Blockchain Based DRM Management System

Submitted in partial fulfillment of the requirements
for the degree of

B.E. Computer Engineering

By

Vendrell Mendonca	**30**	**202125**
Ashwin Pillai	**37**	**202103**
Elbin Thomas Abraham	**45**	**202040**
Kyle Crasto	**62**	**202024**

Guide
Ms. Annies Minu
Assistant Professor

Department of Computer Engineering
St. Francis Institute of Technology
(Engineering College)

University of Mumbai

2023-2024

CERTIFICATE

This is to certify that the project entitled **"Blockchain Based DRM Management System"** is a bonafide work of **"Vendrell Mendonca(30), Ashwin Pillai(37), Elbin Thomas Abraham(45) and Kyle Crasto(62)"** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of B.E. in Computer Engineering.

[illegible] 06/05/24

Ms. Annies Minu

(Guide)

[illegible] 6/05/24

Dr. Kavita Sonawane

(Head Of Department)

[illegible]

Dr. Sincy George

(Principal)

Project Approval Report for B.E.

This project report entitled ***"Blockchain Based DRM Management System"*** by ***Vendrell Mendonca(30), Ashwin Pillai(37), Elbin Thomas Abraham(45) and Kyle Crasto(62)*** is approved for the degree of ***B.E. in Computer Engineering***.

Examiners

1. [illegible]

2. [illegible] 07/05/24

Date: 07/05/2024

Place: Mumbai

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Vendrell

Vendrell Mendonca (30)

[illegible]

Ashwin Pillai (37)

Elbin

Elbin Thomas Abraham (45)

[illegible]

Kyle Crasto (62)

Date: 07/05/2024

Abstract

Our project introduces a Blockchain-Based DRM Management System designed for the music industry. This web application serves as a central hub for record labels, artists, listeners, and commercial entities, streamlining the licensing and distribution of music content while enhancing transparency and fairness. Leveraging blockchain technology, the system maintains an immutable ledger that tracks the entire music licensing and distribution process. This transparency promotes trust among stakeholders, allowing artists to easily monitor their earnings and record labels to efficiently manage royalty payouts. Users can securely purchase music with lifelong ownership rights, and commercial entities can secure music rights. The system supports various user roles, including Record Labels, Artists, Users, and Commercial Entities, each with distinct functions. Record Labels can track active songs, manage artists, set pricing, define contracts, and manage royalties. Artists can monitor earnings, track song performance, and view active contracts. Users and Commercial Entities can easily purchase music and ensure licensing compliance. Our project addresses common challenges in the music industry, including royalty payments and unauthorized music usage. By implementing smart contracts, it automates the licensing process, ensuring prompt payments and accurate music usage monitoring.

Keywords: *Blockchain-Based, DRM Management System, Music Industry, Royalty Payouts, Smart Contracts, Revenue Distribution*

Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Description

Blockchain is a distinct form of database that stores information in a unique manner. It employs a chain of blocks to house data, with each new piece of information entering a fresh block. Once a block is full, it becomes linked to the preceding block, resulting in a chronological chain of data. While diverse data types can find a home on a blockchain, the predominant application has been as a transaction ledger. In Ethereum's context, blockchain is utilized in a decentralized fashion, ensuring that no single individual or entity wields control. Decentralized blockchains, like Ethereum, are characterized by their immutability, signifying that once data is entered, it becomes irreversible. In Ethereum's case, this permanent recording and accessibility of transactions is accessible to all.

In the ever-evolving landscape of the digital music industry, a pressing need has emerged for a sophisticated, transparent, and secure platform to manage the licensing and distribution of revenue for music content. Our project aims to address this need by introducing a cutting-edge Blockchain-Based DRM Management System. This comprehensive web application will serve as the central hub for music record labels, artists, users, and commercial entities to interact, making the process of licensing and distribution of music seamless, efficient, and equitable.

The core functionality of our platform revolves around ensuring that music creators and record labels receive fair compensation for their work. By harnessing the power of blockchain technology, we establish an immutable ledger that tracks each step of the music licensing and distribution process. This transparency promotes trust and enables artists to easily track their earnings, while record labels can manage royalties payouts efficiently. Users and commercial entities can securely purchase music with lifetime validity.

1.2 Problem Formulation

The contemporary music industry faces numerous challenges when it comes to licensing, revenue distribution, and Digital Rights Management (DRM). The existing systems are often riddled with complexities, lack transparency, and suffer from issues like delayed royalty payments, unauthorized usage of music, and disputes over licensing terms. These problems create frustration and financial uncertainty for music creators and record labels, hampering the growth and sustainability of the industry.

