPR Authoring Tool.	10
Figure 3-2: IPR Authoring Tool - Ricardian Toolkit service sequence diagram.	11
Figure 3-3: External architecture - Interconnection between the main components of the OpIS platform.	11
Figure 4-1: Overview of workflow in the Ethereum network.....	14
Figure 4-2: The properties of Distributed Ledger Technology (DLT).	17
Figure 4-3: Generic Chain of Blocks	20
Figure 4-4: Operational mechanism of smart contract.....	22
Figure 4-5: Overview of Ricardian Contracts.	23
Figure 5-1: Login page.....	27
Figure 5-2: Dashboard Page.....	28
Figure 5-3: Pending Contracts	28
Figure 5-4: Template Ricardian Contract	29
Figure 5-5: Accept Contract pop-up	30
Figure 5-6: Edit Contract.....	30
Figure 5-7: User Edited Contract	31
Figure 5-8: Reject Contract Dialog box	31
Figure 5-9: Reject Contract pop-up	31
Figure 5-10: Accepted Contracts	32
Figure 5-11: Accepted Contract Agreement	32
Figure 5-12: Rejected Contracts	33
Figure 5-13: Rejected Contract Agreement	33
Figure 5-14: Deployed Contracts	34
Figure 5-16: Create Contract Component	35
Figure 5-17: Contract Type Dialog Box	35
Figure 5-18: IPR Documentation Page	36
Figure 5-19: IPR Contact Us Page	37
Figure 5-25: Contract Creation & Verification	38
Figure 5-26: Ricardian Contract Deployment	39

1. Introduction

Intellectual Property (IP) functions as a comprehensive representation of the intangible assets owned by a company and legally protected from authorized use and conversion by third parties. The basis of this term, referring to the non-physical assets an organization holds, is resolved around the fact that products which are the results of human effort and intellect should be properly protected by laws and legislations equivalent to their physical property counterparts. Thus, most of the developed economies of the world have issued legal measures in place to safeguard both forms of property. All the above measures, provide economic incentives to people as they allow them to profit from the knowledge and exploitation, they create through them.

The non-physical existence of intellectual property presents difficulties when compared with traditional property which has physical forms. Specifically, unlike physical goods, the indivisible nature of intellectual assets means that there is an unlimited capacity for copying and redistributing and idea, patent or trade secret. This creates the need for finding the golden mean between the giving the incentives for the creation of intellectual assets while also being protective enough to not prevent the sharing and advancement of them. All the above lead to the necessity of Intellectual Property Rights Management (IPRM).

Intellectual Property Management covers the decisions and strategies which are deployed to protect and preserve the integrity of the intellectual property rights of an organization while also helping the maximization of profits from the commercialization benefits.

For this reason, CERTH as part of the iPRODUCE initiative aims to provide an intellectual property management system through Smart Contracts using modern technologies such as Blockchain and visual web tools for the authoring and deployment of them.

1.1. Scope of the Deliverable

The scope of this deliverable is to provide an overview of the Intellectual Property Rights authoring tool and Transaction Management Strategies for iPRODUCE's social manufacturing platform. Specifically, the first version of the designed and developed tool as well as its architectural and technical implementations will be overviewed.

Moreover, Blockchain technology and its role in all stages of an Intellectual Property Rights management system along with its technical description and the way they facilitate the proof-of-authenticity of the aforementioned rights will be also thoroughly presented. Furthermore, the importance of the Smart and Ricardian Contracts as well as the methods the visual tool employs to encode and compile the Contract compatible high-level language into machine language will be also outlined.

Finally, this deliverable aims to cover the technical details of the final version of an intuitive User Interface (UI) for contract template creation aimed at non-experienced users, along with the underline server-client backend technologies deployed for the purposes of the project.

1.2. Structure of the Deliverable

In this deliverable the main objectives and relevant technologies that the Intellectual Property Rights authoring tool employs to encounter the use cases defined by the iPRODUCE platform are described in detail. In addition, a thorough outline of the design of the components and their underline architecture is also provided.

All systems developed for the aims of the deliverable are thoroughly described in the following chapters:

- **IPR Management Strategies:** Outlines the intellectual property management and transaction strategies, contract templates, and non-disclosure agreements utilized by the collaborative production platform's MMCs teams. Business scenarios analysis of the IPR authoring tool are described and highlights how blockchain-based Ricardian contracts enhance CMDFs by offering secure IP protection, transparent collaboration, streamlined licensing, simplified dispute resolution, and monetization opportunities for co-creation.
- **Internal & External Architecture:** Detailed overview of the IPR Tool's architecture and technical decisions that governed the design and implementation process of the current version of the tool presented in this deliverable.
- **Blockchain Technology:** A comprehensive investigation of the role Blockchain technologies play with Smart Contracts, their underline specifics and the leading tools and platforms used in the various processes involved such as the Ethereum and EOS platforms as well as the role Ricardian Contracts play.
- **IPR Authoring Tool Implementation:** Detailed description of the design, development and technical details that were necessary for the construction of the iPRODUCE IPR Authoring Tool visual authoring tool responsible for managing everything related to Ricardian Contracts through an intuitive User Interface.
- **IPR Management Provisions:** Detailed description of how blockchain-based Ricardian contracts enhance CMDFs, like a 3D printing company, by providing secure IP protection, transparent collaboration, streamlined licensing, simplified dispute resolution, and monetization opportunities for co-creation.
- **Conclusions:** Review of the results which came from the initial design and implementation phase of the IPR Authoring Tool's first version.

1.3. Relation with other Tasks and Deliverables

This deliverable reports the second and final version (v2) of the IPR Authoring tool & Transaction Management Strategies for iPRODUCE Social Manufacturing Missions. This updated version incorporates the necessary enhancements and improvements as specified in Deliverable 7.5.

Related Task and Deliverables from the rest of the iPRODUCE Toolchain include the following:

1.3.1. The basis of the iPRODUCE platform social manufacturing architecture is described in WP4-T4.1 – D4.2, which dictated the overall design decisions in IPR Authoring Tool Business Scenarios Analysis

According to the deliverable D2.5 – “Definition of iPRODUCE demonstration”, initially, six cMDFs were designated to represent consumer goods from various industrial sectors, including furniture, automotive, microelectronics, medical, and electronics. These cMDFs outlined sixteen business scenarios that described the application areas of the OpIS platform. However, due to recent developments (particularly the bankruptcy of BETAFACTORY – the Danish cMDF), the total number of cMDFs has been reduced to five. Consequently, an updated version of D2.8 now includes nineteen business case scenarios, reflecting the modifications in the structure of the cMDFs. These business scenarios essentially analyse the functionalities which the OpIS platform offers by providing a comprehensive description on how the user will interact with the platform and the other stakeholders in order to improve the co-creation process

under of the protection of a cMDF ecosystem. A business scenario is a complete description of a business problem, both in business and in architectural terms, which enables individual requirements to be viewed in relation to one another in the context of the overall problem. From the OpIS platform perspective, all the functionalities of the co-creation ecosystem are provided inside the long description of each use case of the D2.5.

All the users who are involved in the co-creation process of a new product or idea, they are called to cover and protect their product/idea. Inside the OpIS platform, it is going to be developed a tool which will protect the legal rights of everyone involved. As the technology is improved, the need to find an efficient, secure and fast way to protect the idea of a product becomes imperative. One of the most important roles of the social manufacturing missions consists the protection of the intellectual rights of the stakeholders during all the co-creation process (from the initial idea to the prototyping of the product). The IPR Authoring Tool using Ricardian Contracts is coming to support IPR during various phases of the co-creation process of a product by providing a legal document in a digitized form. It is important this document will be human-readable and machine-readable as well, so that both machines and people will be able to read and edit this. So, a Ricardian Contract was created as a means of associating a legally binding and digitally connected document with a specific object or value. A Ricardian Contract converts all information from a legal document into a computer readable format. In this sense, it serves as both a legal agreement and a mechanism for securely integrating the agreement into a digital infrastructure, while also providing a high level of security due to cryptographic identification.

The following Greek UC example may help to better appreciate the significance of the IPR Authoring Tool.

Greek cMDF mission is to bridge the gap between SME's and Makerspaces. Aidplex – CERTH, with expertise in medical and 3D printing sector, is going to help any company or customer to achieve better treatment experience. The orthopaedic back brace solution is designed by Aidplex with the aim of higher comfort levels and retrofitting the resulted design with IoT sensors, for scoliosis, hyphosis, or similar spinal deformities. The overarching goal is to finetune the design of a back brace by examining aspects like weight distribution, modularity, size adaptability and overall comfort, whilst IoT sensors will help patients self-assess and adapt their back braces leading to higher degrees of adherence and outcome. In this scenario as well as in all the scenarios which regards the social manufacturing, the need of IPR protection constitutes an important point in the co-creation missions. For co-creation works, the author or creator of the work is the first owner and it is important to protect his/her rights. Inside of this UC, there are 3 involved users (User 1 – Aidplex as designer/manufacturer, User 2 – consumer as patient, User 3 – CERTH as manufacturer). All the users have the need to protect their rights from eavesdropping and copying. Using this iPRODUCE social manufacturing platform is offered the IPR Authoring Tool which comes to protect all the rights of the co-creation and production process by providing a legal binding agreement which will include the following: (1) who is the product, (2) who are the involved manufacturers, and (3) between who the product has been created. It is important to be mentioned some essential parts of this agreement which are:

- Parties: How many parties are involved? Who are the parties making this agreement? Who are their representatives?
- An element in Time: What is the validity of the Contract? Is it applicable for a limited period of time or forever? What does it define in terms of time? For example, a deal needs to be reached withing three months, or the Contract gets null and void.
- Adding Exceptions for Different Possibilities: For example, what happens when one of the parties dies or what happens when an involved company go bankrupt? Or similar exceptions.

- Conditions: Any condition can be claused.

Covering all these parts inside of the agreement, all the users could feel legally guaranteed. Furthermore, one more crucial point of the IPR Authoring Tool is how much security this provides. Each document in the Contract has unique identification by its hash as result Ricardian Contracts are very secure as they use a cryptographic signature. This also offers protection from a commonly used tactic in legal agreements called frog boiling. Under traditional legal agreements, an issuer with the upper hand keeps changing the terms in the agreement during the execution. This is not possible with this IPR Authoring Tool.

By using the IPR Authoring Tool from the starting point of the co-creation and production process, all the involved users (patients, designers, manufacturers) feel the sense of the security from the legal perspective. As they are legally secured for the starting to the end (prototyping) stage, they can move forward to the co-creation and prototyping process without any constraint.

In the above scenario, the new technology which is offered is that the doctor and the patient can be informed when the right time for a new brace has come, due to child's growth, achieving the best possible fit of the medical device using an IoT system. This IoT system constitutes the new technology which do this product unique and innovative. If this product will be co-created using this social manufacturing platform which the iPRODUCE offers, all the involved parties will be legally protected in all the co-creation and prototyping stages. For example, CERTH as 3D manufacturer has an important role during the co-creation and prototyping process as it is responsible to print the 3D printed parts of the back brace solution using its expert premises. In case, CERTH and Aidplex which are the most crucial parties, because they should exchange knowledge (for example 3D CAD models of the back brace), are not legally secured, they will not be able to trust one each other as result, their cooperation becomes uncomfortable and difficult.

So, the IPR Authoring Tool constitutes a necessary component of the co-creation process as this save effort, costs and time by providing machine-readable legal contracts which are not open to any interpretation, which is the main drawback of human-readable legal contracts.

1.3.2. IPR Management Provisions

Collaborative Manufacturing Demonstration Facilities (CMDFs) are spaces where various stakeholders can co-create innovative products using advanced manufacturing technologies. These facilities bring together manufacturers, researchers, designers, entrepreneurs, and other partners to explore new manufacturing technologies, prototype designs, and develop groundbreaking products. CMDFs enable participants to access state-of-the-art equipment, share knowledge and expertise, and collaborate on projects that address real-world challenges and opportunities.

One of the key challenges faced by CMDFs is the management of intellectual property rights (IPR). Intellectual property is of utmost importance in collaborative manufacturing environments, where multiple entities contribute their ideas, designs, and know-how. Protecting the intellectual property rights of participants involved in collaborative projects is essential to ensure fair compensation, recognition, and trust. However, traditional IPR management methods are often complex, costly, and inefficient. They rely on centralized authorities and intermediaries to enforce intellectual property rights and resolve disputes. They also involve manual processes and paperwork that can be prone to errors or manipulation.

To overcome these challenges, CMDFs can implement IPR management provisions using blockchain-based Ricardian contracts. Blockchain-based Ricardian contracts are digital contracts that combine

natural language and code to define the terms and conditions of an agreement. They are executed on a blockchain network, a distributed ledger that securely records and validates transactions among participants. Blockchain-based Ricardian contracts offer several benefits for IPR management in CMDFs:

- **Secure intellectual property protection:** Blockchain technology enables CMDFs to securely store and timestamp digital records of designs, concepts, research findings, and other valuable intellectual property assets created by the participants. The blockchain's immutability and tamper-proof nature ensures that these records cannot be altered or manipulated, providing indisputable evidence of ownership and establishing a robust foundation for intellectual property protection.
- **Transparent attribution and collaboration:** Blockchain technology enables CMDFs to record the contributions and interactions of the participants on the blockchain network, creating an auditable and tamper-proof trail of collaboration. This fosters trust and accountability among the participants, as they can independently verify the authenticity and ownership of the intellectual property assets. This transparency also ensures that each participant receives proper recognition and credit for their creative inputs.
- **Streamlined licensing and royalty management:** Blockchain-based Ricardian contracts enable CMDFs to embed licensing terms and conditions directly into the contract code, defining the rights, limitations, and payment structures for the intellectual property assets. When a licensing agreement is established, the Ricardian contract is executed on the blockchain network, automatically enforcing the agreed-upon terms. Smart contracts handle royalty calculations and distributions based on predefined rules, ensuring fair compensation for the usage of intellectual property.
- **Simplified dispute resolution:** Blockchain-based Ricardian contracts enable CMDFs to incorporate predefined arbitration protocols within the contract code, streamlining the dispute resolution process. In case of disagreements or infringement claims, the parties involved can refer to the arbitration provisions embedded in the contract. The blockchain network provides an immutable and transparent source of evidence, capturing all relevant transactions and interactions related to the intellectual property. This ensures a fair and efficient resolution process, minimizing legal costs and maintaining productive collaborative relationships.
- **Monetization and commercialization opportunities:** Blockchain technology enables CMDFs to showcase the value and uniqueness of their co-created products or designs. By securely recording and validating the development process and ownership of intellectual property on the blockchain network, CMDFs can demonstrate the provenance and originality of their creations. This can attract potential investors, licensing partners, or customers seeking innovative solutions. The streamlined licensing and royalty management facilitated by blockchain-based Ricardian contracts make it easier for CMDFs to establish licensing agreements and monetize their intellectual property.

By implementing IPR management provisions using blockchain-based Ricardian contracts, CMDFs can create a secure and transparent environment for co-creation while protecting the intellectual property rights of all participants. This approach enhances collaboration, innovation, and value creation for both CMDFs and their collaborators.

One example of a CMDF that can benefit from this approach is a 3D printing company that specializes in product design and prototyping. This company collaborates with designers, engineers, manufacturers, and other partners to co-create new products using 3D printing technology. 3D printing is a form of

additive manufacturing that creates physical objects by depositing layers of material on top of each other based on a digital model. 3D printing offers several advantages for product design and development, such as:

- **Rapid prototyping:** 3D printing enables faster and cheaper production of prototypes compared to traditional methods. Prototypes can be created within hours or days instead of weeks or months. This allows for more iterations and feedback, improving the quality and functionality of the final product.
- **Customization and personalization:** 3D printing allows for greater flexibility and creativity in product design. Products can be customized and personalized according to the specific needs and preferences of customers or users. This can enhance customer satisfaction and loyalty, as well as create a competitive edge in the market.
- **Reduced waste and environmental impact:** 3D printing reduces material waste and energy consumption compared to traditional methods. 3D printing only uses the amount of material needed to create the object, minimizing excess material and scrap. 3D printing also enables the use of recycled or biodegradable materials, reducing the environmental impact of product manufacturing.
- **New possibilities and solutions:** 3D printing enables the creation of complex and intricate shapes and structures that are difficult or impossible to achieve with traditional methods. 3D printing also enables the integration of multiple materials and functions within a single object, creating new possibilities and solutions for product design and development.

To facilitate the co-creation process using 3D printing technology, the company implements IPR management provisions using blockchain-based Ricardian contracts. This approach offers several benefits for the company and its collaborators:

- **Secure intellectual property protection:** Using blockchain technology, the company can create secure and permanent digital records of the designs, concepts, and other valuable intellectual property assets that the collaborators produce. These records are unchangeable and resistant to tampering, which proves who owns them and helps protect their intellectual property rights.
- **Transparent attribution and collaboration:** The company uses the blockchain network to track and store the co-creation activities and outputs of the collaborators, making them verifiable and immutable. This builds trust and responsibility among the collaborators, as they can confirm the origin and ownership of the intellectual property assets. This clarity also guarantees that each collaborator gets fair recognition and credit for their creative inputs.
- **Streamlined licensing and royalty management:** The company incorporates licensing terms and conditions directly into the contract code, specifying the rights, restrictions, and payment structures for the intellectual property assets. When a licensing agreement is formed, the Ricardian contract is executed on the blockchain network, automatically implementing the agreed-upon terms. Smart contracts manage royalty calculations and distributions based on predefined rules, ensuring equitable compensation for the usage of intellectual property.
- **Simplified dispute resolution:** The company includes predefined arbitration protocols within the contract code, simplifying the dispute resolution process. In case of conflicts or infringement claims, the parties involved can refer to the arbitration provisions embedded in the contract. The blockchain network provides an unchangeable and transparent source of evidence, capturing all relevant transactions and

interactions related to the intellectual property. This ensures a just and effective resolution process, reducing legal costs and preserving productive collaborative relationships.

- **Monetization and commercialization opportunities:** The company utilizes blockchain technology to showcase the value and uniqueness of its co-created products or designs. By safely preserving and validating the development process and ownership of intellectual property on the blockchain network, the company can demonstrate the origin and novelty of its creations. This can attract potential investors, licensing partners, or customers seeking innovative solutions. The simplified licensing and royalty management facilitated by blockchain-based Ricardian contracts make it easier for the company to form licensing agreements and monetize its intellectual property.

By implementing IPR management provisions using blockchain-based Ricardian contracts, a company or fablab, etc. (e.g. 3D printing company) creates a secure and transparent environment for co-creation while protecting the intellectual property rights of all participants. This approach enhances collaboration, innovation, and value creation for both the company and its collaborators.

- Internal and external Architecture
- The KPIs and Use Case definition of the iPRODUCE initiative is defined in T2.4 that the IPR Authoring tool Implementation section aims to cover.
- An outline of the issues regarding current IPR management techniques and issues is provided in the processes of T2.5. The IPR Management strategies chapter offers the solution of Ricardian Contracts to combat said issues.
- In WP3, the first concepts of the OpIS platform, training toolkit and local CMDFs are examined which affect the incorporation of the solutions offered in the IPR Management strategies and IPR Authoring tool Implementation sections.
- D7.1 describes the commonly used tools and technologies as well as the upcoming relevant ones, regarding the makerspace contexts, serving as the initial state of the art for this report.

2. IPR Management strategies

2.1. Introduction

iPRODUCE identifies a set of IPR and transaction management mechanisms inside the OpIS platform that can be used to facilitate the formation and operation of ad hoc multi-sided teams, that take on collaborative manufacturing jobs under accountability. For the 'seamless' implementation of such strategies the project will use Ricardian contracts.

The legal advantages of a Ricardian contract arise from the use of mark-up language embedded in a mostly legal prose document, resulting in reduced transaction costs, faster dispute resolution, better contract enforcement and increased transparency.

The benefits of a Ricardian contract from a data processing perspective come from the software design plan that digitizes documents and lets them participate in financial transactions, such as payments, without losing the richness of the contract tradition. One important thing that should be mentioned is that publishing the content and referencing it through the unique cryptographic message nullifies frauds based on multiple presentations.

2.2. Template functionalities

During the iPRODUCE project, the implementation of an IPR Authoring Tool will be compassed. IPR Authoring Tool is a visual authoring tool to determine a set of simple, yet relevant, ruse that are involved in the context of Design Thinking process stages for product co-creation. Its operational sequent steps include:

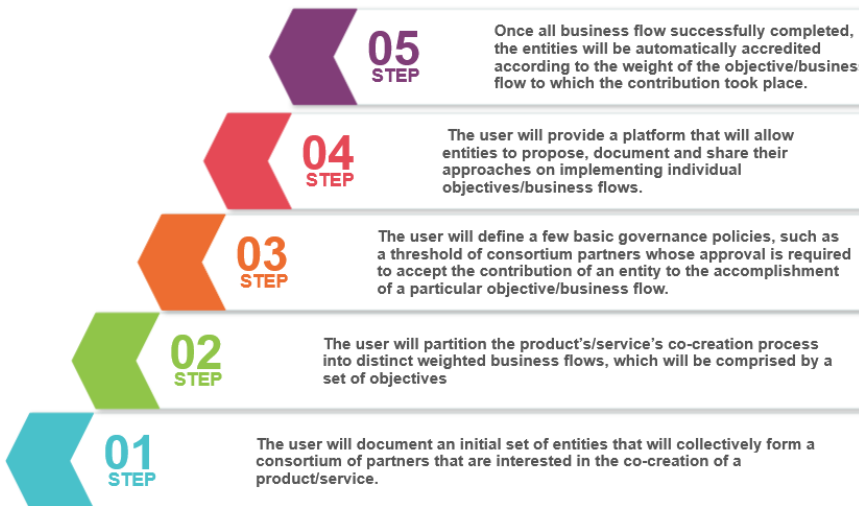


Figure 2-1: Operational Sequent Steps of IPR Authoring Tool.

As the above figure indicates, the first step is that the user will create a list of entities that will get together to form a consortium of partners interested in co-creating a product or service. As a second step, the user will divide the co-creation process for the product/service into discrete weighted business processes, each with a set of objectives. Subsequently, the user will set a few fundamental governance policies, such as a minimum number of consortium partners required to accept an entity's contribution to the achievement of a specific objective/business flow. Furthermore, the user will provide a platform for entities to propose, document, and communicate their approaches to achieving specific goals/business objectives. Finally, once all business flows have been completed successfully, the entities will be automatically accredited based on the weight of the contribution to the objective business flow. All the above steps constitute the necessary requirements in a high-level overview. In the following sections the internal architecture is presented as well as the interconnection of the IPR Authoring Tool with the other components of the OpIS platform is introduced.

2.3. IPR Business Analysis and Management Provisions

The iPRODUCE project combines the use of the IPR Authoring Tool and blockchain-based Ricardian contracts to safeguard intellectual property rights (IPR) and promote secure collaboration in the co-creation and manufacturing processes. The IPR Authoring Tool offers machine-readable legal agreements, ensuring clarity and minimizing infringement risks. Meanwhile, cMDFs implement Ricardian contracts on the blockchain for transparent attribution, streamlined licensing, and simplified dispute resolution. These measures benefit companies by enabling faster prototyping, reduced waste, and new design possibilities. Overall, this approach enhances collaboration, innovation, and value creation, while providing comprehensive protection and efficient management of IPR.

2.3.1. IPR Authoring Tool Business Scenarios Analysis

According to the deliverable D2.5 – “Definition of iPRODUCE demonstration”, initially, six cMDFs were designated to represent consumer goods from various industrial sectors, including furniture, automotive, microelectronics, medical, and electronics. These cMDFs outlined sixteen business scenarios that described the application areas of the OpIS platform. However, due to recent developments (particularly the bankruptcy of BETAFACTORY – the Danish cMDF), the total number of cMDFs has been reduced to five. Consequently, an updated version of D2.8 now includes nineteen business case scenarios, reflecting the modifications in the structure of the cMDFs. These business scenarios essentially analyse the functionalities which the OpIS platform offers by providing a comprehensive description on how the user will interact with the platform and the other stakeholders in order to improve the co-creation process under the protection of a cMDF ecosystem. A business scenario is a complete description of a business problem, both in business and in architectural terms, which enables individual requirements to be viewed in relation to one another in the context of the overall problem. From the OpIS platform perspective, all the functionalities of the co-creation ecosystem are provided inside the long description of each use case of the D2.5.

All the users who are involved in the co-creation process of a new product or idea, they are called to cover and protect their product/idea. Inside the OpIS platform, it is going to be developed a tool which will protect the legal rights of everyone involved. As the technology is improved, the need to find an efficient, secure and fast way to protect the idea of a product becomes imperative. One of the most important roles of the social manufacturing missions consists the protection of the intellectual rights of the stakeholders during all the co-creation process (from the initial idea to the prototyping of the product). The IPR Authoring Tool using Ricardian Contracts is coming to support IPR during various phases of the co-creation process of a product by providing a legal document in a digitized form. It is important

this document will be human-readable and machine-readable as well, so that both machines and people will be able to read and edit this. So, a Ricardian Contract was created as a means of associating a legally binding and digitally connected document with a specific object or value. A Ricardian Contract converts all information from a legal document into a computer readable format. In this sense, it serves as both a legal agreement and a mechanism for securely integrating the agreement into a digital infrastructure, while also providing a high level of security due to cryptographic identification.

The following Greek UC example may help to better appreciate the significance of the IPR Authoring Tool.

Greek cMDF mission is to bridge the gap between SME's and Makerspaces. Aidplex – CERTH, with expertise in medical and 3D printing sector, is going to help any company or customer to achieve better treatment experience. The orthopaedic back brace solution is designed by Aidplex with the aim of higher comfort levels and retrofitting the resulted design with IoT sensors, for scoliosis, hyphosis, or similar spinal deformities. The overarching goal is to finetune the design of a back brace by examining aspects like weight distribution, modularity, size adaptability and overall comfort, whilst IoT sensors will help patients self-assess and adapt their back braces leading to higher degrees of adherence and outcome. In this scenario as well as in all the scenarios which regards the social manufacturing, the need of IPR protection constitutes an important point in the co-creation missions. For co-creation works, the author or creator of the work is the first owner and it is important to protect his/her rights. Inside of this UC, there are 3 involved users (User 1 – Aidplex as designer/manufacturer, User 2 – consumer as patient, User 3 – CERTH as manufacturer). All the users have the need to protect their rights from eavesdropping and copying. Using this iPRODUCE social manufacturing platform is offered the IPR Authoring Tool which comes to protect all the rights of the co-creation and production process by providing a legal binding agreement which will include the following: (1) who is the product, (2) who are the involved manufacturers, and (3) between who the product has been created. It is important to be mentioned some essential parts of this agreement which are:

- Parties: How many parties are involved? Who are the parties making this agreement? Who are their representatives?
- An element in Time: What is the validity of the Contract? Is it applicable for a limited period of time or forever? What does it define in terms of time? For example, a deal needs to be reached withing three months, or the Contract gets null and void.
- Adding Exceptions for Different Possibilities: For example, what happens when one of the parties dies or what happens when an involved company go bankrupt? Or similar exceptions.
- Conditions: Any condition can be clausued.

Covering all these parts inside of the agreement, all the users could feel legally guaranteed. Furthermore, one more crucial point of the IPR Authoring Tool is how much security this provides. Each document in the Contract has unique identification by its hash as result Ricardian Contracts are very secure as they use a cryptographic signature. This also offers protection from a commonly used tactic in legal agreements called frog boiling. Under traditional legal agreements, an issuer with the upper hand keeps changing the terms in the agreement during the execution. This is not possible with this IPR Authoring Tool.

By using the IPR Authoring Tool from the starting point of the co-creation and production process, all the involved users (patients, designers, manufacturers) feel the sense of the security from the legal perspective. As they are legally secured for the starting to the end (prototyping) stage, they can move forward to the co-creation and prototyping process without any constraint.

In the above scenario, the new technology which is offered is that the doctor and the patient can be informed when the right time for a new brace has come, due to child's growth, achieving the best possible fit of the medical device using an IoT system. This IoT system constitutes the new technology which do this product unique and innovative. If this product will be co-created using this social manufacturing platform which the iPRODUCE offers, all the involved parties will be legally protected in all the co-creation and prototyping stages. For example, CERTH as 3D manufacturer has an important role during the co-creation and prototyping process as it is responsible to print the 3D printed parts of the back brace solution using its expert premises. In case, CERTH and Aidplex which are the most crucial parties, because they should exchange knowledge (for example 3D CAD models of the back brace), are not legally secured, they will not be able to trust one each other as result, their cooperation becomes uncomfortable and difficult.

So, the IPR Authoring Tool constitutes a necessary component of the co-creation process as this save effort, costs and time by providing machine-readable legal contracts which are not open to any interpretation, which is the main drawback of human-readable legal contracts.

2.3.2. IPR Management Provisions

Collaborative Manufacturing Demonstration Facilities (CMDFs) are spaces where various stakeholders can co-create innovative products using advanced manufacturing technologies. These facilities bring together manufacturers, researchers, designers, entrepreneurs, and other partners to explore new manufacturing technologies, prototype designs, and develop groundbreaking products. CMDFs enable participants to access state-of-the-art equipment, share knowledge and expertise, and collaborate on projects that address real-world challenges and opportunities.

One of the key challenges faced by CMDFs is the management of intellectual property rights (IPR). Intellectual property is of utmost importance in collaborative manufacturing environments, where multiple entities contribute their ideas, designs, and know-how. Protecting the intellectual property rights of participants involved in collaborative projects is essential to ensure fair compensation, recognition, and trust. However, traditional IPR management methods are often complex, costly, and inefficient. They rely on centralized authorities and intermediaries to enforce intellectual property rights and resolve disputes. They also involve manual processes and paperwork that can be prone to errors or manipulation.

To overcome these challenges, CMDFs can implement IPR management provisions using blockchain-based Ricardian contracts. Blockchain-based Ricardian contracts are digital contracts that combine natural language and code to define the terms and conditions of an agreement. They are executed on a blockchain network, a distributed ledger that securely records and validates transactions among participants. Blockchain-based Ricardian contracts offer several benefits for IPR management in CMDFs:

- **Secure intellectual property protection:** Blockchain technology enables CMDFs to securely store and timestamp digital records of designs, concepts, research findings, and other valuable intellectual property assets created by the participants. The blockchain's immutability and tamper-proof nature ensures that these records cannot be altered or manipulated, providing indisputable evidence of ownership and establishing a robust foundation for intellectual property protection.
- **Transparent attribution and collaboration:** Blockchain technology enables CMDFs to record the contributions and interactions of the participants on the blockchain network, creating an auditable and tamper-proof trail of collaboration. This fosters trust and accountability among the participants, as they

can independently verify the authenticity and ownership of the intellectual property assets. This transparency also ensures that each participant receives proper recognition and credit for their creative inputs.

- **Streamlined licensing and royalty management:** Blockchain-based Ricardian contracts enable CMDFs to embed licensing terms and conditions directly into the contract code, defining the rights, limitations, and payment structures for the intellectual property assets. When a licensing agreement is established, the Ricardian contract is executed on the blockchain network, automatically enforcing the agreed-upon terms. Smart contracts handle royalty calculations and distributions based on predefined rules, ensuring fair compensation for the usage of intellectual property.
- **Simplified dispute resolution:** Blockchain-based Ricardian contracts enable CMDFs to incorporate predefined arbitration protocols within the contract code, streamlining the dispute resolution process. In case of disagreements or infringement claims, the parties involved can refer to the arbitration provisions embedded in the contract. The blockchain network provides an immutable and transparent source of evidence, capturing all relevant transactions and interactions related to the intellectual property. This ensures a fair and efficient resolution process, minimizing legal costs and maintaining productive collaborative relationships.
- **Monetization and commercialization opportunities:** Blockchain technology enables CMDFs to showcase the value and uniqueness of their co-created products or designs. By securely recording and validating the development process and ownership of intellectual property on the blockchain network, CMDFs can demonstrate the provenance and originality of their creations. This can attract potential investors, licensing partners, or customers seeking innovative solutions. The streamlined licensing and royalty management facilitated by blockchain-based Ricardian contracts make it easier for CMDFs to establish licensing agreements and monetize their intellectual property.

By implementing IPR management provisions using blockchain-based Ricardian contracts, CMDFs can create a secure and transparent environment for co-creation while protecting the intellectual property rights of all participants. This approach enhances collaboration, innovation, and value creation for both CMDFs and their collaborators.

One example of a CMDF that can benefit from this approach is a 3D printing company that specializes in product design and prototyping. This company collaborates with designers, engineers, manufacturers, and other partners to co-create new products using 3D printing technology. 3D printing is a form of additive manufacturing that creates physical objects by depositing layers of material on top of each other based on a digital model. 3D printing offers several advantages for product design and development, such as:

- **Rapid prototyping:** 3D printing enables faster and cheaper production of prototypes compared to traditional methods. Prototypes can be created within hours or days instead of weeks or months. This allows for more iterations and feedback, improving the quality and functionality of the final product.
- **Customization and personalization:** 3D printing allows for greater flexibility and creativity in product design. Products can be customized and personalized according to the specific needs and preferences of customers or users. This can enhance customer satisfaction and loyalty, as well as create a competitive edge in the market.

- **Reduced waste and environmental impact:** 3D printing reduces material waste and energy consumption compared to traditional methods. 3D printing only uses the amount of material needed to create the object, minimizing excess material and scrap. 3D printing also enables the use of recycled or biodegradable materials, reducing the environmental impact of product manufacturing.
- **New possibilities and solutions:** 3D printing enables the creation of complex and intricate shapes and structures that are difficult or impossible to achieve with traditional methods. 3D printing also enables the integration of multiple materials and functions within a single object, creating new possibilities and solutions for product design and development.

To facilitate the co-creation process using 3D printing technology, the company implements IPR management provisions using blockchain-based Ricardian contracts. This approach offers several benefits for the company and its collaborators:

- **Secure intellectual property protection:** Using blockchain technology, the company can create secure and permanent digital records of the designs, concepts, and other valuable intellectual property assets that the collaborators produce. These records are unchangeable and resistant to tampering, which proves who owns them and helps protect their intellectual property rights.
- **Transparent attribution and collaboration:** The company uses the blockchain network to track and store the co-creation activities and outputs of the collaborators, making them verifiable and immutable. This builds trust and responsibility among the collaborators, as they can confirm the origin and ownership of the intellectual property assets. This clarity also guarantees that each collaborator gets fair recognition and credit for their creative inputs.
- **Streamlined licensing and royalty management:** The company incorporates licensing terms and conditions directly into the contract code, specifying the rights, restrictions, and payment structures for the intellectual property assets. When a licensing agreement is formed, the Ricardian contract is executed on the blockchain network, automatically implementing the agreed-upon terms. Smart contracts manage royalty calculations and distributions based on predefined rules, ensuring equitable compensation for the usage of intellectual property.
- **Simplified dispute resolution:** The company includes predefined arbitration protocols within the contract code, simplifying the dispute resolution process. In case of conflicts or infringement claims, the parties involved can refer to the arbitration provisions embedded in the contract. The blockchain network provides an unchangeable and transparent source of evidence, capturing all relevant transactions and interactions related to the intellectual property. This ensures a just and effective resolution process, reducing legal costs and preserving productive collaborative relationships.
- **Monetization and commercialization opportunities:** The company utilizes blockchain technology to showcase the value and uniqueness of its co-created products or designs. By safely preserving and validating the development process and ownership of intellectual property on the blockchain network, the company can demonstrate the origin and novelty of its creations. This can attract potential investors, licensing partners, or customers seeking innovative solutions. The simplified licensing and royalty management facilitated by blockchain-based Ricardian contracts make it easier for the company to form licensing agreements and monetize its intellectual property.

By implementing IPR management provisions using blockchain-based Ricardian contracts, a company or fablab, etc. (e.g. 3D printing company) creates a secure and transparent environment for co-creation while protecting the intellectual property rights of all participants. This approach enhances collaboration, innovation, and value creation for both the company and its collaborators.

3. Internal and external Architecture

In this section, the internal and the external architecture will be introduced. The “internal” architecture deals with the sub components related to the overview of the system. The “external” architecture concerns the interconnection of the system with the other components of an IoT system, such as the social manufacturing platform.

3.1. Internal Architecture

The sub components of the IPR Authoring Tool are depicted in the Table 1:

Table 3-1: Subcomponents - Ricardian Toolkit

Main Component	Sub - components
IPR Authoring Tool - Ricardian Toolkit	Human readable IP rules manager
	Machine readable contract manager
	Decentralised IP rules manager

In the following Figure 3-1 the interconnection between the sub components of the IPR Authoring Tool - Ricardian Toolkit is illustrated:

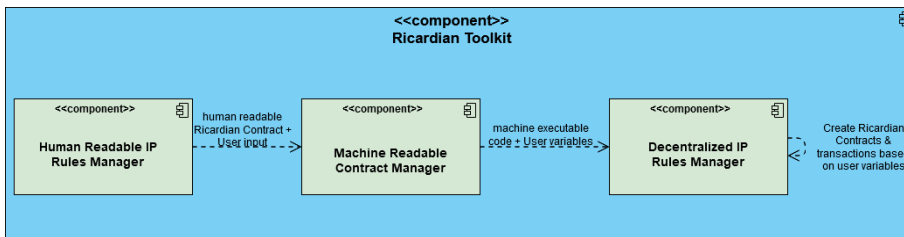


Figure 3-1: Component diagram of the IPR Authoring Tool.