To address these challenges, our project aims to create a robust Blockchain-Based DRM Management System. The core problem we seek to solve is the lack of a streamlined and secure platform that offers all stakeholders in the music ecosystem, from artists and record labels to users and commercial entities, a user-friendly, transparent, and fair solution for licensing and revenue distribution. The decentralized nature of blockchain technology allows for smart contracts that can automate the process, ensuring that payments are made promptly and in accordance with the specified terms, and that music usage is closely monitored.

1.3 Motivation

Through this system, we aspire to empower artists and record labels with the tools they need to manage their music catalog, while offering users and commercial entities the flexibility and control they seek when accessing music. In doing so, we believe we can revolutionize the way the music industry handles DRM and revenue distribution, fostering a more equitable and prosperous environment for all stakeholders.

1.4 Proposed Solution

The proposed solution is a Blockchain-Based Digital Rights Management (DRM) Management System designed to revolutionize the licensing and distribution of music via a web application. This system will facilitate efficient revenue generation, royalty distribution, and licensing control for record labels, artists, users, and commercial entities.

At its core, the system will employ blockchain technology to ensure secure and transparent transactions. A landing page will serve as the gateway, allowing users to log in or sign up, differentiating between record labels, artists, users, and commercial entities.

Record Labels will have the ability to manage their catalog, including adding songs, set royalty payout criteria, and determine pricing for both end-users and commercial entities, using smart contracts. Artists will have real-time access to earnings and performance data for their music. Users and commercial entities can purchase music with lifetime validity only accessing the music via the site.

The platform will enable the creation of licensing agreements and royalty distribution according to record labels specified terms through the use of smart contracts, ensuring a secure and automated process. This system offers a robust and technologically advanced solution for the dynamic music industry, revolutionizing DRM and revenue management.

1.5 Scope of the project

The Blockchain-Based DRM Management System is a comprehensive solution for the digital music industry, falling under the domain of digital rights management (DRM). This project encompasses user roles and authentication, account management, artist and record label management, digital asset management, licensing management, earnings and royalties tracking, user functionality, and personalized dashboards. It leverages blockchain technology to secure and decentralize data, ensuring data integrity and resistance to tampering and piracy. This system is designed to provide a transparent, efficient, and secure platform for DRM, licensing, and royalty management, promoting trust and fair compensation in the digital music industry.

Chapter 2

Review of Literature

PAPER 01

Title: A digital rights management system based on a scalable blockchain[1]

Authors: Abba Garba, Ashutosh Dhar Dwivedi, Mohsin Kamal, Gautam Srivastava, Muhammad Tariq, M. Anwar Hasan, and Zhong Chen

The paper proposes a distributed media transaction framework for Digital Rights Management (DRM) based on digital watermarking and a scalable blockchain model. The proposed DRM model allows only authorized users to use online content and provide original multimedia content. While the digital watermarking is used to reclaim the copyright ownership of offline contents in the event when the contents are leaked.

The paper focuses on improving the classic blockchain systems to make it suitable for a DRM model. The authors argue that even though the Internet promotes data sharing and transparency, it does not protect digital content. In today's digital world, it has become a difficult task to release a DRM system that can be considered well-protected. Digital content that becomes easily available in open-source environments will in time be worthless to the creator. There may only be a one-time payment to creators upon initial upload to a given platform after which time the rights of the intellectual property are shifted to the platform itself. However, due to the online availability of content, anyone can download content and make copies. The value of digital content slowly decreases, because the value of content can usually be determined through the difficulty of its accessibility. There is no way to track the leakage or copyright for the spread of digital material. The proposed DRM model is based on a scalable blockchain model that allows for secure and efficient transactions between users. The authors claim that their proposed system is more secure than traditional DRM systems because it uses blockchain technology, which is inherently secure and tamper-proof.

However, there are some limitations to this proposed system. For example, it may not be suitable for all types of digital content, such as those that require high-speed data transfer or those that require real-time processing. Additionally, there may be some concerns about privacy and security when using blockchain technology for DRM purposes. [1] The paper acknowledges the role of the administrator in overseeing licensing and revenue distribution for music, as well as the use of smart contracts to automate licensing processes. However, it does not extensively delve into licensing negotiation, the establishment of fair revenue-sharing models, and the intricacies of integrating these aspects into the proposed DRM framework.