The following Table 3-2 describes the component diagram of the IPR Authoring Toolkit:

Table 3-2: Subcomponent description.

Sub-components	Description
Human Readable IP Rules Manager	The ‘Human Readable IP Rules Manager’ which will enable the Users to compile the human readable Ricardian contracts by adding input to a dedicated User Interface
Machine-Readable Contract Manager	The Machine-Readable Contract Manager that will translate the human readable Ricardian contracts into machine executable code
Decentralised IP Rules Manager’	The ‘Decentralised IP Rules Manager’ which will instantiate the Ricardian contracts deployed to the Blockchain network (using blockchain/DLT) and will submit transactions that will call the functions of these Smart Contracts using the User input added to the first component as parameter values for the calls.

On figure 3-2 is shown the IPR Authoring Tool service sequence diagram.

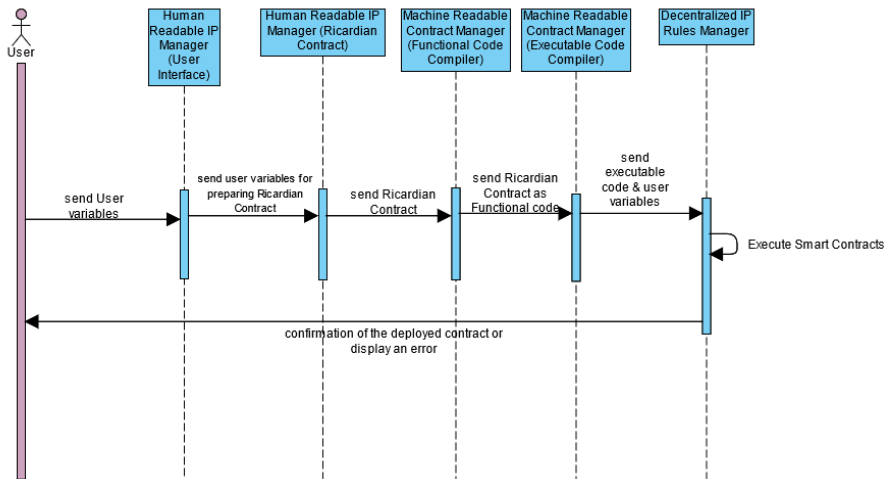


Figure 3-2: IPR Authoring Tool - Ricardian Toolkit service sequence diagram.

3.2. External Architecture

According to the D4.2 “OpIS Architecture and Design for Social Manufacturing 2”, the interconnection of the IPR Authoring Tool with the other main components of the OpIS platform is illustrated in the figure below:

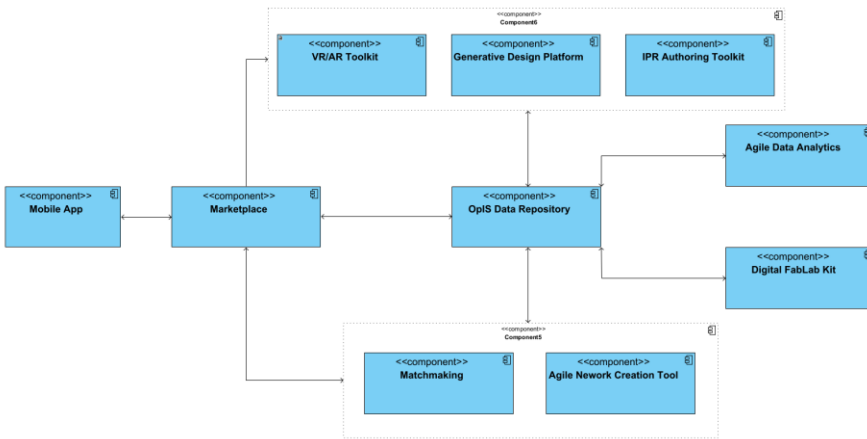


Figure 3-3: External architecture - Interconnection between the main components of the OpIS platform.¹

¹ D4.1 OpIS Architecture and Design for Social Manufacturing

As the figure indicates, the main component of the iPRODUCE OpIS platform is the OpIS Data Repository, which connect all the other components under the umbrella of a REST API backend. Specifically, this constitutes the connector plugin which interconnect all the components of the OpIS platform by providing a data exchanging method. The OpIS data repository is responsible for the navigation of the user inside the OpIS platform. The Marketplace provides accessibility to the products, teams, activities and users' data. Through the Matchmaking and Agile Network Creation Tools, user is able to search for the suitable partners. These tools aim at fostering the creation of collaborative networks and empowering them to jointly address specific business opportunities. Regarding the AR/VR Toolkit, this is a real time social manufacturing space for co-creation process under Augmented and Virtual environments. Similarly, the Generative Design Platform is a digital toolkit which is used so that user will participate to the co-creation process. Using a mobile application which is developing to obtain Voice of Customer feedback, users can actively solicit input (such as surveys) about new ideas. Additional, iPRODUCE defines a number of IPR and transaction management strategies that can be applied to facilitate the formation and operation of multi-party ad hoc teams, which will undertake collaborative manufacturing missions (using the IPR Authoring Tool – Ricardian Toolkit). From the visualization perspective, the Agile Data Analytics and Visualization Suite is a tool which focus on analysis and storage of Big Data (Analysis of the feedback from the cMDFs potential users, trainees, user preferences about the products/services, market trends, datasheet and technical manual of equipment). Finally, the Digital FabLab Kit constitutes a toolkit which is responsible for digitizing existing knowledge and common practices in makerspaces. It mainly addresses two aspects: (1) Digitization of training content and (2) digitization of production processes.

It is important to be mentioned in which time the user interacts with the IPR Authoring Tool. The Landing page constitutes the starting point of a user in the OpIS platform. Through the landing page, the user will be able to navigate in all the components of the iPRODUCE platform. While accessing the marketplace, the user has the capability to publish his own work and discover potential collaborations, irrespective of their role as a maker or consumer, all within the same platform. When a user is interested to create a new team in order to trigger the co-creation process, (s)he is able to use the Matchmaking and Agile Network Creation Tools so that a team will be defined by inviting SMEs / makerspaces / FabLabs. Each team which is created can co-design a lot of products based on the team's preferences. In case the co-creation process has been started, all the involved users can modify a product by using the relevant tools (Generative Design Platform, AR/VR Tool). Each one of these tools has its properties and functionalities which are described in the corresponding deliverables (D5.3, D5.4). In the time of the submission of a new product, all the involved users have been informed about the new contract which defines their collaboration. A user will be able to access the IPR Authoring Toolkit and navigate to his/her pending / accepted / rejected contracts by clicking "Review Contract" button component inside Marketplace or by navigating through the Landing Pahe to the corresponding tool. In Section 5 IPR Authoring Tool Implementation, the User Interface of the IPR Authoring Tool is presented as well as all the functionalities of this tool are analyzed.

Comentado [TK1]: According to a previous decision of the project this is not accurate

Comentado [TK2]: Make sure that what you say is accurate – you may be asked to show this

4. Blockchain technology

Distributed ledger technology (DLT) has become a hot topic in a variety of industries in recent years, sparked by early interest in blockchain and then expanding into a more in-depth discussion of the underlying technology. DLT offers higher speed and efficiency, reimagined business models, more transparency, and higher confidence along the transaction value chain. In this section, a benchmarking study is presented by providing details about (1) the existed blockchain Frameworks, (2) the functionalities and the components of the blockchain contracts, (3) on how the Ricardian and the Smart contracts were created by using the initial blockchain contracts and finally (4) the analysis from the legal perspective. The results of this study are presented inside the following sections and conclude to these:

- The EOS system is more suitable for the IPR and transaction methods which are introduced inside the iPRODUCE project, in contrast of the Ethereum
- The Ricardian Contracts have a lot of benefits in order to be used as transaction method inside the iPRODUCE platform, instead of the Smart Contracts.

4.1. Frameworks

The blockchain is a tried-and-true technology that can be used in any situation. Blockchain-based applications are gaining popularity across the board. An evaluation of the two most useful blockchain frameworks can be found below.

4.1.1. Ethereum

Ethereum is the most frequently used development platform for smart contracts (section 4.2.1) and it may be thought of as a transaction-based state machine that starts with a starting state and incrementally performs transactions to turn it into certain end states. These are the final states that have been regarded as the canonical “version” in the universe of Ethereum.² Unlike Bitcoin’s UTXO model, the idea of accounts is introduced in Ethereum. There are two types of accounts: 1) Externally Owned Accounts (EOAs) and 2) Contract Accounts. The difference is that the former is controlled by private keys with no associated code, but the latter is controlled by contract code that has an associated code.

An EOA is the only way for users to start a transaction. Binary data (payload) and Ether can both be included in the transaction. A smart contract is generated when the recipient of a transaction is zero-account 0. The account is activated and the corresponding code is executed in the local EVM if the receiver is a contract account (the payload is provided as input data). The transaction is sent to the blockchain network where it is verified by miners, as shown in the figure below.

² Ethereum Yellow Paper. (2018). [Online]. Available: <https://ethereum.github.io/yellowpaper/paper.pdf>

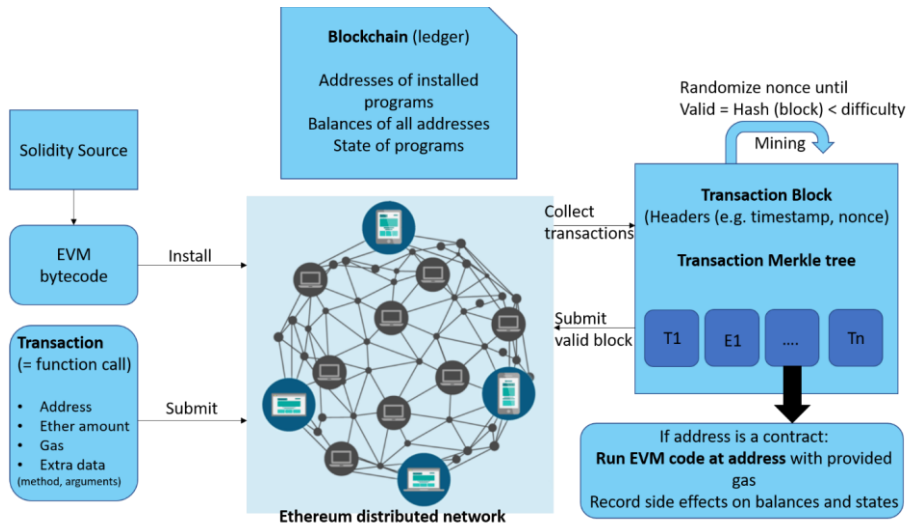


Figure 4-1: Overview of workflow in the Ethereum network.³

To avoid network abuse issues and circumvent the inevitable problems that arise from Turing completeness, all programmable computations (e.g., creating contracts, executing message calls, using and accessing account storage, and performing virtual machine operations) are chargeable in Ethereum – a reward for miners who donate their computational resources. The gas unit is used to measure the amount of feed necessary for computations.

4.1.2. EOS

EOSIO is a decentralized enterprise system that runs DApps (Decentralized Applications) at industrial scale – software that relies on the EOSIO blockchain to cryptographically record transactions. The transfer of the EOSIO currency token is the most popular transaction, called EOS. EOSIO, unlike Bitcoin and Ethereum, is a DPoS-based (Delegated Proof-of-Stake) system that can scale to millions of transactions per second, making it an appealing alternative for new DApp developers.

Within EOSIO, there are four fundamental ideas to grasp. To engage with the EOSIO blockchain, an entity must first create and account. A transaction, which consists of one or more actions to be done, can then be used to trigger a smart contract using this unique identity. A transfer of an EOS token from one account holder to another, for example, is an activity. Accounts that intend to invoke a contract must first delegate proper permissions to it, granting it the ability to act on their behalf. The rest of this section delves deeper into each of these four topics.

4.1.2.1. EOSIO's Smart Contract and Transactions

The official language for DApp developers to construct smart contracts on EOSIO is C++. Specifically, the source code of Smart Contract is compiled down to WebAssembly bytecode. When the bytecode is called, it is performed in EOSIO's Wasm VM, which results in transactions being recorded on the

³ K. Bhargavan et al., "Formal verification of smart contracts: Short paper", in Proc. ACM Workshop Program. Lang. Anal. Security (PLAS), Vienna, Austria, Oct. 2016, pp. 91-96

blockchain, such as transferring EOS. An action is a base32 encoded 64-bit integer that may be used to represent a single operation and is a fundamental aspect of smart contract communication. There are two sorts of actions: external and inline. An external action is when the user initiates an action from the outside, while an inline action is when the user invokes another action from within a smart contract (the same or external).

One or more actions can be included in each transaction. There are two sorts of actions that can be sent for communication in EOSIO: inline actions, which execute activities in the same transaction as the original action, and deferred actions, which execute operations in a future (delayed) transaction. .If deferred actions are planned, they will always run asynchronously ,inline actions, on the other hand, are guaranteed to run in a synchronous manner. If an inline action fails or throws an exception, the transaction is rolled back.

4.1.2.2. EOSIO's Account Management

In EOSIO, accounts are the entities that can conduct transactions. A human-readable name (up to 12 characters) is recorded in the blockchain as an EOSIO account. Accounts are permission structures which can define contract senders and receivers in practice. Accounts can also grant contracts permissions and be set up to allow a single user or a group of users to submit or push any lawful transaction to the blockchain.

It is important to mention that an EOSIO account is not the same as (and more complicated) an Ethereum account. First and foremost, accounts are hierarchical in nature and can only be established from an existing account, resulting in a tree structure. This means that the resources required to construct new accounts must be taken from current accounts (with the exception of eosio, the root of the tree, which was produced by the blockchain system when the mainnet was deployed). This will unavoidably deplete system resources (RAM) and therefore EOSIO account creation is not free.

An EOSIO account's permissions are used to authorize actions and transactions for other accounts.⁴ Specifically, the account can assign specific actions to public/private keys, and a given key pair can only perform the corresponding action. An EOSIO account comes with two public keys by default: the owner key (which identifies the account's owner) and the active key (which identifies the account's current state) (which grants access to activities with the account). These two keys allow you to manage accounts with two native named permissions: owner and active. EOSIO additionally provides for customized named permissions for advanced users in addition to the native permissions. EOSIO provides you customizable named permissions in addition to native permissions for enhanced account management.

4.1.3. EOS in comparison with Ethereum within iPRODUCE

As there are several different platforms out there that offer somewhat similar services, such as the ones presented in this study, there needs to be some points of differentiation and categorization. Firstly, it is crucial to be mentioned that inside the OpIS platform, iPRODUCE identifies a set of IPR and transaction management methods that can be utilized to assist the formation and operation of ad hoc multi-sided teams that take on collaborative manufacturing activities under supervision. The project will use Ricardian Contracts as the DoA indicates and form the following section proves as the best choice, in order to ensure that such techniques be implemented in a 'seamless' manner.

⁴ Y.Huang, H. Wang, Understanding (Mis)behaviour on the EOSIO Blockchain Proc. ACM Meas. Anal. Comput. Syst., Vol. 4, No. 2, Article 37. Publication date: June 2020.

Comparison EOS to Ethereum, the EOS system exceeds that the capabilities of Ethereum in some ways. Ethereum was developed and designed in a way where the system can only manage 15-20 transactions per second. On the other hand, EOS was created to address the demand for large-scale decentralized applications. This means that the EOS system is designed to be scalable, fast, and flexible. On other blockchain networks, the lack of these features can act as a bottleneck, necessitating a unique solution in order to grow correctly.

Parallel execution and an asynchronous network communication mechanism allow the EOS system to reach this level of scalability and flexibility. In order to achieve more efficiency, the system also isolates numerous modules, such as the authentication and execution processes.⁵

By technical perspective, the EOS system (as its benefits indicate) is deemed suitable to be used inside the iPRODUCE project in order to execute transactions using its blockchain technology.

4.2. Blockchain contracts

In this section, the blockchain contracts are introduced. It is deemed necessary all the components of the blockchain contracts to be analysed in order to be known their architecture and their structure. In the following subsections, the meanings of the Smart and Ricardian Contracts are presented by providing a detailed analysis as well as a comparison methodology. Specifically, through this study, it is critical to end up the importance of the Ricardian Contracts under the umbrella of iPRODUCE project.

Blockchains are 'tamper evident' and 'tamper resistant' digital ledgers implemented in a distributed manner (i.e., without a central repository) and usually without a central authority (i.e., a bank, company or government). At their most basic level, they allow a group of people to keep track of their transactions in a shared ledger within that community, so that in the normal operation of the blockchain network, no transaction, once published can be altered. The blockchain concept was integrated with other technologies and computer principles in 2008 to create modern cryptocurrencies: electronic cash protected by cryptographic processes rather than a central repository or authority.

With the creation of the Bitcoin network in 2009, the first of several modern cryptocurrencies, this technology became well known. In Bitcoin, and similar systems, the transfer of digital information representing electronic money takes place in a distributed system. Users of Bitcoin can digitally sign and transfer their rights to this information to another user, and the Bitcoin blockchain records this transfer in a public way so that other network members may independently verify the transactions' legitimacy. A distributed set of people maintains and manages the Bitcoin blockchain independently. This, along with cryptographic mechanisms, makes the blockchain resistant to attempts to alter the ledger after the fact (modify blocks or forge transactions). Blockchain technology has enabled the development of many cryptocurrency systems such as Bitcoin and Ethereum. For this reason, blockchain technology is often seen as tied to Bitcoin or possibly cryptocurrency solutions in general. However, the technology is available for a broader selection of applications and is being explored for a variety of industries.

The complexity of blockchain technology, as well as its reliance on cryptographic primitives and distributed systems, makes it difficult to comprehend. However, each component can be easily described and used as a building block to understand the larger complex system. Blockchains can be informally defined as:

⁵ <https://academy.ivanontech.com/blog/smart-contract-platforms-eos-vs-ethereum-vs-rsk-vs-cardano>

Blockchains are block-based distributed digital ledgers comprising cryptographically signed transactions. After validation and a consensus decision, each block is cryptographically linked to the one before it (making it tamper-proof). It becomes more difficult to change order blocks as new blocks are introduced (which increases tamper resistance). New blocks are propagated across the network's copies of the ledger, and any conflicts between blocks are automatically resolved according to specified rules. Distributed Ledger Technology (DLT) is a decentralized database that is administered by various people.

Blockchain is a type of DLT in which transactions are recorded with an immutable cryptographic signature called a hash.

The properties of Distributed Ledger Technology (DLT) are presented according the following scheme:

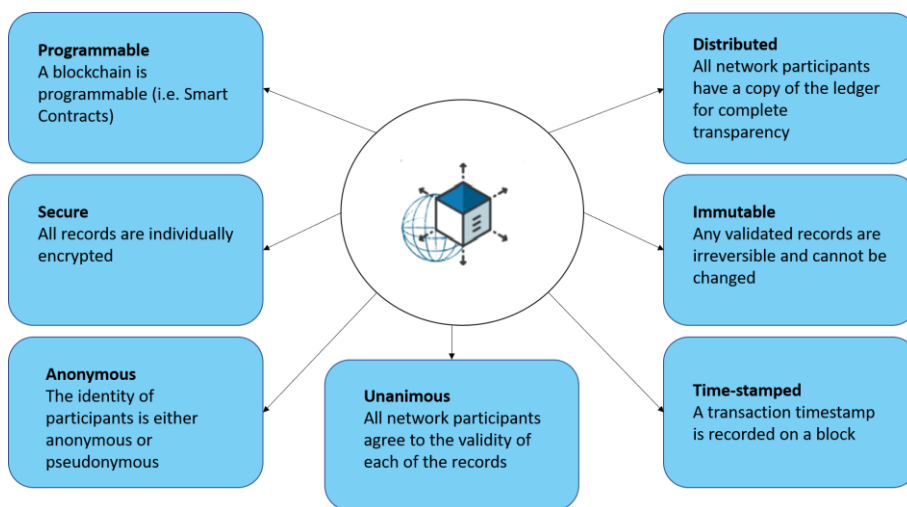


Figure 4-2: The properties of Distributed Ledger Technology (DLT).⁶

Although blockchain technology appears to be complex, it may be simplified by looking at each component separately. From a high level overview, blockchain technology uses well-known informatics mechanisms and cryptographic primitives (cryptographic hash functions, digital signatures, asymmetric cryptography) mixed with data storage concepts (such as append-only ledgers). In the following points, it is discussed each major component: cryptographic hash functions, transactions, asymmetric-key cryptography, addresses, ledgers, blocks and how blocks are chained together.

Cryptographic Hash Functions

The usage of cryptographic hash functions for many processes is a key aspect of blockchain technology. Hashing is a way of applying a cryptographic hash function to data in order to produce a somewhat unique output (called a message digest, or simply digest) for nearly any size input (e.g., a file, text, or image). It allows individuals to independently take input data, hash that data, and get the same result-proving that the data has not been altered. Even the tiniest modification in the input (such as a single bit) results in a completely different output digest.

⁶ <https://en.wikipedia.org/wiki/Blockchain>

Cryptographic hash functions have these important security properties:

1. They are preimage resistant.
2. They are resistant to a second preimage.
3. They are collision resistant.

One particular cryptographic hash function used in many blockchain implementations is the Secure Hash Algorithm (SHA) with an output size of 256 bits (SHA-256). Many computers support this algorithm in hardware, allowing it to be computed quickly. SHA-256 has an output of 32 bytes (1 byte = 8 bits, 32 bytes = 256 bits), which is generally displayed as a 64-character hexadecimal string.

Transactions

A transaction represents an interaction between parties. For example, in cryptocurrencies, a transaction represents a transfer of cryptocurrency between users of the blockchain network. In business-to-business scenarios, a transaction could be a way of recording activity that take place on digital or physical assets. A blockchain block can have zero or more transactions in it. For some blockchain implementations, preserving the security of the blockchain network requires a steady supply of new blocks (even if there are no transactions); a constant supply of new blocks being published prevents malicious users from ever “catching up” and producing a longer, modified blockchain.

The data that makes up a transaction may be different for each blockchain implementation, but the mechanism for the transaction is largely the same. Information is sent to the blockchain network by a user of the blockchain network. The information sent may include the sender’s address (or other relevant identifier), the sender’s public key, a digital signature, transaction inputs, and transaction outputs.

A single cryptocurrency transaction usually necessitates at least the following information, but may include more:

- **Inputs** – The inputs are typically a list of the digital assets to be transferred. The source of the digital asset is referenced in a transaction — either the previous transaction when it was provided to the sender, or the origination event in the case of fresh digital assets.
- **Outputs** – The outputs are usually the accounts that are the recipients of the digital assets, along with the set of digital assets they will receive.

Asymmetric-Key Cryptography

Blockchain technology uses asymmetric-key cryptography (also called public key cryptography). Asymmetric-key cryptography uses a pair of keys: a public key and a private key that are mathematically linked. The public key is made public without compromising the process’ security, but the private key must stay secret if the data’s cryptographic protection is to be maintained. Even if there is a relationship between the two keys, the private key cannot be efficiently determined from knowledge of the public key. One can encrypt with a private key and then decrypt with the public key. Alternatively, one can encrypt with a public key and then decrypt with a private key.

By offering a way to check the integrity and authenticity of transactions while allowing the transactions to remain public, asymmetric-key cryptography establishes a trust relationship between users who do not know or trust each other. To achieve this, the transactions are ‘digitally signed’. This means that a private key is used to encrypt a transaction in such a way that anyone with the public key can decrypt it.

The following points will introduce the use of asymmetric-key cryptography in many blockchain networks:

- Private keys are used to digitally sign transactions.
- Public keys are used to derive addresses.
- Public keys are used to verify signatures generated with private keys.
- Asymmetric-key cryptography makes it possible to verify that the user transferring value to another user is in possession of the private key that can sign the transaction.

Addresses and Address Derivation

Some blockchain networks utilize an address, which is a short alphanumeric string obtained from a cryptographic hash function and some additional data from the blockchain network user's public key (e.g., version number, checksums). Most blockchain implementations use addresses as the "to" and "from" endpoints in a transaction. Addresses are shorter than public keys and are not secret. One method of generating an address is to create a public key, apply a cryptographic hash function to it, and convert the hash to text:

Public key → cryptographic hash function → address

Each blockchain implementation may implement a different method for deriving an address. For permissionless blockchain networks, that allow anonymous account creation, a blockchain network user can generate as many asymmetric-key pairs, and thus addresses as desired, allowing varying degrees of pseudo-anonymity. Addresses can act as a public identifier for a user on a blockchain network, and often an address is converted to a QR code for ease of use with mobile devices.

Ledgers

A ledger is a list of all the transactions that have occurred. Pen and paper ledgers have been used to record the transaction of commodities and services throughout history. In modern times, ledgers are stored digitally, frequently in huge databases administered on behalf of a community of users by a centralized, trusted third party (i.e., the owner of the ledger). These centralized ownership ledgers can be deployed either centralized or distributed (i.e., single server or coordinating cluster of servers).

Blocks

Software is used by users of the blockchain network to submit candidate transactions to the blockchain network (desktop applications, smartphone applications, digital wallets, web services, etc.). These transactions are sent to a node or nodes within the blockchain network by the software. The selected nodes can be both non-publishing full nodes and publishing nodes. The submitted transactions are subsequently propagated to the rest of the network's nodes, but this does not guarantee that the transaction will be included to the blockchain.. In many blockchain implementations, once a pending transaction has been propagated to nodes, it must wait in a queue until it is added to the blockchain by a publishing node.

When a publishing node publishes a block, transactions are added to the blockchain. A block is made up of two parts: a block header and block data. The metadata for that block is contained in the block header. A list of approved and legitimate transactions made to the blockchain network is contained in the block data. Validity and authenticity are ensured by ensuring that the transactions are properly formed and that each of the digital asset providers in each transaction (specified in the transaction's "input" values) has cryptographically signed the transaction. This verifies that the digital asset providers for a transaction had access to the private key with which to sign the available digital assets. Other full nodes verify the legitimacy and authenticity of all transactions in a published block before accepting it, and reject it if it contains invalid transactions.

Chaining Blocks

The blockchain is formed by a chain of blocks, each of which contains the hash digest of the previous block's header. If a previously published block was updated, the hash value would be changed. This in turn causes all subsequent blocks to also have different hashes as well, since they contain the hash of the previous block. In this way it is possible to easily detect and reject modified blocks. The figure below shows a generic chain of blocks.⁷

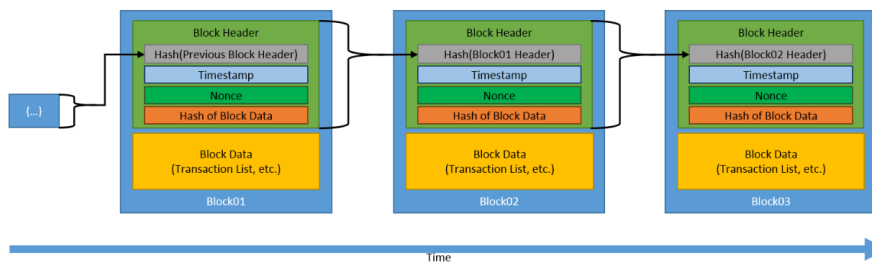


Figure 4-3: Generic Chain of Blocks

4.2.1. Smart Contracts

Nick Szabo, who coined the phrase smart contract in 1994, defines it as "a computerized transaction protocol that performs the provisions of a contract." Smart contract design aims to meet common contract terms (such as payment terms, liens, secrecy, and even enforcement), reduce malicious and unintentional exceptions, and eliminate the need for trusted intermediates. Smart contracts extend and leverage blockchain technology. A smart contract is a collection of code and data (also known as functions and state) that is put on the blockchain network via cryptographically signed transactions (e.g., Ethereum's smart contract, Hyperledger Fabric's chaincode). The smart contract is executed by nodes in the blockchain network; all nodes that execute the smart contract must get the same outcomes from it, and the results are recorded on the blockchain.

Users of the Blockchain network can create transactions that send data to public functions offered by a smart contract. The smart contract executes the appropriate method using the data provided by the user to perform a service. Also, because the code resides on the blockchain, it is tamper-proof and can therefore be used as a trusted third party (among other purposes). A smart contract can perform calculations, store information, expose properties to reflect a publicly available state, and automatically send money to other accounts if necessary. It does not even necessarily have to perform a financial function.

Smart contract code can represent a multi-party transaction, typically in the context of a business process. In a multi-party scenario, the advantage is that it can provide provable data and transparency that promotes trust, provides insights that enable better business decisions, reduces the costs of reconciliation that exists in traditional business-to-business applications, and reduces the time to complete a transaction.

⁷ D. Yaga, P. Mell, N. Roby, K. Scarfone, Blockchain Technology Overview, National Institute of Standards and Technology, 2018

Smart contracts must be deterministic, meaning that given an input they must always produce the same output based on that input. Also, all nodes executing the smart contract must agree on the new state that will be reached after execution. To achieve this, smart contracts cannot process data that it is not passed directly to them (e.g., smart contracts cannot obtain data from web services from the smart contract – it would have to be passed as a parameter). Any smart contract that uses data from outside the context of its own system is called an 'Oracle'.

When publishing new blocks on many blockchain implementations, the publishing nodes run the smart contract code at the same time. Some blockchain implementations include publishing nodes that do not run smart contract code but rather validate the outcomes of the nodes that do. The user making a transaction to a smart contract on permissionless blockchain networks with smart contracts (such as Ethereum) will have to pay for the cost of the code execution. Based on the intricacy of the code, there is a limit on how much execution time can be consumed by a call to a smart contract. When this limit is reached, the transaction is discarded and the executions are halted. This technique not only pays the publishers for running the smart contract code, but also discourages malicious users from deploying and then accessing smart contracts that will perform a denial of service on the publishing nodes by using all available resources (e.g., using infinite loops).⁸

The operational mechanism of smart contracts is shown in Figure below.

⁸ D. Yaga, P. Mell, N. Roby, K. Scarfone, Blockchain Technology Overview, National Institute of Standards and Technology, 2018

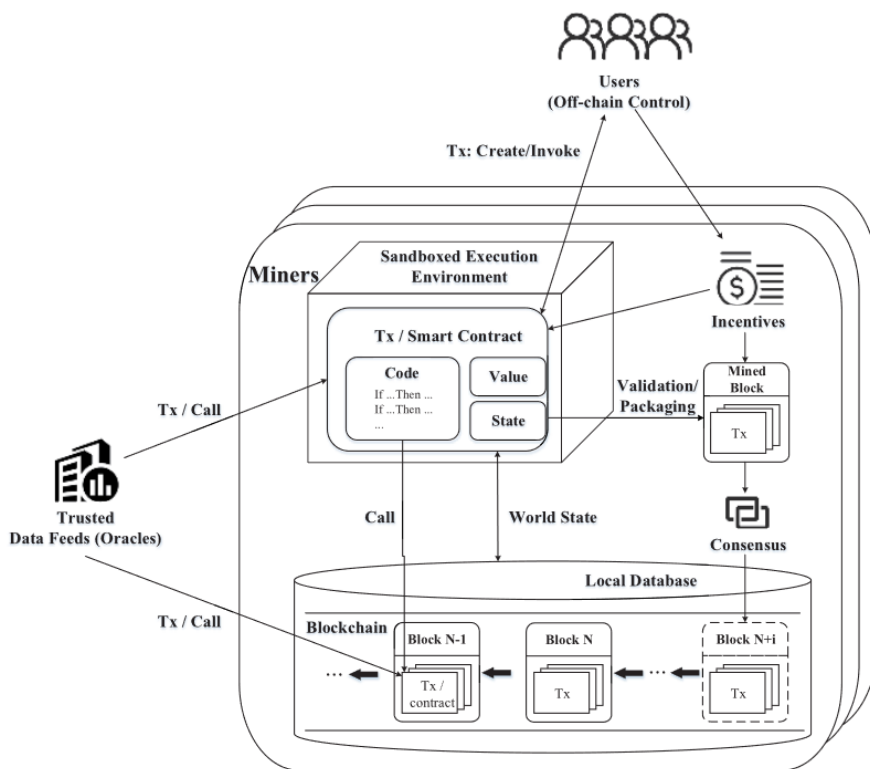


Figure 4-4: Operational mechanism of smart contract.

Smart contracts usually have two attributes: 1) value and 2) state. The activation conditions of the contract terms and the corresponding response actions are shown by activation condition statements such as “If-Then” statements. All parties agree on and sign the smart contract, which is then sent to the blockchain network as a transaction. The transaction is then transmitted through the P2P network, verified by the miner, and stored in a specific block on the blockchain. The contract creator gets the returned parameters (for example, the contract address) and the user can call the contract by submitting a transaction. More specifically, after receiving the contract creation or call transaction, the miner creates the contract or executes the contract code in its sandbox execution environment (EVM). Based on input from a trusted data source (also known as Oracle) and system status, the contract determines whether the current plan meets the trigger conditions. If the conditions are met, the response action will be strictly executed. After verifying the transaction, pack it into a new block. After the entire network reaches a consensus, the new block will be linked to the blockchain.

4.2.2. Ricardian Contracts

This type of contracts is derived from the work of financial cryptography expert Ian Grigg, which was completed in the mid-90s as a contributions to Ricard (an asset transfer system established in 1995-

1996). The design system and style are named after David Ricardo in order to recognize his formative contribution to international trade theory.

The Ricardian contract, according to its author, is a "cryptographically signed and validated digital contract that establishes the terms and circumstances of an interaction between two or more peers." Importantly it is both human and machine readable and digitally signed".

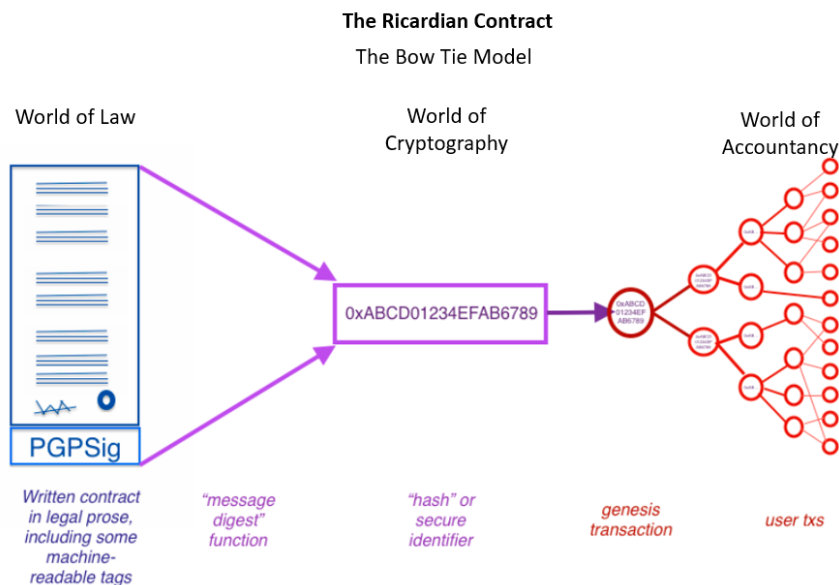


Figure 4-5: Overview of Ricardian Contracts.⁹

A Ricardian contract is a legally binding agreement that is digitally linked to a specific object or value. Its implementations could differ. A Ricardian contract puts all the information of the legal document on a format that can be executed by software. In this way, it's both a legal agreement between the parties and a protocol that combines an agreement with cryptographic identification to provide a high level of security.

The main characteristics of this type of contract are the following:

- Human parsable;
- Document is printable;
- Program parsable;
- Every form (displayed, printed, and parsed) is clearly equivalent;
- Signed by issuer;
- Can be identified securely, where security means that changing the link between a reference and the contract is impractical;

⁹ <https://medium.com/tonetwork/ricardian-contracts-legally-binding-agreements-on-the-blockchain-4c103f120707>

4.2.2.1. Difference Between Smart and Ricardian Contracts

There is a slight misunderstanding as to whether or not it is correct to equate a smart contract with a Ricardian. Although they share a number of similarities, they are distinct concepts. Sure, it is possible to implement a Ricardian contract as a smart contract, but not every Ricardian contract is a smart contract. Accordingly, not every smart contract is a Ricardian contract.

Smart contracts are a sort of digital contract that can be performed automatically after it has been agreed upon. A Ricardian contract, on the other hand, adheres to the contract model, which records a contract's "intentions" and "actions" regardless of whether the contract has been implemented or not. Ricardian contracts can relate to code by employing hashes that refer to external documents. In the future, there will be more interaction between this type of contract and smart contracts, and transactions will likely be conducted based on various hybrid forms.¹⁰ In the following table, a comparison analysis of the Smart and Ricardian Contract is presented.

Table 4-1: Smart versus Ricardian Contracts.

	Smart Contracts	Ricardian Contract
Purpose	Execute the term of an agreement	As a legal document, record the conditions of an agreement
Flow	Actions on blockchain-based applications can be automated	It can also automate blockchain-based application activities
Validity	It is not a legally binding document	It is a legally binding document or agreement
Versatility	They can not be Ricardian Contracts	Any Ricardian Contract can be a Smart Contract as well
Readability	Smart Contracts are machine-readable but not necessarily human-readable	Ricardian Contracts are both machine-readable as well as human-readable

As the table indicates, Ricardian Contracts are superior to Smart Contracts. Firstly, the purpose regarding the Ricardian Contracts includes the legal perspective. From iPRODUCE point of view, it is necessary to present a legal document in terms of an agreement. Regarding the validity, it is important to be mentioned that Ricardian Contract constitutes a legally binding document or agreement. The IPR and transaction mechanisms of the iPRODUCE project require that an agreement should be both machine-readable as well as human-readable, so that users and machine can read and edit an agreement in terms of Ricardian Contract.