PAPER 02

Title: Blockchain based digital rights management systems: Design principles for the music industry[2]

Authors: Raffaele Fabio Ciriello, Alexander Christian G. Torbensen, Michael R. P. Hansen, and Jonas Hedman.

The paper proposes design principles for blockchain-based DRM systems that provide an integrated and flexible solution by enabling transparent music li-

censing structures, consistent and complete rights metadata, and efficient and transparent royalty payout. The authors analyze the music industry as a case in point and argue that existing centralized digital rights management (DRM) systems mostly serve the interests of major publishers, with scant inclusion of rights owners, creators, and consumers.
The proposed solution can be achieved by :

1. Store public metadata on a distributed ledger : The evolution to on-demand music streaming, together with the absence of a global rights database, has resulted in the development of complex licensing processes over time, which are error-prone and time-consuming. [2]

2. Validate metadata via a consensus mechanism on a permissioned blockchain : Various organizations in the music industry currently maintain separate databases, leading to inconsistent and incomplete music metadata. [2]

3. Algorithmically enforce royalty payout via stablecoin : Complex licensing structures and dispersed metadata lead to massive delays in the payout of royalties to rights owners, with one billion US dollars annually unallocated. [2]

The paper highlights that most commercial DRM systems use digital watermarks to identify IP rights owners and are centrally controlled by major labels and publishers. The assumption is that DRM systems benefit rights owners by making piracy costly and difficult. However, these systems do not protect the interests of rights owners, creators, and consumers. The proposed blockchain-based DRM system is more secure than traditional DRM systems because it uses blockchain technology, which is inherently secure and tamper-proof.

However, there are some limitations to this proposed system. For instance, it may not be suitable for all types of digital content, such as those that require high-speed data transfer or those that require real-time processing. Additionally, there may be some concerns about privacy and security when using blockchain technology for DRM purposes.

PAPER 03

Title: Blockchain Technology and Its Applications in Digital Content Copyright Protection[3]

Authors: Jiyin Shen

The paper discusses the application of blockchain technology in digital content copyright protection. It highlights the growing challenges in digital copyright protection, such as difficulty in confirming rights, infringement issues, and the complexity of safeguarding digital rights. The paper proposes that blockchain technology, with its features of decentralization, timestamp, and distributed storage, can provide potential solutions to these problems.

It starts by introducing blockchain technology, its origins, and features, emphasizing its decentralized and distributed nature. The paper then delves into the dilemmas of contemporary digital right protection, including issues related to digital right confirmation, infringement, and safeguarding.

The advantages of applying blockchain in digital right protection are discussed, including timestamping for copyright confirmation, decentralization for improved efficiency, cost-saving, and security, as well as distributed storage to meet the growing demand for digital products. [3]

The paper also identifies several challenges when applying blockchain to digital right protection, including the immaturity of blockchain technology, high development costs, legal issues related to standardization and global circulation, and human resource challenges in terms of expertise and user adoption.

Overall, the paper suggests that while blockchain holds promise for addressing digital copyright protection issues, it still faces various practical challenges that need to be addressed for successful implementation.

PAPER 04

Title: Innovating in the Music Industry: Blockchain, Streaming and Revenue Capture[4]

Authors: Maria Alice Bosseljon

The paper examines the current state of the music industry, where streaming platforms capture a large share of the revenue while artists receive a smaller portion. It proposes that blockchain technology can be a solution to establish a more equitable and transparent system for royalty distribution. The author investigates how blockchain can be implemented through a survey and interviews with music industry professionals.

The author conducted a survey and interviews with music industry professionals to assess the feasibility of implementing blockchain. While some professionals expressed familiarity with the technology, there are significant obstacles to its adoption.[4] These include a lack of general understanding about blockchain and resistance from established players in the music industry.

Despite these challenges, the research highlights the potential of blockchain to significantly impact the music industry. Further research is recommended to explore how blockchain can be most effectively integrated into the current music ecosystem. This aligns with the ongoing conversation regarding blockchain-based DRM (Digital Rights Management) systems, which aim to create a more balanced approach that protects artist copyrights while providing consumers with access to music.