4.3. Legal Analysis

In the above sections, the blockchain technology was introduced. In this section, the legal perspective of the blockchain technology is explained. To illustrate this point, the focus is on the various ways in which blockchain technology is being used to create structures that resemble existing legal concepts, including contracts, companies and securities. In what follows, it is analyzed how existing law will apply to these new blockchain-based structures.

¹⁰ <https://www.elinext.com/industries/financial/trends/smart-vs-ricardian-contracts/>

4.3.1. Smart Contracts and Contract Law

The term smart contract is arguably misleading, as it refers to automatically executed computer code (Section 4.2.1). This raises the question of whether a smart contract qualifies as a legal contract.

A legal contract is usually defined as a legally enforceable agreement or promise. It is typically entered into by voluntary offer and acceptance and, in common law jurisdictions, by considerations of: the value offered by each party. Many types of contracts can be formed in any way: orally, in writing, or by actions, such as agreeing to terms in electronic media by clicking. However, the law in some countries may require that certain types of contracts be recorded in a particular form or concluded in a particular manner, for example real estate transactions must be recorded in writing and notarized by a notary public.

Since smart contracts self-execute predetermined code, it could be argued that they cannot be broken. The smart and the Ricardian contract will always do exactly what their code says. However, as noted above, it is likely that the legal contract between the parties will contain obligations that go beyond the code itself and are based on other communications. If that is the case, not all obligations can be fully and correctly captured by the underlying smart contract. As a result, there may be a mismatch between what the parties have agreed to do and what the smart contract code executes, leading to non-performance. Smart contracts are, by their nature, limited to the terms of the contract that can be specified in computer – readable code, and further constrained by any limitations imposed by the blockchain system in which the contract operates. As a result, they are unable to capture the real-world complexity of all but the simplest transactions.

As mentioned in Section 4, the code of a smart contract is executed by all nodes in the network and is publicly visible. However, many contracts contain commercially sensitive information. Therefore, smart contracts are unsuitable for contracts that contain information that would otherwise be subject to a non-disclosure agreement or confidentiality clause. In the worst-case, disclosure of information through a smart contract could result in an unintended loss of trade secret protection or a breach of confidentiality. This is less of an issue on centralized platforms, where trusted nodes can control the visibility of the blockchain.¹¹

¹¹ J. Bacon, J.D. Michels, C. Millard, J. Singh, Blockchain Demystified: A technical and legal introduction to distributed and centralised ledgers, 25 Rich.J.L. & Tech, no. 1, 2018

5. IPR Authoring tool Implementation

5.1. Overview

The IPR Authoring Tool aims to become a visual authoring tool that defines a set of rules that are involved in the context of Design Thinking process stages for product co-creation process of the cMDF. The IPR Authoring Tool can be accessed through its dedicated domain name or through the Marketplace, which is the starting point of the OpIS platform. As described in section 3.2 that covers the External Architecture of the IPR Authoring Tool after a product is submitted, the involved users are notified regarding the new contract. The notification panel at the “Contracts” tab component of the Marketplace redirects the involved OpIS platform users to the IPR Authoring Tool to review the new contract that defines their collaboration.

The IPR Authoring Tool was created from scratch and the design and implementation, is based to the functional requirements that are extracted from the WP4. Specifically, the following functional requirements are depicted:

- User will document an initial set of entities that will collectively form a consortium of partners
- User will partition the product’s co-creation process into distinct weighted business flows.
- User will define a few basic governance policies
- User will provide a platform that will allow entities to propose, document and share their approaches on implementing individual business flow.
- The entities will be automatically accredited according to the weight of the business flow.

The purpose of the iPRODUCE IPR Authoring Tool is to provide a simple and comprehensive interface to define the terms of the Ricardian Contracts to extend the potential and accrued benefits of open collaboration for the co-creation products. The users of the OpIS data repository can use the IPR Authoring Tool in the following cases:

1. Accept Ricardian Contract
2. Edit Ricardian Contract
3. Retrieve Ricardian Contract
4. Reject Ricardian Contract
5. Create Ricardian Contract

To cover all the requirements, the architecture of the IPR Authoring Tool follows the client-server model. The front-end (client-side) contains the Web Interface and the back-end (server-side) manages the requests, the data and the inter-process communication with the back-end components of the platform. The front-end includes the following components: i) Dashboard page, ii) Pending Contracts, iii) Accepted Contracts, iv) Rejected Contracts, v) Deployed Contracts vi) Create Contract vii) Documentation page, viii) Contact Us page. The back-end includes a REST API layer and a KAFKA connector, in which all web services are defined and allow communication with the layer containing the core component functions of the backend. Specifically, the back-end communicates with the i) iPRODUCE block-chain platform ii) OpIS Data Repository. Through the Kafka connector and the OpIS data repository the connection, which encapsulates the users data and the product co-creation data, with the Marketplace can be established.

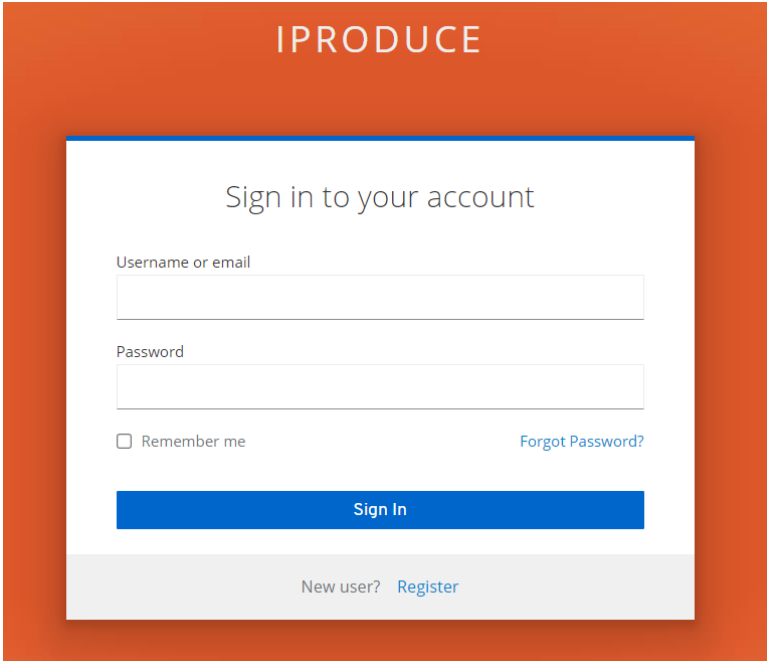
The functionalities that are presented, are supported by the current version of the IPR Authoring Tool. These functionalities will be enriched and enhanced, as the project work cycle progresses.

5.2. Front-end

In this section, we present the User Interface implementation by the current version of the IPR Authoring Tool. The initial page of the IPR Authoring Tool is the “Welcome Page” and the user can log-in to the IPR Authoring Tool providing the necessary credentials. In the next sub-sections the functionalities regarding the user interface are depicted,

5.2.1. Welcome Page

To streamline the login process and improve overall user experience, IPR Authoring Tool has implemented the SSO authoring mechanism of the OpIS data repository. By integrating OpIS data repository SSO authoring mechanism, users can log in once and gain access to multiple tools of the iProduce platform seamlessly, without the need for repeated authentication. The implementation of SSO authentication within our tool establishes a solid foundation for streamlined and secure user authentication. By leveraging SSO authoring mechanism of the OpIS data repository, our tool enhances user experience, improves security, and simplifies administration. The information required for the login can be seen in Figure 5-1.



IPRODUCE

Sign in to your account

Username or email

Password

Remember me [Forgot Password?](#)

Sign In

New user? [Register](#)

Figure 5-1: Login page

5.2.2. Dashboard Page

In the figure below, we present the overview of the page where user can see basic information concerning the status of the Ricardian contracts that is involved. More specifically, the user can check about its:

- Pending Contracts
- Accepted Contracts
- Rejected Contracts
- Deployed Contracts
- User Edited Contracts

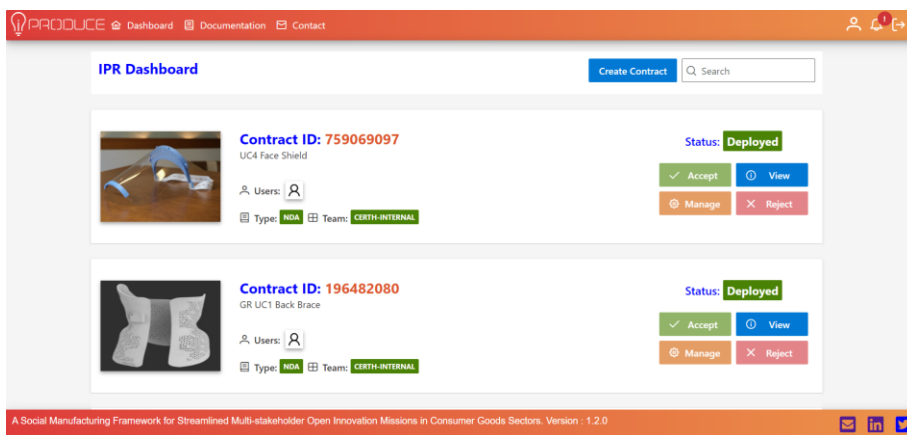


Figure 5-2: Dashboard Page

5.2.2.1. Pending Contracts

Within the IPR Authoring Tool's dashboard page, there exists a filtering component that categorizes and showcases Contracts based on their respective statuses. Contracts marked as "pending" necessitate unanimous acceptance from all users before they can be generated and deployed in the blockchain. Furthermore, users have the ability to modify or reject pending contracts. The associated component dedicated to pending contracts presents pertinent details such as the Contract ID, Product, Type, Team, and the involved Partners. Through the filtering mechanism, users can effortlessly access an accordion-style component, enabling them to delve into the specifics of the Ricardian Contract, as well as ascertain its current status. The figure below depicts the pending contracts component.

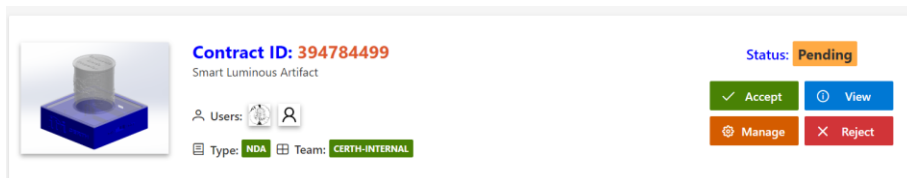


Figure 5-3: Pending Contracts

The “Pending Contracts” component in addition contains the button components i) “View Contract” ii) “Accept Contract” ii) “Manage Contract” iv) “Reject Contract”.

- **View Contract**

Clicking View Contract component a Dialog box with the Template Ricardian Contract created by the IPR Authoring Tool appears. Within component the user can communicate through video and chat with the involved partners. The Figure 5-4 depicts a multi-party template Ricardian Contract.

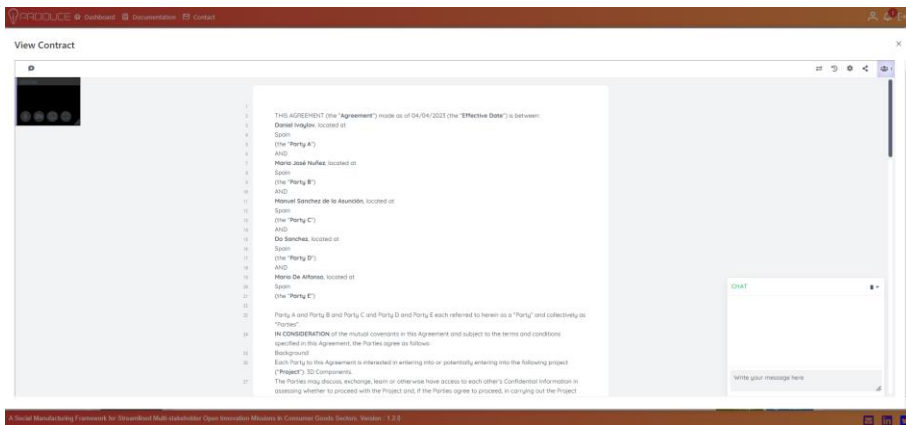


Figure 5-4: Template Ricardian Contract

- **Accept Contract**

By selecting the "Accept Contract" component, the user signifies their acceptance of the Ricardian Contract. If the acceptance process is successful, a pop-up notification message promptly appears to inform the user that the contract has been accepted successfully. Once the Ricardian Contract receives acceptance from all users involved, it is generated and deployed onto the blockchain.

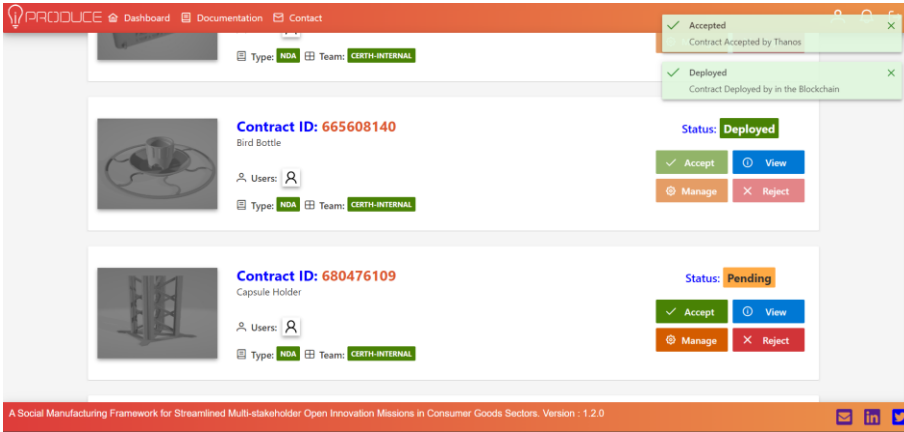


Figure 5-5: Accept Contract pop-up

- **Manage Contract**

By selecting the "Manage Contract" component, a Dialog box containing the Template Ricardian Contract, generated by the IPR Authoring Tool, is displayed. The user is given the ability to edit and update the contract as needed. Additionally, within this component, the user can engage in video and chat communication with the partners involved. It's important to note that if a user makes edits to the contract, all the involved users must reconfirm their acceptance of the updated contract.

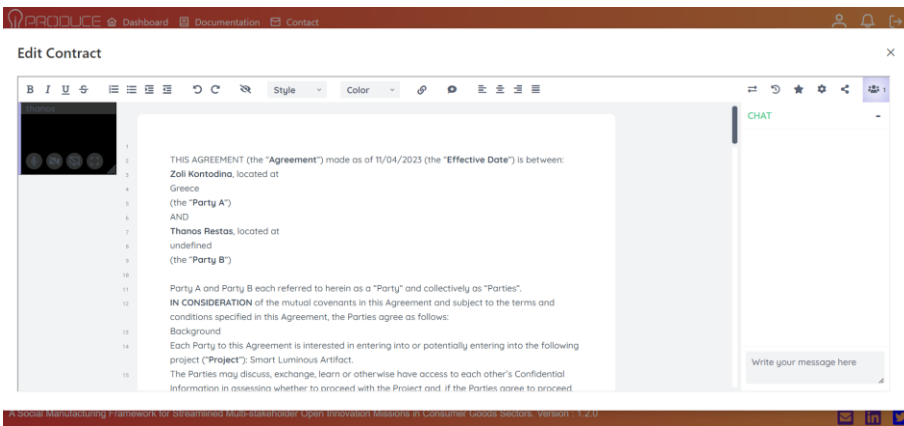


Figure 5-6: Edit Contract

In case of a successful edit, the status of the contract will be changed to User Edited.

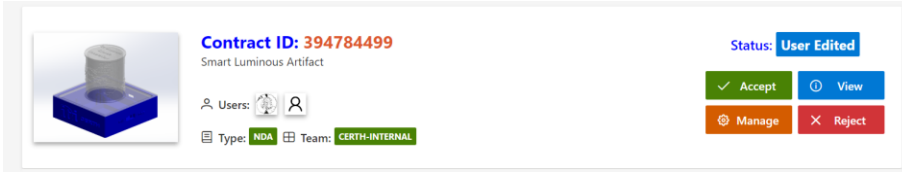


Figure 5-7: User Edited Contract

- **Reject Contract**

Clicking at Reject Contract component, a Dialog Box appears where the user can insert the reason of this Ricardian Contract rejection. In case of a successful contract rejection, a pop-up notification message will appear to inform the user that the contract has been rejected.

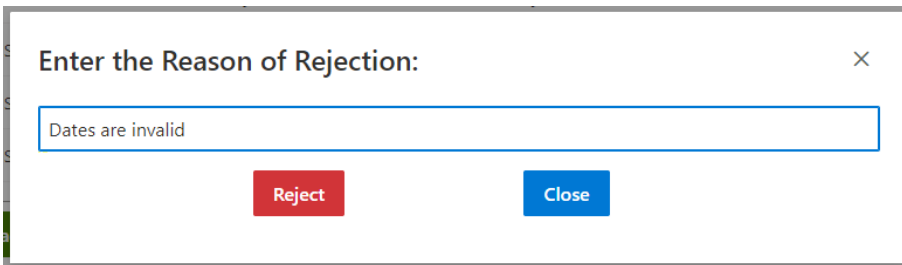


Figure 5-8: Reject Contract Dialog box

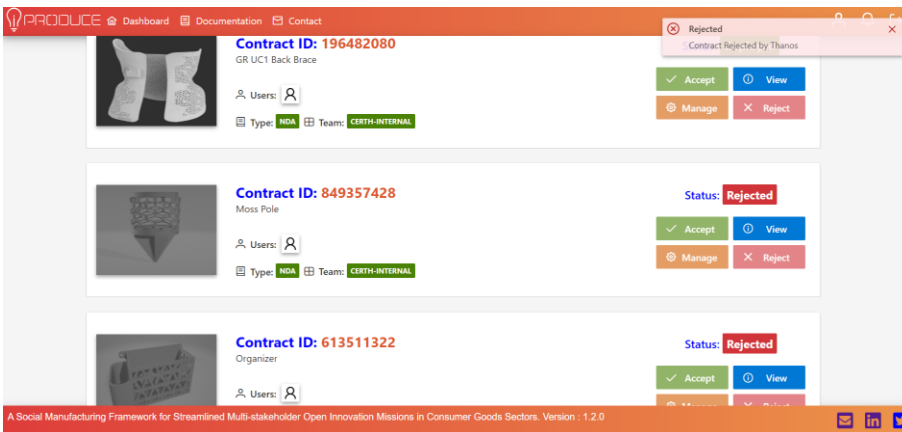


Figure 5-9: Reject Contract pop-up

5.2.2.2. Accepted Contracts

The dashboard page of the IPR Authoring Tool includes a filtering feature that organizes and displays Contracts based on their current status. Contracts labelled as "accepted" have been reviewed and approved by the user involved. The component related to accepted contracts presents important

information like the Contract ID, Product, Type, Team, and the involved Partners. By using the filtering feature, users can easily access an accordion-style component that allows them to explore the details of the Ricardian Contract and determine its current status. The image below illustrates the component displaying accepted contracts.