PAPER 05

Title: Blockchain-Based Multimedia Content Protection: Review and Open Challenges[5]

Authors: Abdul Qureshi and David Megías Jiménez

The paper provides a holistic survey of multimedia content protection applications in which blockchain technology is being used . The authors develop a taxonomy to classify these applications with reference to the technical aspects of blockchain technology, content protection techniques, namely, encryption, digital rights management, digital watermarking and fingerprinting (or transaction tracking), and performance criteria. [5] The study of the literature reveals that there is currently no complete and systematic taxonomy dedicated to blockchain-based copyright protection applications. Moreover, the number of successfully developed blockchain-based content protection systems is very low. [5] This points towards a research gap. To fill this gap, the authors propose a taxonomy that integrates technical aspects and application knowledge and can guide the researchers towards the development of blockchain-based multimedia copyright protection systems. Furthermore, the paper discusses some technical challenges and outlines future research directions.

The key contribution of this paper is to provide a holistic survey of multimedia content protection applications that use the blockchain technology. A taxonomy is developed to classify these applications based on technical blockchain characteristics, content protection mechanisms, and performance criteria. [5] The proposed taxonomy integrates technical aspects and application knowledge and can guide researchers towards the development of blockchain-based multimedia copyright protection systems.

Overall, this paper presents an interesting approach to DRM using blockchain technology. While there are some limitations to this approach, it has the potential to provide a more secure and efficient way of managing digital rights.

PAPER 06

Title: Blockchain-based digital rights management for digital content[6]

Authors: Ramani S, Sri Vishva E, Lakshit Dua, Arya Abrol, and Marimuthu Karuppiah

The paper proposes a blockchain-based digital rights management system. The authors argue that the existing digital rights management systems are not sufficient to protect the copyright of digital content. The proposed system is based on blockchain technology, which provides a secure and tamper-proof way of managing digital rights.

The proposed system is designed to provide a secure and efficient way of managing digital rights. It uses blockchain technology to store the metadata of digital content, such as the author, title, and date of creation. The metadata is stored in a distributed ledger, which makes it difficult for anyone to tamper with the data. The authors also propose a smart contract-based payment system that ensures that the content creators receive fair compensation for their work.

Overall, this paper presents an interesting approach to DRM using blockchain technology. While there are some limitations to this approach, it has the potential to provide a more secure and efficient way of managing digital rights.

PAPER 07

Title: Securing Music Sharing Platforms: A Blockchain-Based Approach [7]

Authors: Adjei-Mensah Isaac, Isaac Osei Agyemang, Collins Sey, Linda Delali Fiasam ,and Abdulhaq Adetunji Salako

The paper discusses the problems with illegal music downloads and copyright infringement. It proposes a system that uses blockchain technology to store music and track copyright information. This system would allow artists to be compensated for their work.

In the past, artists have had difficulty collecting royalties for their music because of illegal music downloads.[7] Blockchain technology could provide a solution to this problem. Blockchain is a distributed ledger that can be used to track ownership of digital assets. In a music sharing platform that uses blockchain, each song would have a unique identifier that would be stored on the blockchain. This would allow anyone to track the ownership of a song and ensure that artists are compensated for their work.[7]

The paper's proposed system would also allow for more efficient royalty payments. Currently, royalty payments can be complex and time-consuming. With blockchain, royalty payments could be automated and distributed directly to artists. This would save time and money for both artists and music platforms.

Chapter 3

System Analysis

3.1 Functional Requirements

- User Registration : The system should have different user roles: Record label, Artist, Users (Listeners), and Commercial Entities. User authentication and authorization should be implemented to ensure that only authorized users can access their respective functionalities.

- Song Management : Record Labels should be able to add songs to the marketplace with details such as name, pricing for personal and commercial use, royalty percentage, genre. All artists associated with a song should be specified upon adding a song by their artist ID. Songs should be uniquely identified by a song ID.

- Song Purchase : Users (both Personal and Commercial) should be able to purchase songs from the marketplace. The system should enforce different pricing for personal and commercial use. Upon purchasing a song, appropriate earnings should be distributed to the record label and associated artists based on the defined royalty percentage. The revenue from each song must be split evenly amongst the artists.

- Royalties Management : Record Labels should have control over royalties payout for all the artists under their them. Artists should be able to track their earnings and view the details of royalties received.

- User Type Validation : Certain actions within the contract should be restricted based on user types (e.g., only record labels can add songs). The system should validate the user type before allowing specific operations.

- Earnings Tracking : The contract should accurately track the earnings of record labels and artists. Earnings should be updated accordingly upon song purchases.
- User Details Retrieval : Users should be able to retrieve their details from the contract. Record Labels should be able to see the songs they have published. Artists should be able to see the songs they are associated with.
- Dashboard : Each user role should have a personalized dashboard displaying relevant information and actions. The dashboard should provide quick access to active songs, contracts, earnings, and other relevant data.