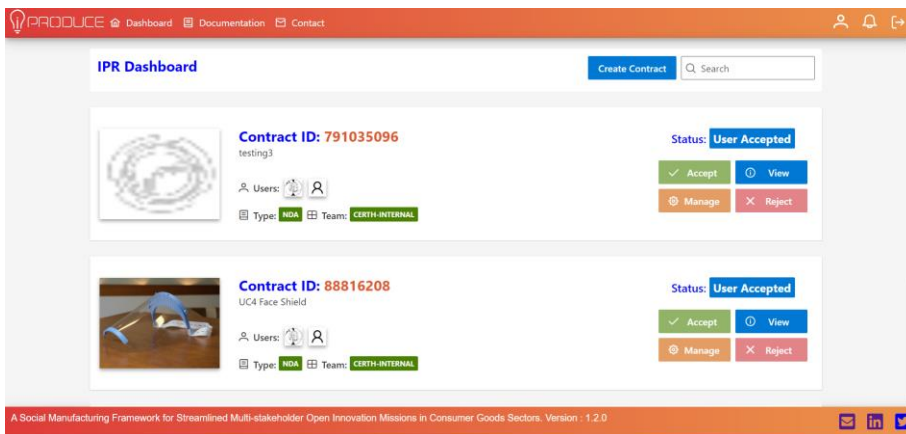


Figure 5-10: Accepted Contracts

The “Accepted Contracts” tab in addition contains “View Contract” button component. Clicking Show Contract component a Dialog box with the Accepted Ricardian Contract appears. Figure 5-11 depicts a Accepted Contract.

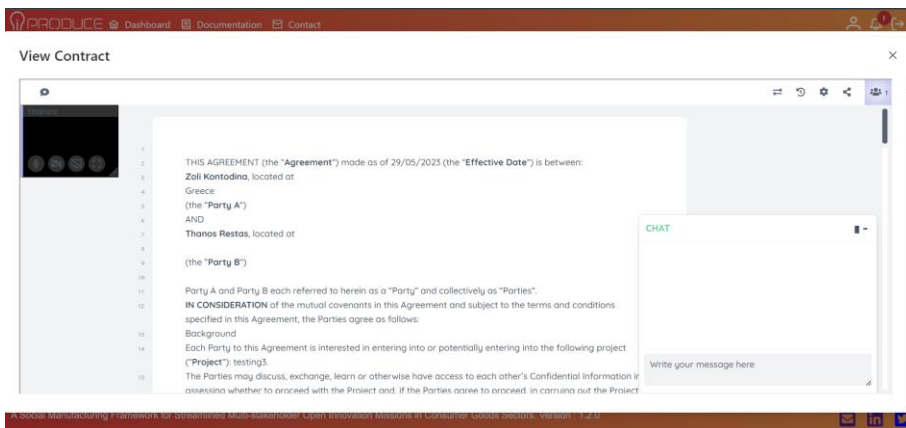


Figure 5-11: Accepted Contract Agreement

5.2.2.3. Rejected Contracts

The dashboard page of the IPR Authoring Tool includes a filtering feature that organizes and displays Contracts based on their current status. Contracts labeled as "rejected" have been rejected by at least

one of the users involved. The component related to accepted contracts presents important information like the Contract ID, Product, Type, Team, and the involved Partners. By using the filtering feature, users can easily access an accordion-style component that allows them to explore the details of the Ricardian Contract and determine its current status. The image below illustrates the component displaying rejected contracts.

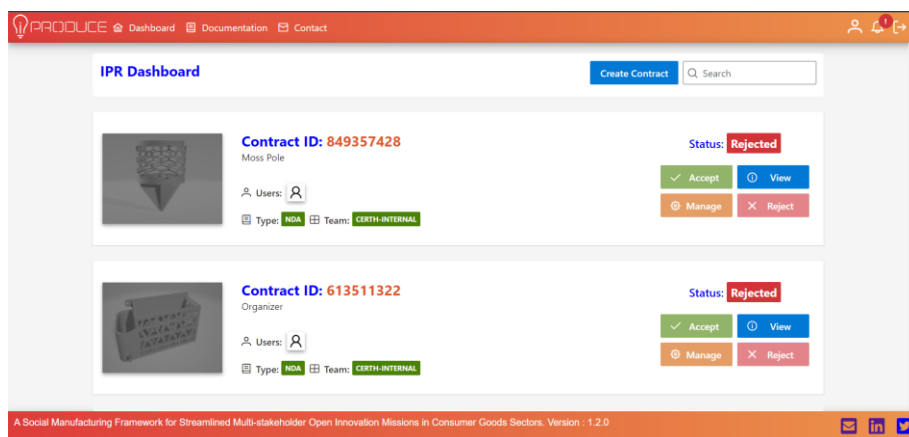


Figure 5-12: Rejected Contracts

In addition, the “Rejected Contract” contains the “View Contract” button component. Clicking on Show contract component, a dialogue box with the rejected Ricardian contract will appear.

Figure 5-13: Rejected Contract Agreement

5.2.2.4. Deployed Contracts

Contracts labeled as "deployed" have been accepted by all the users involved and have been deployed in the block-chain. The component related to deployed contracts presents important information like the Contract ID, Product, Type, Team, and the involved Partners. By using the filtering feature, users can easily access an accordion-style component that allows them to explore the details of the Ricardian Contract and determine its current status. The image below illustrates the component displaying deployed contracts.

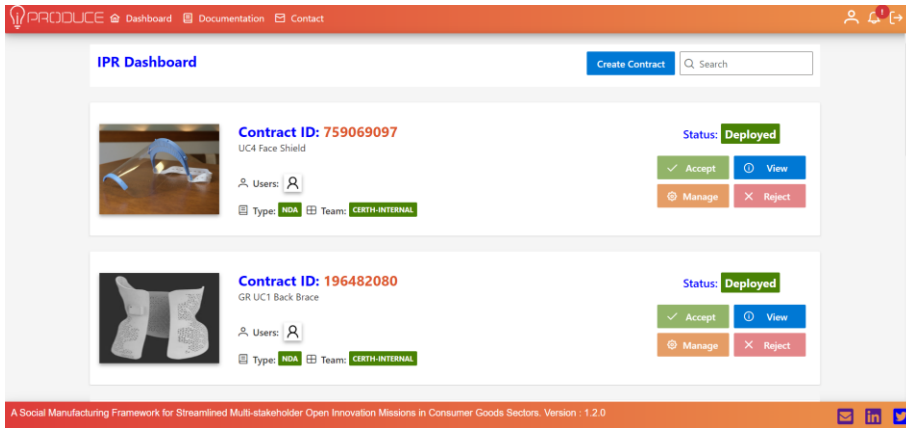


Figure 5-14: Deployed Contracts

The “Deployed Contracts” in addition, contain the button component “View Contract”.

- **View Contract**

Clicking View Contract component a Dialog box with the Deployed Ricardian Contract is shown.

5.2.2.5. Create Contract

Within the IPR dashboard page, there exists a button component labelled as "Create Contract". By clicking on this button, the user is able to generate a new Ricardian Contract for all the products associated with their team. When prompted, the user can add the Type for the contract. This functionality enables the user to conveniently initiate the creation of a new contract tailored to their team's products. The images below illustrate the Create Contract component.

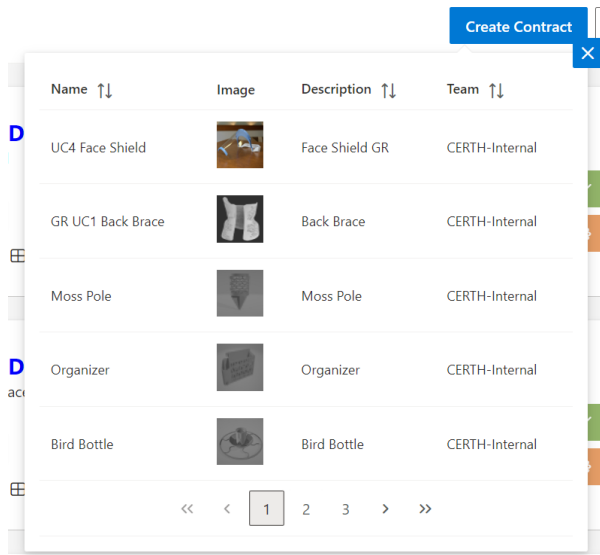


Figure 5-15: Create Contract Component

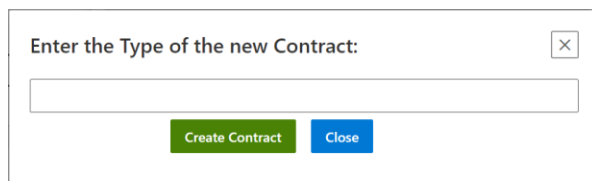


Figure 5-16: Contract Type Dialog Box

5.2.3. Documentation Page

The documentation page serves as a rich source of documentation for the IPR (Intellectual Property Rights) tool developed as part of the iProduce project. This comprehensive documentation aims to provide users with detailed insights into the functionalities, features, and usage instructions of the IPR tool.

The documentation begins with an introduction, offering an overview of the iProduce project and its objectives related to intellectual property management. It sets the context for understanding the significance of the IPR tool and how it aligns with the project's goals.

Moving further, the documentation delves into the core features and capabilities of the IPR tool. It explores various aspects, such as user management, contract creation, contract lifecycle management, and integration with blockchain technology. Each feature is accompanied by clear explanations, step-by-step instructions, and illustrative examples, ensuring that users can navigate and utilize the tool effectively.

Furthermore, the documentation provides comprehensive guidance on the different components and modules of the IPR tool. It covers topics like the dashboard, contract management, partner management, and communication features.

Additionally, the documentation offers insights into the user interface, highlighting the various elements, menus, and navigation options available within the IPR tool. This helps users familiarize themselves with the tool's layout and enables efficient navigation through its different sections.

To ensure a smooth user experience, the documentation includes troubleshooting tips, and best practices for using the IPR tool effectively. This further enhances user understanding and addresses common queries or challenges that may arise during usage.

In summary, the webpage's documentation for the IPR tool provides a comprehensive resource for users of the iProduce project. It equips users with the knowledge required to navigate, utilize, and maximize the potential of the IPR tool in effectively managing intellectual property rights. Whether it's understanding the tool's features, following step-by-step instructions, or troubleshooting issues, users can rely on this documentation to make the most of the IPR tool's capabilities.

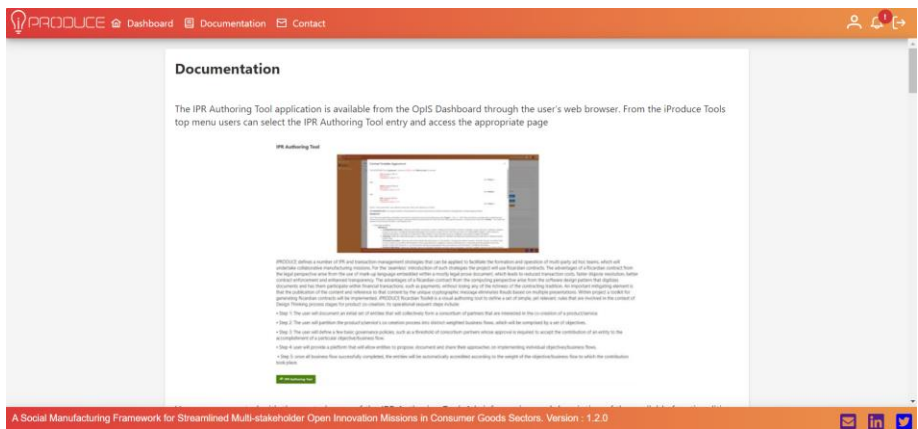


Figure 5-17: IPR Documentation Page

5.2.4. Contact Page

The Contact Page serves as a contact information resource for users of the IPR Authoring tool developed by the iProduce project. This page offers users a convenient way to get in touch with the relevant contacts and seek assistance or provide feedback regarding the tool.

The contact page provides essential information allowing users to reach out to the appropriate individuals or departments for their specific needs. This ensures that users can connect with the right people who can address their queries or concerns effectively.

Users can find contact information for various aspects related to the IPR Authoring Tool, such as technical support, general inquiries, or feedback regarding the tool's functionality. This enables users to seek assistance in case they encounter any technical difficulties, require clarification on specific features, or have suggestions for improvements.

By providing a dedicated contact page, the iProduce project demonstrates its commitment to maintaining effective communication channels with users of the IPR tool. It highlights the project's focus on user support, ensuring that users have a means to communicate their concerns, seek help, or share valuable feedback to enhance the overall user experience.

Overall, the contact page serves as a vital link between the iProduce project team and the users of the IPR tool. It fosters open communication, encourages user engagement, and enables a collaborative environment where users' needs and concerns can be addressed promptly and effectively.

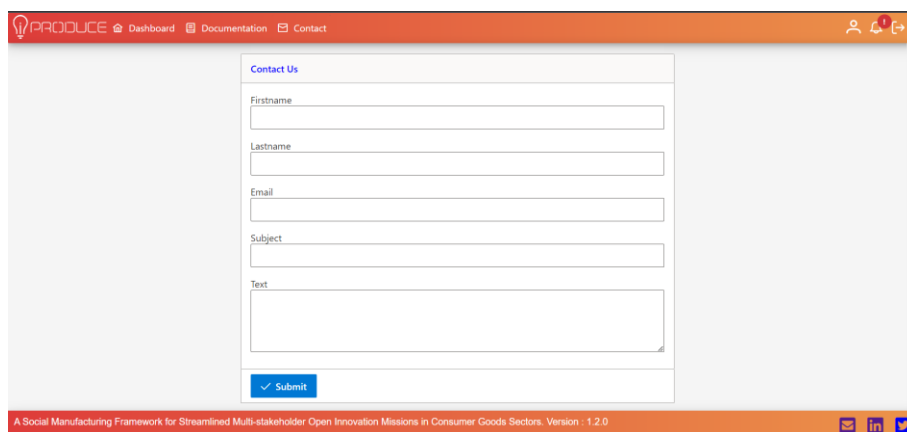


Figure 5-18: IPR Contact Us Page

5.3. Template Creation & Verification

In this section, we provide the implementation of the Creation and Verification of the Ricardian Contracts. On the IPR Authoring Tool backend, basic template functionalities that capture fundamental concepts of real-world contracts are defined. To formally verify the correctness of each template, the IPR Authoring Tool encodes them as predefined expressions in Scala language. The advantage of encoding the Ricardian contracts in a functional programming language such as Scala is that it empowers the formal verification of the correctness of each template which is a feature of Scala functional programming language. Moreover, more complex procedures such as auctions, revenue distribution schemes and simple asset lifecycle management can be encoded. The figure below depicts the Ricardian Contract template Creation and Verification procedure.

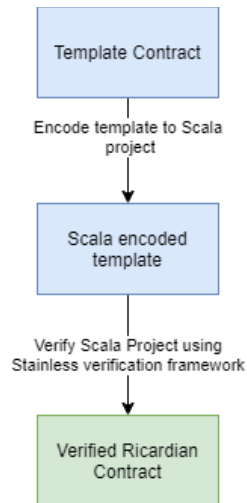


Figure 5-19: Contract Creation & Verification

In the development view of Ricardian Contract creation and verification, the Template is predefined as a simple HTML file. Using HTML to Scalatags converter extension of Scalatags library for Scala, empowers to encode the template in plain Scala code thus, it gives the ability to create a Scala Project with the encoded template. Using the Stainless verification framework, the Scala Project can be verified.

5.4. Ricardian Contract Deployment

In this section, we provide the Ricardian Contract Deployment in the blockchain implementation. The IPR Authoring Tool employs the EOSIO platform to create a local blockchain environment. As described in 4.1.2.1 the EOSIO adopts C++ as the official language for the development of smart contracts. Using the EOSIO WebAssembly compiler, the Ricardian Contract can be deployed to the blockchain and the verified template contract can be submitted as a transaction to the Ricardian Contract. The IPR Authoring Tool after a successful transaction, saves the transaction ID to be able to retrieve a Ricardian Contract through its ID.

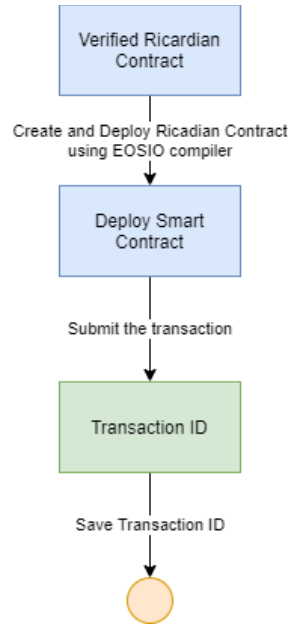


Figure 5-20: Ricardian Contract Deployment

5.5. The Application Layer

5.5.1. Component Specifications and Functional View

In this section, the IPR Authoring Tool components are presented. In a common template for all the components, all the functionalities, inputs/output, functional and non-functional requirements are reported. All the components fulfil the Functional requirements described in section 2.2.

Table 5-1: Human Readable IP Rules Manager

Description	
Name of Component/Service:	Human Readable IP Rules Manager.
Type:	Component
Functionality:	The Human Readable IP Rules Manager component, providing a User Interface enables the user and administrator to create, edit, reject, compile and deploy a human readable Ricardian Contract.
Input components	<ul style="list-style-type: none"> User input
Output components	<ul style="list-style-type: none"> Machine-Readable Contract Manager.
Non-Functional Requirements	<ul style="list-style-type: none"> The application must be responsive. The application must have a login. The application must be accessible via web. The connection with the other modules must be safe. The user interface shall be user-friendly and intuitive.

Table 5-2: Machine-Readable Contract Manager

Description	
Name of Component/Service:	Machine-Readable Contract Manager.
Type:	Component
Functionality:	The Machine-Readable Contract Manager component, translates the human readable Ricardian Contract into machine executable code. The component is responsible for the template creation and verification of the Ricardian contract. After a successful creation of the contract, the Machine-Readable Contract Manager sends the machine executable code plus the user valuables to the Decentralized IP rules manager.
Input components	<ul style="list-style-type: none"> Human Readable IP Rules Manager.
Output components	<ul style="list-style-type: none"> Decentralized IP Rules Manager.
Non-Functional Requirements	<ul style="list-style-type: none"> The application must be responsive. The connection with the other modules must be safe. IP protection for product owner. IP protection for cMDF data.