3.2 Non-Functional Requirements

- Security : The contract should implement proper access control to prevent unauthorized actions. Measures should be taken to prevent common vulnerabilities such as reentrancy, integer overflow, etc.
- Scalability : The contract should be designed to handle a potentially large number of users, songs, and transactions. Considerations should be made to ensure scalability without compromising performance.
- Reliability : The contract should be thoroughly tested to ensure reliability and correctness of operations. Error handling and recovery mechanisms should be in place to handle unexpected scenarios gracefully.
- Usability : User interactions with the contract should be intuitive and user-friendly. Error messages should be informative and helpful for users to understand issues.
- Efficiency : Operations within the contract should be optimized for gas efficiency to minimize transaction costs.

3.3 Specific Requirements

Hardware Specifications:

- Operating System : Windows 10 or above / Mac OS.
- RAM : 8GB or above.
- CPU : Intel core i5 9400F and above.
- Memory : 250 GB SSD.
- Screen Resolution : 1280 x 800 minimum.
- Internet : Minimum of 1 Mbps.

Software Specifications:

- A web browser like Google Chrome or Microsoft Edge.
- VS Code or other user-friendly interfaces like Komodo IDE, Mobirise.
- Ganache : A testing blockchain enviroment.
- MetaMask or any other cryptocurrency wallet used to interact with the Ethereum blockchain.