Table 5-3: Decentralized IP rules manager

Description	
Name of Component/Service:	Decentralized IP rules manager.
Type:	Component
Functionality:	The Decentralized IP rules manager, instantiates the Ricardian contracts deployed to the Blockchain network and is responsible to submit the proper actions and transactions, -based on the User Input- included in the Deployed Smart contracts.
Input components	<ul style="list-style-type: none"> Machine-Readable Contract Manager.
Output components	<ul style="list-style-type: none"> Smart contracts actions Transaction block/ID.
Non-Functional Requirements	<ul style="list-style-type: none"> The application must be responsive. The connection with the other modules must be safe. IP protection for product owner. IP protection for cMDF data.

5.5.2. Back-end services

In this section, we present the endpoints exposed by the IPR Authoring Tool back-end (server-side) for proper communication with the Web interface (Client application). The JSON objects provided in the input and output rows of the template tables are the example Data models of the IPR Authoring Tool. The data models will be enriched and improved, as the project workflow progresses.

Table 5-4: Retrieve Accepted Ricardian Contracts

Description	
Functionality description	This service call retrieves the Accepted Ricardian Contracts.

Technical specifications	<p>This module is developed by utilizing NodeJS and mongoDB database.</p> <p><i>Server Hardware</i> CPU: Intel(R) Core (TM) i5-9600K CPU @ 3.70GHz Memory: 16GB</p> <p><i>Software</i> Operating System: Windows 10 Pro mongoDB: 4.2.14 NodeJS version: 15.6.0</p>
Endpoint	/getAcceptedContracts
Method	GET
Input Data	user, sessionToken
Output Data	<pre>[{ "Id":3, "Partners":[{ "name":"John", "street":"New Street 1", "city":"Thessaloniki", "country":"Greece", "postalCode":"11111", "status":1, "date":"1/8/2021" }, { "name":"Smith", "street":"New Street 2", "city":"Thessaloniki", "country":"Greece", "postalCode":"11111", "status":1, "date":"1/8/2021" }, { "name":"Sam", "street":"New Street 3", "city":"Thessaloniki", "country":"Greece", "postalCode":"11111", "status":1, "date":"1/8/2021" }], "Product":"Chair_v1", "date":"1/8/2021" }]</pre>

Table 5-5: Retrieve Pending Ricardian Contracts

Description	
Functionality description	This service call retrieves the Pending Ricardian Contracts.
	This module is developed by utilizing NodeJS and mongoDB database.

Technical specifications	<p><i>Server Hardware</i> CPU: Intel(R) Core (TM) i5-9600K CPU @ 3.70GHz Memory: 16GB</p> <p><i>Software</i> Operating System: Windows 10 Pro mongoDB: 4.2.14 NodeJS version: 15.6.0</p>
Endpoint	/getPendingContracts
Method	GET
Input Data	user, sessionToken
Output Data	<pre>[{ "Id":1, "Partners":[{ "name":"John", "street":"New Street 1", "city":"Thessaloniki", "country":"Greece", "postalCode":"11111", "status":0, "date":"1/8/2021" }, { "name":"Smith", "street":"New Street 2", "city":"Thessaloniki", "country":"Greece", "postalCode":"11111", "status":0, "date":"1/8/2021" }, { "name":"Sam", "street":"New Street 3", "city":"Thessaloniki", "country":"Greece", "postalCode":"11111", "status":2, "date":"1/8/2021" }], "Product":"Chair_v1", "date":"1/8/2021" }]</pre>

Table 5-6: Retrieve Rejected Ricardian Contracts

Description	
Functionality description	This service call retrieves the Rejected Ricardian Contracts.
	This module is developed by utilizing NodeJS and mongoDB database.
	<p><i>Server Hardware</i> CPU: Intel(R) Core (TM) i5-9600K CPU @ 3.70GHz</p>

Technical specifications	Memory: 16GB Software Operating System: Windows 10 Pro mongoDB: 4.2.14 NodeJS version: 15.6.0
Endpoint	/getRejectedContracts
Method	GET
Input Data	user, sessionToken
Output Data	[<pre> { "Id":7, "Partners":[{ "name":"John", "street":"New Street 1", "city":"Thessaloniki", "country":"Greece", "postalCode":"11111", "date":"1/8/2021" }, { "name":"Smith", "street":"New Street 2", "city":"Thessaloniki", "country":"Greece", "postalCode":"11111", "date":"1/8/2021" }, { "name":"Sam", "street":"New Street 3", "city":"Thessaloniki", "country":"Greece", "postalCode":"11111", "date":"1/8/2021" }], "Product":"Chair_v1", "date":"1/8/2021", "reason":{ "user":"John", "msg":"dates are not valid" } } </pre>]

Table 5-7: Retrieve Deployed Ricardian Contracts

Description	
Functionality description	This service call retrieves the Deployed Ricardian Contracts.
	This module is developed by utilizing NodeJS and mongoDB database. <i>Server Hardware</i> CPU: Intel(R) Core (TM) i5-9600K CPU @ 3.70GHz Memory: 16GB

Technical specifications	<i>Software</i> Operating System: Windows 10 Pro mongoDB: 4.2.14 NodeJS version: 15.6.0
Endpoint	/getDeployedContracts
Method	GET
Input Data	user, sessionToken
Output Data	<pre>[{ "Id":5, "Partners":[{ "name":"John", "street":"New Street 1", "city":"Thessaloniki", "country":"Greece", "postalCode":"11111", "date":"1/8/2021" }, { "name":"Smith", "street":"New Street 2", "city":"Thessaloniki", "country":"Greece", "postalCode":"11111", "date":"1/8/2021" }, { "name":"Sam", "street":"New Street 3", "city":"Thessaloniki", "country":"Greece", "postalCode":"11111", "date":"1/8/2021" }], "Product":"Chair_v1", "date":"1/8/2021", "trx":"cc3e10df6df6df619157e55c152519243f797b87a93c62b7913525028727b0b3" }]</pre>

Table 5-8: Retrieve transactionID from the blockchain

Description	
Functionality description	This service call retrieves the Deployed Ricardian Contracts from transaction ID in the blockchain.
Technical specifications	This module is developed by utilizing Python and EOSIO blockchain platform. <i>Server Hardware</i> CPU: Intel(R) Core (TM) i5-9600K CPU @ 3.70GHz Memory: 16GB <i>Software</i>

	Operating System: Kubuntu 20.04 Python version: 3.6 Flask version: 2.0 EOSIO version: 2.0
Endpoint	/getTransaction
Method	GET
Input Data	transaction
Output Data	<pre> { "block_num": 1074, "block_time": "2021-05-19T11:09:42.500", "id": "cc3e10df6df6df619157e55c152519243f797b87a93c62b7913525028727b0b3", "last_irreversible_block": 197914, "traces": [{ "account_ram_deltas": [], "act": { "account": "john", "authorization": [{ "actor": "smith", "permission": "active" }], "data": { "user": "smith" }, "hex_data": "0000000080969dc4", "name": "createric" }, "action_ordinal": 1, "block_num": 1074, "block_time": "2021-05-19T11:09:42.500", "closest_unnotified_ancestor_action_ordinal": 0, "console": "action deployed, smith", "context_free": false, "creator_action_ordinal": 0, "elapsed": 57, "error_code": null, "except": null, "producer_block_id": null, "receipt": { "abi_sequence": 1, "act_digest": "c29c5e5984c53257a580bed26748a347b8e1ee183b77a37925a3d698e626e9d1", "auth_sequence": [["smith", 1]], "code_sequence": 1, "global_sequence": 1089, "receiver": "john", "recv_sequence": 1 }, "receiver": "john", </pre>

```

      "trx_id":
      "cc3e10df6df6df619157e55c152519243f797b87a93c62b7913525028727b0b3"
    },
    ],
    "trx": {
      "receipt": {
        "cpu_usage_us": 265,
        "net_usage_words": 13,
        "status": "executed",
        "trx": [
          1,
          {
            "compression": "none",
            "packed_context_free_data": "",
            "packed_trx":
            "14f2a4603004f65ee203000000001000000000301b7d0
            00040eaa6cd44501000000080969dc40000000a8ed323208000000080969dc4
            00",
            "signatures": [
              "SIG_K1_KdRk9YS89YfAd1xv8g1da9LLG97kZpKkuVv4bgFw9C
              XcdU5RYwBSfaHy8evHS7rHyCELR99avx6woi4343fE5aMeydPyhX"
            ]
          }
        ]
      },
      "trx": {
        "actions": [
          {
            "account": "john",
            "authorization": [
              {
                "actor": "smith",
                "permission": "active"
              }
            ],
            "data": {
              "user": "smith"
            },
            "hex_data": "0000000080969dc4",
            "name": "createric"
          }
        ],
        "context_free_actions": [],
        "context_free_data": [],
        "delay_sec": 0,
        "expiration": "2021-05-19T11:10:12",
        "max_cpu_usage_ms": 0,
        "max_net_usage_words": 0,
        "ref_block_num": 1072,
        "ref_block_prefix": 65167094,
        "signatures": [
          "SIG_K1_KdRk9YS89YfAd1xv8g1da9LLG97kZpKk
          uVv4bgFw9CXcdU5RYwBSfaHy8evHS7rHyCEL
          R99avx6woi4343fE5aMeydPyhX"
        ],
        "transaction_extensions": []
      }
    }
  }
}

```

Table 5-9: Reject Ricardian Contract

Description	
Functionality description	This service call rejects a Ricardian Contract.
Technical specifications	<p>This module is developed by utilizing NodeJS and mongoDB database.</p> <p><i>Server Hardware</i> CPU: Intel(R) Core (TM) i5-9600K CPU @ 3.70GHz Memory: 16GB</p> <p><i>Software</i> Operating System: Windows 10 Pro mongoDB: 4.2.14 NodeJS version: 15.6.0</p>
Endpoint	/rejectContract
Method	POST
Input Data	<pre>{ "Id":7, "Partners":[{ "name":"John", "street":"New Street 1", "city":"Thessaloniki", "country":"Greece", "postalCode":"11111", "date":"1/8/2021" }, { "name":"Smith", "street":"New Street 2", "city":"Thessaloniki", "country":"Greece", "postalCode":"11111", "date":"1/8/2021" }, { "name":"Sam", "street":"New Street 3", "city":"Thessaloniki", "country":"Greece", "postalCode":"11111", "date":"1/8/2021" }], "Product":"Chair_v1", "date":"1/8/2021", "reason":{ "user":"John", "msg":"dates are not valid" } }</pre>
Output Data	<pre>{ "msg": "Contract ID:7 rejected" }</pre>

Table 5-10: Add Ricardian Contract to Pending Contracts Page

Description	
Functionality description	This service call adds a Ricardian Contract to the pending contracts page.
Technical specifications	<p>This module is developed by utilizing NodeJS and mongoDB database.</p> <p><i>Server Hardware</i> CPU: Intel(R) Core (TM) i5-9600K CPU @ 3.70GHz Memory: 16GB</p> <p><i>Software</i> Operating System: Windows 10 Pro mongoDB: 4.2.14 NodeJS version: 15.6.0</p>
Endpoint	/addContract
Method	POST
Input Data	<pre>{ "Id":3, "Partners":[{ "name":"John", "street":"New Street 1", "city":"Thessaloniki", "country":"Greece", "postalCode":"11111", "status":1, "date":"1/8/2021" }, { "name":"Smith", "street":"New Street 2", "city":"Thessaloniki", "country":"Greece", "postalCode":"11111", "status":1, "date":"1/8/2021" }, { "name":"Sam", "street":"New Street 3", "city":"Thessaloniki", "country":"Greece", "postalCode":"11111", "status":1, "date":"1/8/2021" }], "Product":"Chair_v1", "date":"1/8/2021" }</pre>
Output Data	<pre>{ "msg": "Contract ID:3 added to the pending contracts page" }</pre>

Table 5-11: Deploy Ricardian Contract

Description	
Functionality description	This service call deploys a Ricardian Contract to the blockchain.
Technical specifications	<p>This module is developed by utilizing NodeJS and mongoDB database.</p> <p><i>Server Hardware</i> CPU: Intel(R) Core (TM) i5-9600K CPU @ 3.70GHz Memory: 16GB</p> <p><i>Software</i> Operating System: Windows 10 Pro, Kubuntu 20.04 mongoDB: 4.2.14 NodeJS version: 15.6.0 Python version: 3.6 Flask version: 2.0 EOSIO version: 2.0</p>
Endpoint	/deployContract
Method	POST
Input Data	<pre>{ "Id":3, "Partners":[{ "name":"John", "street":"New Street 1", "city":"Thessaloniki", "country":"Greece", "postalCode":"11111", "status":1, "date":"1/8/2021" }, { "name":"Smith", "street":"New Street 2", "city":"Thessaloniki", "country":"Greece", "postalCode":"11111", "status":1, "date":"1/8/2021" }, { "name":"Sam", "street":"New Street 3", "city":"Thessaloniki", "country":"Greece", "postalCode":"11111", "status":1, "date":"1/8/2021" }], "Product":"Chair_v1", "date":"1/8/2021" }</pre>
Output Data	<pre>{ "msg": "Contract ID:3 deployed successfully" "transaction": { "block_num": 1074,</pre>

```

"block_time": "2021-05-19T11:09:42.500",
"id":
"cc3e10df6df6df619157e55c152519243f797b87a93c62b7913525028727b0b3",
"last_irreversible_block": 197914,
"traces": [
  {
    "account_ram_deltas": [],
    "act": {
      "account": "john",
      "authorization": [
        {
          "actor": "smith",
          "permission": "active"
        }
      ],
      "data": {
        "user": "smith"
      },
      "hex_data": "0000000080969dc4",
      "name": "createric"
    },
    "action_ordinal": 1,
    "block_num": 1074,
    "block_time": "2021-05-19T11:09:42.500",
    "closest_unnotified_ancestor_action_ordinal": 0,
    "console": "action deployed, smith",
    "context_free": false,
    "creator_action_ordinal": 0,
    "elapsed": 57,
    "error_code": null,
    "except": null,
    "producer_block_id": null,
    "receipt": {
      "abi_sequence": 1,
      "act_digest":
"c29c5e5984c53257a580bed26748a347b8e1ee183b77a37925a3d698e626e9d1",
      "auth_sequence": [
        [
          "smith",
          1
        ]
      ],
      "code_sequence": 1,
      "global_sequence": 1089,
      "receiver": "john",
      "rcv_sequence": 1
    },
    "receiver": "john",
    "trx_id":
"cc3e10df6df6df619157e55c152519243f797b87a93c62b7913525028727b0b3"
  }
],
"trx": {
  "receipt": {
    "cpu_usage_us": 265,
    "net_usage_words": 13,
    "status": "executed",
    "trx": [
      1,

```

```

    {
      "compression": "none",
      "packed_context_free_data": "",
      "packed_trx":
"14f2a4603004f65ee203000000001000000000301b7d0
00040eaa6cd44501000000080969dc400000000a8ed323208000000080969dc4
00",
      "signatures": [
        "SIG_K1_KdRk9YS89YfAd1xv8g1da9LLG97kZpKkuVv4bgFw9C
XcdU5RYwBSfaHy8evHS7rHyCELR99avx6woi4343fE5aMeydPyhX"
      ]
    }
  ],
  "trx": {
    "actions": [
      {
        "account": "john",
        "authorization": [
          {
            "actor": "smith",
            "permission": "active"
          }
        ],
        "data": {
          "user": "smith"
        },
        "hex_data": "0000000080969dc4",
        "name": "createric"
      }
    ],
    "context_free_actions": [],
    "context_free_data": [],
    "delay_sec": 0,
    "expiration": "2021-05-19T11:10:12",
    "max_cpu_usage_ms": 0,
    "max_net_usage_words": 0,
    "ref_block_num": 1072,
    "ref_block_prefix": 65167094,
    "signatures": [
      "SIG_K1_KdRk9YS89YfAd1xv8g1da9LLG97kZpKk
uVv4bgFw9CXcdU5RYwBSfaHy8evHS7rHyCEL
R99avx6woi4343fE5aMeydPyhX"
    ],
    "transaction_extensions": []
  }
}
}

```

Table 5-12: Retrieve Users' Notifications

Description	
Functionality description	This service call retrieves the users' notifications
	This module is developed by utilizing NodeJS and mongoDB database.