3.4 Use-Case Diagrams and Description

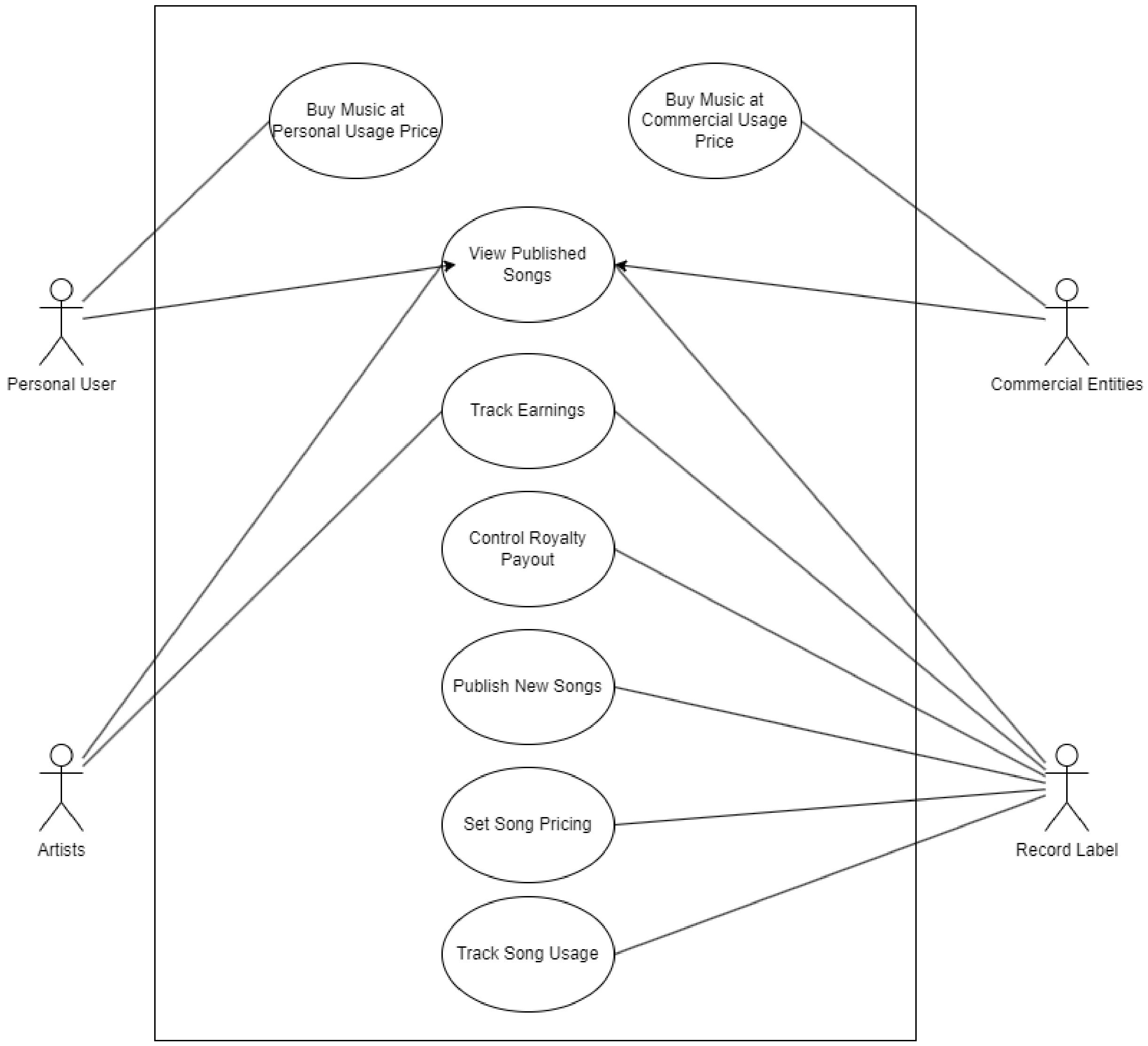


Figure 3.1: Use-Case representation of DRM management system

Here's a description of the main actors and use cases involved:

Actors:

- Record Label: Represents a company or entity responsible for managing and distributing music content.
- Artist: Represents musicians, songwriters, composers, singers, conductors, and bandleaders who create music content.
- Personal User: Represents individuals who purchase and listen to music.

- Commercial Entity: Represents businesses or organizations that purchase music rights for specific purposes like ads, concerts, etc.

Use Cases:

1. View and Track Active Songs (Record Label): The Record Label can view and track all the active songs published under their label. This includes monitoring sales, streams, and other relevant data.

2. Control Royalties Payout (Record Label): Record Labels can control royalties payout for all the different artists under their label. They can specify how earnings are distributed among artists and stakeholders.

3. Set Song Pricing (Record Label): The Record Label decides the pricing of songs for both individual users and commercial entities. This may involve setting base fees and percentage-based pricing.

4. Upload New Song (Record Label): Record Labels can upload new songs to the platform, including song metadata and copyright information.

5. Specify Contracts (Record Label): Record Labels can define contracts with artists, specifying royalty splits for earnings.

6. Track Earnings (Artist): Artists can track their earnings from music sales, royalties, and other revenue streams. This includes accessing detailed earnings reports.

7. View Active Contracts (Artist): Artists can view the details of active contracts with Record Labels, including terms and earnings agreements.

8. Purchase Music (Personal User): Personal Users can purchase music and gain lifetime ownership of the purchased songs.

9. Define Usage Rights (Commercial Entity): Commercial entities can purchase music rights for a for use cases such as in advertisements, events or concerts.