Technical specifications	<p><i>Server Hardware</i> CPU: Intel(R) Core (TM) i5-9600K CPU @ 3.70GHz Memory: 16GB</p> <p><i>Software</i> Operating System: Windows 10 Pro mongoDB: 4.2.14 NodeJS version: 15.6.0</p>
Endpoint	/getUserIPRnotifications
Method	GET
Input Data	user, sessionToken
Output Data	<pre> { "message": "success", "contracts": [{ "users": [{ "userID": "011caa51-abab-4890-ba3e-3fc767a39247", "status": "1", "date": "27/03/2023", "notifications": [{ "message": "A contract for iProductId 610842578 is Pending", "date": "27/03/2023", "status": "read" }, { "message": "Contract ID: 759069097 is edited by a team member", "date": "27/03/2023", "status": "read" }], "userIndex": 1 }], "objectId": "759069097" }, { "users": [{ "userID": "011caa51-abab-4890-ba3e-3fc767a39247", "status": "1", "date": "04/04/2023", "notifications": [{ "message": "A contract for iProductId 221786921 is Pending", "date": "04/04/2023", "status": "read" }, { "message": "Contract ID: 196482080 is edited by a team member", "date": "04/04/2023", "status": "read" }] }] }] } </pre>

```

    ],
    "userIndex": 1
  }
],
"objectId": 196482080
}
]
}

```

Table 5-13: Retrieves the Users' Contracts, Teams, Products, and Team members

Description	
Functionality description	This service call retrieves the users' contracts, teams, products, and team members
Technical specifications	<p>This module is developed by utilizing NodeJS and mongoDB database.</p> <p><i>Server Hardware</i> CPU: Intel(R) Core (TM) i5-9600K CPU @ 3.70GHz Memory: 16GB</p> <p><i>Software</i> Operating System: Windows 10 Pro mongoDB: 4.2.14 NodeJS version: 15.6.0</p>
Endpoint	/IPRTeamProdUsers
Method	GET
Input Data	user, sessionToken
Output Data	<pre> { "message": "success", "users": [{ "username": "john", "firstName": "smith", "lastName": "john", "email": "test@iti.gr", "userID": "011caa51-abab-4890-ba3e-3fc767a39247", "teams": ["645559076"], "attributes": { "likes": 0, "views": 9, "updatedAt": "2023-06-20T08:28:25.932Z", "avatar": "", "city": "", "communitiesList": [], "communityMemberships": 0, "company": "", "facebookAccount": "", "favoritesList": [], "instagramAccount": "" } }] } </pre>

```

    "isAdmin": false,
    "isCommunityAdmin": false,
    "isLeadUser": false,
    "joinDate": "1679407896.070679",
    "lastLoginDate": "1685356443.486095",
    "location": "",
    "marketplaceRole": "",
    "messageList": [],
    "notificationList": [
      "883769700"
    ],
    "productList": [
      "280271994",
      "579124702"
    ],
    "profileDescription": "",
    "profilePublic": false,
    "sector": "",
    "skills": "",
    "teamInvitations": [],
    "teamMemberships": 1,
    "teamsList": [
      "6"
    ],
    "transactionsList": [],
    "twitterAccount": ""
  },
  "createdAt": "2023-03-21T14:10:24.361Z"
}
],
"products": [
  {
    "baseProduct": {
      "productName": "UC4 Face Shield",
      "productPitch": "Face Shield GR",
      "materialsList": [
        "ABS,Plexy glass "
      ],
      "toolsList": [
        "3D FDM Printer Ender 3 pro heavily modified "
      ],
      "creationDate": "2023-03-27T13:41:48.176573Z",
      "lastModified": "2023-03-28T09:37:21.182431Z"
    },
    "productCustomization": [],
    "productDescription": [],
    "labels": [],
    "surveyList": [],
    "comments": [],
    "image": {
      "name": "2dc820dd39a3c485b12b26f96eb5a405_image.jpeg",
      "url": "https://iproduce-
tools.iti.gr/opis/parse/files/iProduceID/2dc820dd39a3c485b12b26f96eb5a405_image.j
peg"
    },
    "attachmentsITI": [
      {

```

```

      "url": "https://iproduce-
tools.iti.gr/opis/parse/files/iProduceID/e2a2f92009b21174a613726477b06f7e_UC4_Cu
stomized_Face_Shield.fbx",
      "name":
      "e2a2f92009b21174a613726477b06f7e_UC4_Customized_Face_Shield.fbx"
    },
    "productId": 610842578,
    "owner": null,
    "teamId": "645559076",
    "isPublic": false,
    "createdByTeam": true,
    "likes": 0,
    "views": 0,
    "price": 18,
    "creatorUserID": "011caa51-abab-4890-ba3e-3fc767a39247",
    "createdAt": "2023-03-27T13:41:48.640Z",
    "updatedAt": "2023-03-28T09:37:22.157Z",
    "objectId": 610842578
  }
],
"contracts": [
  {
    "users": [
      {
        "userID": "011caa51-abab-4890-ba3e-3fc767a39247",
        "status": "1",
        "date": "27/03/2023",
        "notifications": [
          {
            "message": "A contract for iProductId 610842578 is Pending",
            "date": "27/03/2023",
            "status": "read"
          }
        ],
        "userIndex": 1
      }
    ],
    "admin": {
      "status": "1",
      "deploy": true,
      "userID": "011caa51-abab-4890-ba3e-3fc767a39247"
    },
    "productId": 610842578,
    "date": "27/03/2023",
    "teamId": 645559076,
    "lastEdited": 1679924508844,
    "padID": 1679924508656,
    "createdAt": "2023-03-27T13:41:48.865Z",
    "updatedAt": "2023-05-29T13:16:56.264Z",
    "descPR": "Face Shield GR",
    "image": "https://iproduce-
tools.iti.gr/opis/parse/files/iProduceID/2dc820dd39a3c485b12b26f96eb5a405_image.j
peg",
    "namePR": "UC4 Face Shield",
    "productIndex": 0,
    "status": "Pending",
    "teamName": "CERTH-INTERNAL",
    "type": "NDA",

```

```

    "objectId": 759069097
  }
],
"teams": [
  {
    "members": [
      "69d43ff6-85d4-4009-957d-c87b732292ed",
      "011caa51-abab-4890-ba3e-3fc767a39247"
    ],
    "teamId": 645559076,
    "teamName": "CERTH-Internal",
    "profileDescription": "CERTH-Internal",
    "likes": 0,
    "views": 10,
    "profilePublic": false,
    "joinDate": "2023-03-27T13:39:47.346965Z",
    "lastLoginDate": "2023-06-20T08:33:42.068391Z",
    "productsList": [
      610842578
    ],
    "creatorUserID": "011caa51-abab-4890-ba3e-3fc767a39247",
    "createdAt": "2023-03-27T13:39:47.534Z",
    "updatedAt": "2023-06-20T08:58:01.070Z",
    "objectId": 645559076
  }
]
}
}

```

Table 5-14: Retrieve User's Avatar

Description	
Functionality description	This service call retrieves the user's avatar
Technical specifications	<p>This module is developed by utilizing NodeJS and mongoDB database.</p> <p><i>Server Hardware</i> CPU: Intel(R) Core (TM) i5-9600K CPU @ 3.70GHz Memory: 16GB</p> <p><i>Software</i> Operating System: Windows 10 Pro mongoDB: 4.2.14 NodeJS version: 15.6.0</p>
Endpoint	/getUserAvatar
Method	GET
Input Data	user, sessionToken
Output Data	Image data

Table 5-15: Retrieves Block-chain Account Ricardian Contracts

Description	
Functionality description	This service call creates a team contract in the Block-chain

Technical specifications	<p>This module is developed by utilizing NodeJS and mongoDB database.</p> <p><i>Server Hardware</i> CPU: Intel(R) Core (TM) i5-9600K CPU @ 3.70GHz Memory: 16GB</p> <p><i>Software</i> Operating System: Windows 10 Pro mongoDB: 4.2.14 NodeJS version: 15.6.0</p>
Endpoint	/ getUsersContracts
Method	GET
Input Data	user
Output Data	<pre> { "transaction_id": "1377ac90bf562790155524bff3ef2e09850396d7eeeb132c0f2d8fd9aae75c8", "processed": { "id": "1377ac90bf562790155524bff3ef2e09850396d7eeeb132c0f2d8fd9aae75c8", "block_num": 86608, "block_time": "2023-06-21T10:01:28.500", "producer_block_id": null, "receipt": { "status": "executed", "cpu_usage_us": 326, "net_usage_words": 13 }, "elapsed": 326, "net_usage": 104, "scheduled": false, "action_traces": [{ "action_ordinal": 1, "creator_action_ordinal": 0, "closest_unnotified_ancestor_action_ordinal": 0, "receipt": { "receiver": "iprcontract", "act_digest": "41acac8a75ad4a903a3eb7e4adf2361e1bb077cb772f09fd0ceae16c643a82a0", "global_sequence": 86710, "recv_sequence": 85, "auth_sequence": [["contractacc", 85]], "code_sequence": 8, "abi_sequence": 3 }, "receiver": "iprcontract", "act": { "account": "iprcontract", "name": "getcontract", "authorization": [{ "actor": "user", "permission": "active" }] } }] } } </pre>

```

    }
  ],
  "data": {
    "user": "user"
  },
  "hex_data": "00103219999b2745"
},
"context_free": false,
"elapsed": 35,
"console": "[[{"Message ID": 0, "Content": "\"content\""}, {"Message ID": 1,
"Content": "\"content\""}]]",
"trx_id":
"1377ac90bf562790155524bff3ef2e09850396d7e000b132c0f2d8fd9aae75c8",
"block_num": 86608,
"block_time": "2023-06-21T10:01:28.500",
"producer_block_id": null,
"account_ram_deltas": [],
"account_disk_deltas": [],
"except": null,
"error_code": null,
"return_value_hex_data": ""
}
],
"account_ram_delta": null,
"except": null,
"error_code": null
},
"data": "[[{"Message ID": 0, "Content": "\"content\""}, {"Message ID": 1, "Content":
"\" content\""}]]"
}

```

Table 5-16: Create Block-chain Account

Description	
Functionality description	This service call creates users account into the block-chain
Technical specifications	<p>This module is developed by utilizing NodeJS and mongoDB database.</p> <p><i>Server Hardware</i> CPU: Intel(R) Core (TM) i5-9600K CPU @ 3.70GHz Memory: 16GB</p> <p><i>Software</i> Operating System: Windows 10 Pro mongoDB: 4.2.14 NodeJS version: 15.6.0</p>
Endpoint	/createAccount
Method	POST
Input Data	{ "username" : "newuser" }
Output Data	{ "message": "user created", "name": "newuser", "privetekey": "123", "publicKey": "123" }

Table 5-17: Import Contract into the Block-chain

Description	
Functionality description	This service call imports a contract into the block-chain
Technical specifications	<p>This module is developed by utilizing NodeJS and mongoDB database.</p> <p><i>Server Hardware</i> CPU: Intel(R) Core (TM) i5-9600K CPU @ 3.70GHz Memory: 16GB</p> <p><i>Software</i> Operating System: Windows 10 Pro mongoDB: 4.2.14 NodeJS version: 15.6.0</p>
Endpoint	/createContract
Method	POST
Input Data	<pre>{ "username": "newuser", "content": "contract_text", "authors": ["newuser"] }</pre>
Output Data	<pre>{ "transaction_id": "67a9be4fd94e60b92f3c12becc544e0cd762ce4f7faa54db0b50e8f5f3f3849a", "processed": { "id": "67a9be4fd94e60b92f3c12becc544e0cd762ce4f7faa54db0b50e8f5f3f3849a", "block_num": 87523, "block_time": "2023-06-21T10:09:06.000", "producer_block_id": null, "receipt": { "status": "executed", "cpu_usage_us": 167, "net_usage_words": 16 }, "elapsed": 167, "net_usage": 128, "scheduled": false, "action_traces": [{ "action_ordinal": 1, "creator_action_ordinal": 0, "closest_unnotified_ancestor_action_ordinal": 0, "receipt": { "receiver": "iprcontract", "act_digest": "13beb7a357566b632ee8c45e2b595ceb704893026d09a36140ae81f2906ddd9", "global_sequence": 87628, "recv_sequence": 86, "auth_sequence": [["newuser", 1]] } }] } }</pre>


```

    ]
  ],
  "code_sequence": 8,
  "abi_sequence": 3
},
"receiver": "iprcontract",
"act": {
  "account": "iprcontract",
  "name": "addcontract",
  "authorization": [
    {
      "actor": "newuser",
      "permission": "active"
    }
  ]
},
"data": {
  "sender": "newuser",
  "content": "contract_text",
  "users": [
    "newuser"
  ]
},
"hex_data":
"000000e02aacb99a0d636f6e74726163745f7465787401000000e02aacb99a"
},
"context_free": false,
"elapsed": 67,
"console": "Message ID: 2\n",
"trx_id":
"67a9be4fd94e60b92f3c12becc544e0cd762ce4f7faa54db0b50e8f5f3f3849a",
"block_num": 87523,
"block_time": "2023-06-21T10:09:06.000",
"producer_block_id": null,
"account_ram_deltas": [
  {
    "account": "newuser",
    "delta": 143
  }
],
"account_disk_deltas": [],
"except": null,
"error_code": null,
"return_value_hex_data": ""
}
],
"account_ram_delta": null,
"except": null,
"error_code": null
},
"data": "Message ID: 2\n"
}

```

5.5.3. Approach and used technologies for IPR

The software packages that were used to develop the IPR Authoring Tool are described in detail in this section.



5.5.3.1. Angular

Angular is an HTML and TypeScript-based open-source platform and framework for creating single-page client applications. Angular is the successor of AngularJS, and all references to Angular are for versions 2 and higher. Features such as generics, static-typing and ES6 capabilities are available in Angular. AngularJS was initially released by Google on October 20, 2010. The latest stable version of Angular is v12.0.1.

5.5.3.2. NodeJS

Node.js is a scalable network application runtime that is asynchronous and event-driven in JavaScript. It is often used for the implementation of web servers among others. Node.js is open-source MIT-licensed and can run on various platforms such as Windows, Linux, Unix, Mac OS X, etc. Node.js performance is fast due to the asynchronous way of handling the requests. Moreover, it is memory efficient. The Node package manager of Node.js which is widely used for installing JavaScript libraries, includes hundreds of thousands of packages.

5.5.3.3. RESTful web API

Representational State Transfer (REST) is frequently used for the creation of interactive applications that use Web Services. REST is a software architectural style that uses a subset of HTTP. The architectural style of REST was presented by Roy Fielding in 2000¹². The following six constraints must be satisfied for an interface to be referred as RESTful.

- Uniform interface
- Client-server
- Stateless
- Cacheable
- Layered system
- Code on demand

5.5.3.4. Flask

Flask is a Python-based web application framework. Flask framework is based on the Werkzeug WSGI toolkit and the Jinja2 template engine, which are both Pocco projects. Python web application implementations use Web Server Gateway Interface (WSGI) as a standard. The Werkzeug WSGI toolkit used in the Flask framework, implements all the required WSGI requests, response objects, and utility functions. In addition, the Jinja2 template engine included in Flask manages the rendering of dynamic web pages. Flask is designed to keep the core of the application simple and scalable and in addition, supports extensions for database support.

¹² https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

5.5.3.5. Etherpad

Etherpad is an open-source web-based tool that facilitates real-time collaborative editing of documents. Developed in 2008 and later released as an open-source project, Etherpad allows multiple users to work on the same document simultaneously, providing unique colors to identify contributions. It offers features like live chat, revision history, and basic formatting options, while remaining lightweight and easy to set up on personal servers. With applications in education, software development, journalism, and more, Etherpad promotes seamless and efficient document collaboration, enhancing productivity and creativity for teams and individuals.

6. Conclusions

This deliverable represents the second and final version, encompassing the comprehensive architectural and functional description of the IPR Authoring tool and Transaction Management Strategies for iPRODUCE Social Manufacturing Missions. All the proposed objectives have been successfully accomplished in this version.

The IPR Authoring Tool has been meticulously developed, encompassing multiple software components, over the first 18 months. These components have been fully integrated and interconnected with other OpIS platform elements, including the Marketplace and the Matchmaking toolkit. Additionally, extensive improvements have been made to the User Interface, back-end services, and template functionalities. These enhancements effectively cover the use cases, functional and non-functional requirements, resulting in a highly intuitive and user-friendly interface.

In terms of technology, this deliverable provides an insightful overview of the basic technologies employed and the features implemented during the design and development process of the IPR Authoring tool and Transaction Management Strategies for iPRODUCE Social Manufacturing Missions. The report thoroughly explains Blockchain Technology, highlighting various leading platforms, and delves into the significant role of Smart Contracts, particularly Ricardian Contracts.

Moreover, the deliverable offers a comprehensive description of the creation process of the IPR Authoring Visual tool, providing an intricate account of both the internal workings of the backend and frontend components. It also outlines the detailed procedure for user authentication and contract template processing and deployment.

Lastly, a comprehensive outline of the general additions, improvements, and fixes that have been implemented in the final version of the platform is being reported. The modifications have been carefully aligned with the specific needs and expectations of the iPRODUCE platform.

In summary, this second and final version of the report successfully encompasses the complete architectural and functional description of the IPR Authoring tool and Transaction Management Strategies for iPRODUCE Social Manufacturing Missions. All the outlined objectives have been accomplished, resulting in a robust platform that effectively meets the project's requirements. The insights presented herein serve as a solid foundation for the ongoing development and future success of the iPRODUCE platform.

7. References

- [1] Narayanan, A., Bonneau, J., Felten, E., Miller, A., and Goldfeder, S., Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction, Princeton University Press, 2016.
- [2] D. Yaga, P. Mell, N. Roby, K. Scarfone, Blockchain Technology Overview, National Institute of Standards and Technology, 2018
- [3] National Institute of Standards and Technology, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, Federal Information Processing Standards (FIPS) Publication 202, August 2015.
- [4] National Institute of Standards and Technology (NIST), Digital Signature Standard, Federal Information Processing Standards (FIPS) Publication 186-4, July 2013
- [5] Ethereum Yellow Paper. (2018). [Online]. Available: <https://ethereum.github.io/yellowpaper/paper.pdf>
- [6] K. Bhargavan et al., "Formal verification of smart contracts: Shortpaper," in Proc. ACM Workshop Program. Lang. Anal. Security (PLAS), Vienna, Austria, Oct. 2016, pp. 91–96,
- [7] 2019. EOSIO Official Portal. <https://eos.io/>.
- [8] Y. Huang, H. Wang, Understanding (Mis)behaviour on the EOSIO Blockchain Proc. ACM Meas. Anal. Comput. Syst., Vol. 4, No. 2, Article 37. Publication date: June 2020.
- [9] J. Bacon, J.D. Michels, C. Millard, J. Singh, Blockchain Demystified: A technical and legal introduction to distributed and centralised ledgers, 25 Rich. J.L. & Tech, no. 1, 2018

ANNEX

Address	A short, alphanumeric string derived from a user's public key using a hash function, with additional data to detect errors. Addresses are used to send and receive digital assets.
Assets	Anything that can be transferred
Asymmetric-key cryptography	Users have a private key that is kept secret and is used to generate a public key in this cryptographic scheme (which is freely provided to others). Users can use their private key to digitally sign data, and anyone with the associated public key can verify the signature.
Block	A data structure containing a block header and block data.
Block data	The portion of a block that contains a set of validated transactions and ledger events.
Block header	Block metadata is the part of a block that comprises information about the block itself, such as a timestamp, a hash representation of the block data, the hash of the preceding block's header, and a cryptographic nonce (if needed).
Blockchain	Blockchains are block-based distributed digital ledgers comprising cryptographically signed transactions. After validation and consensus, each block is cryptographically connected to the one before it (making it tamper obvious). Older blocks become increasingly difficult to change as new blocks are added (creating tamper resistance). New blocks are replicated across copies of the ledger within the network, and any conflicts are resolved automatically using established rules
Cryptocurrency	Within the system, a digital asset/credit/unit that is cryptographically sent from one blockchain network user to another. The publishing node comprises a transaction that sends the freshly minted cryptocurrency to one or more blockchain network users in the case of cryptocurrency creation (such as the reward for mining).
Cryptographic hash function	A function that maps a bit string of arbitrary length to a fixed-length bit string.
Digest	See hash digest
Hash chain	An append-only data structure in which data is organized into data blocks, each of which contains a hash of the previous data block's data. Because any update to a data block changes the hash digest recorded by the succeeding data block, this data structure gives evidence of manipulation.
Hash digest	The output of a hash function (e.g., $\text{hash}(\text{data}) = \text{digest}$).
Hashing	By applying a cryptographic hash function to the input data, a way of producing a relatively unique output (called a hash digest) for an input of practically any size (a file, text, image, etc.).
Ledger	A record of transactions.
Node	An individual system within the blockchain network.
Permissionless	A system where all users' permissions are equal and not set by any administrator or consortium.
Public key cryptography	See Asymmetric-key cryptography.
Smart contract	A collection of code and data (also known as functions and state) that is put on the blockchain network using cryptographically signed transactions. The smart contract

	is executed by nodes in the blockchain network; all nodes must produce the same results, and the results are recorded on the blockchain.
Tamper evident	A process which makes alterations to the data easily detectable.
UTXO	A transaction output that has not yet been spent is referred to as a UTXO. Only unspent outputs can be utilized as inputs in an accepted transaction in a valid blockchain payment system (such as Bitcoin). When a transaction occurs, inputs are deleted and new UTXOs are created, which can then be consumed in future transactions.