Chapter 4

Analysis Modeling

4.1 Data Modeling

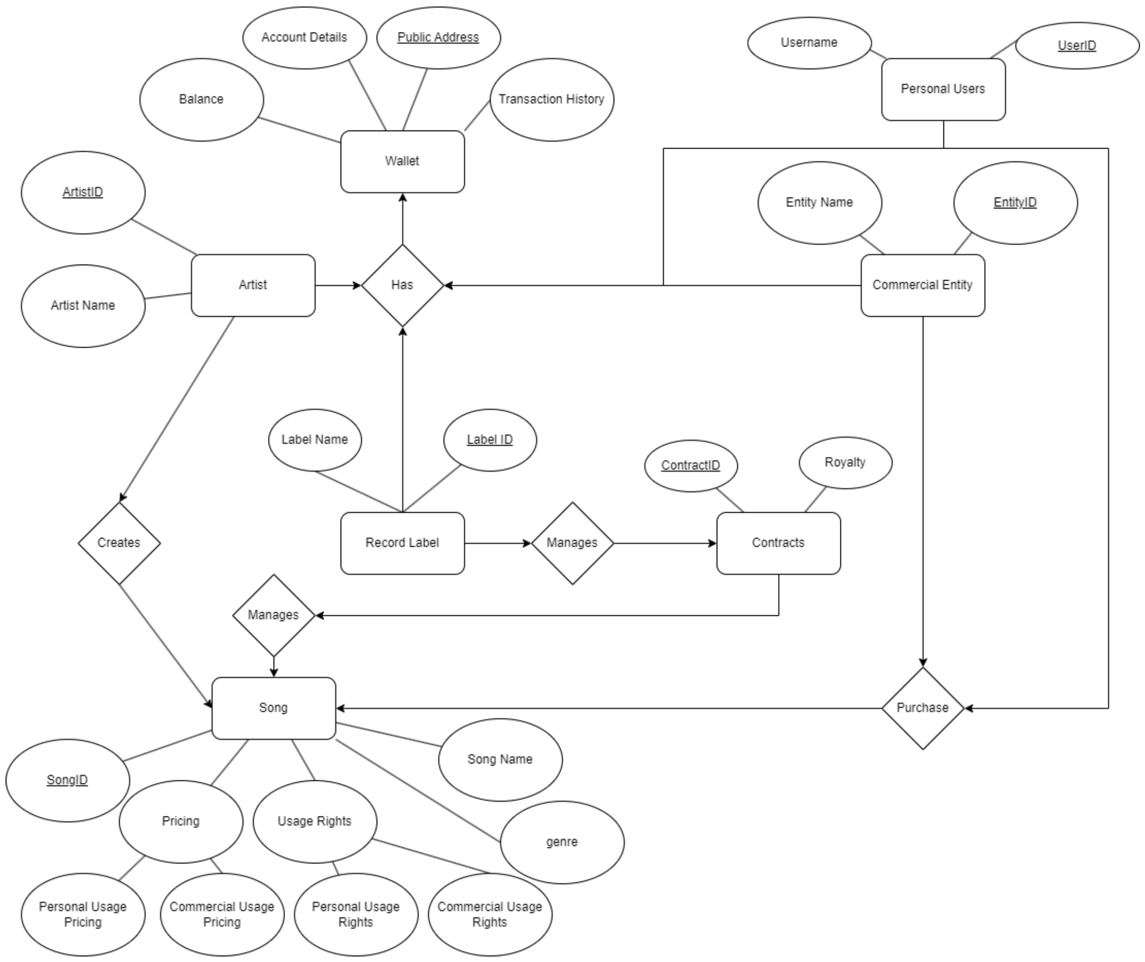


Figure 4.1: ER diagram of DRM management system

The provided Entity-Relationship (ER) diagram outlines the structure of the Blockchain-Based DRM Management System. The system manages a network of interconnected entities, including artists, record labels, commercial entities, users, and digital assets such as songs. The key components of the system are as follows:

- Accounts and Wallets: Each user in the system is associated with an account that holds their public address, balance, and transaction history. Wallets are used to store and manage digital assets.
- Artists and Record Labels: Artists create digital assets and are associated with a unique ArtistID. Record labels, represented by Label Name and Label ID, manage the digital rights to these assets.
- Personal and Commercial Entities: Personal and Commercial entities, identified by Entity Name and EntityID, purchase digital rights.
- Digital Asset Details: Digital assets, such as songs, are represented by Song Name and Song ID. They have associated pricing and usage rights, which dictate how they can be used.
- Royalties and Payments: The system automates royalty payments to artists, songwriters, publishers, and record labels through smart contracts, ensuring fair compensation.

This system leverages blockchain technology to secure and decentralize data, making it resistant to tampering and piracy. This ensures that all artists receive their due royalties, contributing to fair compensation for content creators.

4.2 Activity Diagram

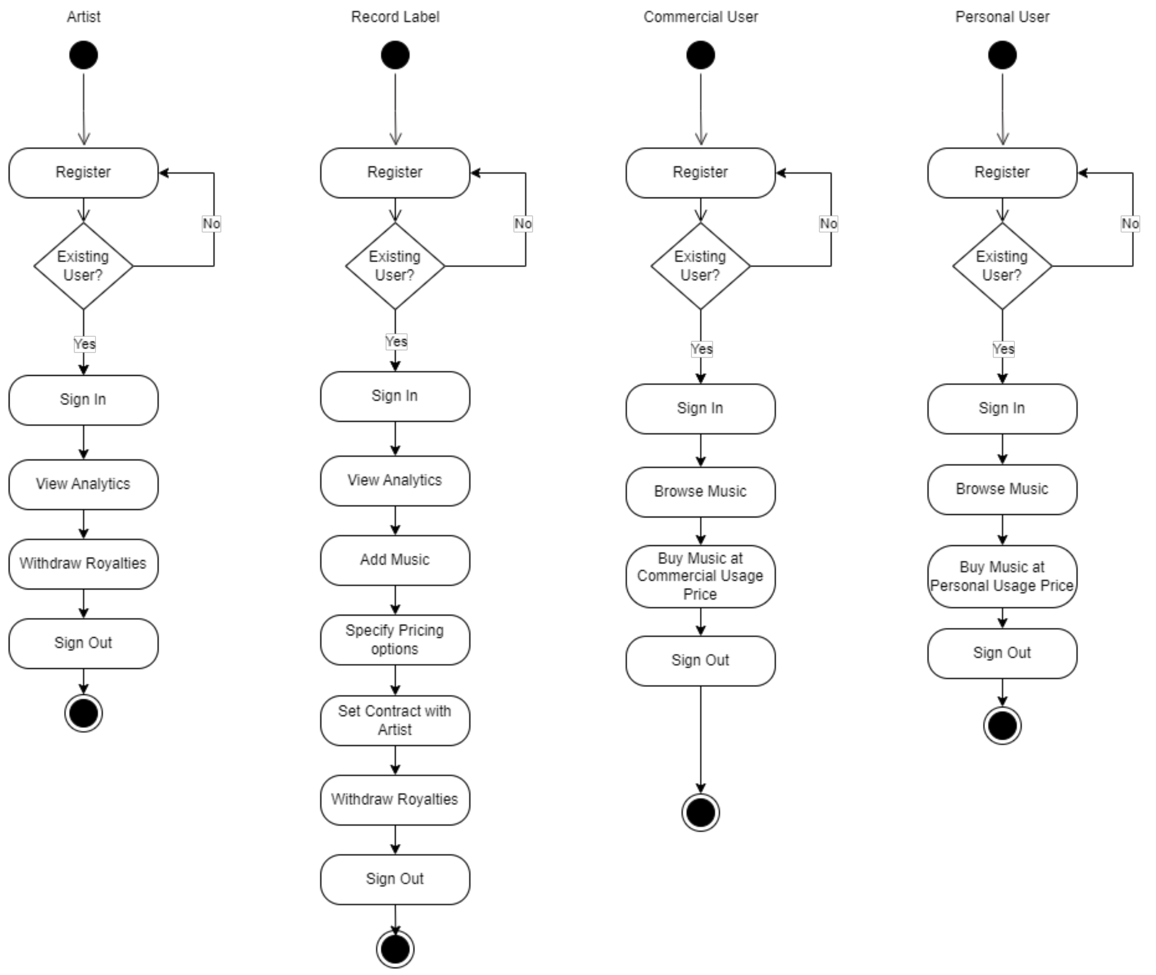


Figure 4.2: Activity diagram of DRM management system

4.3 Functional Modeling

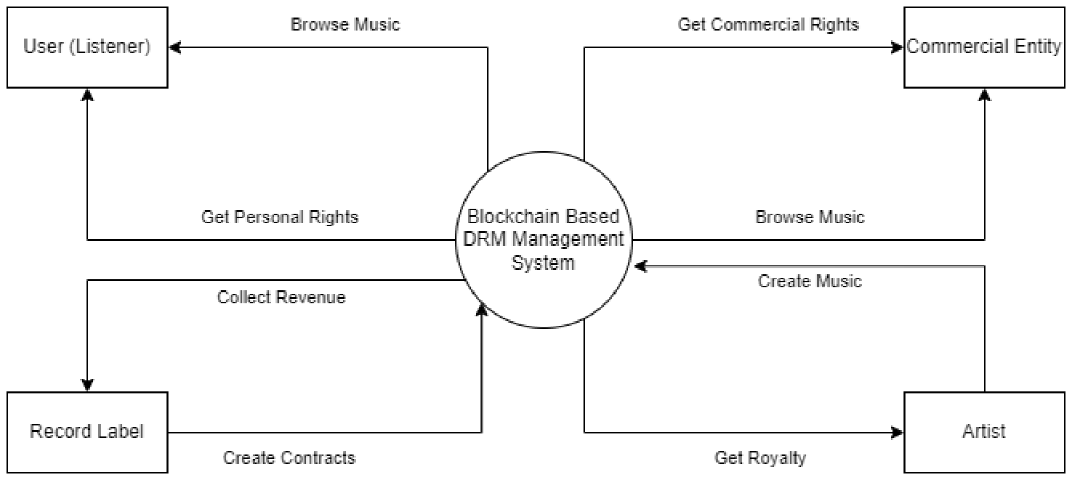


Figure 4.3: DFD Level 0 of DRM management system

In the DFD level 0, shows a general overview of the system. It descrives how different entities interact with the system.

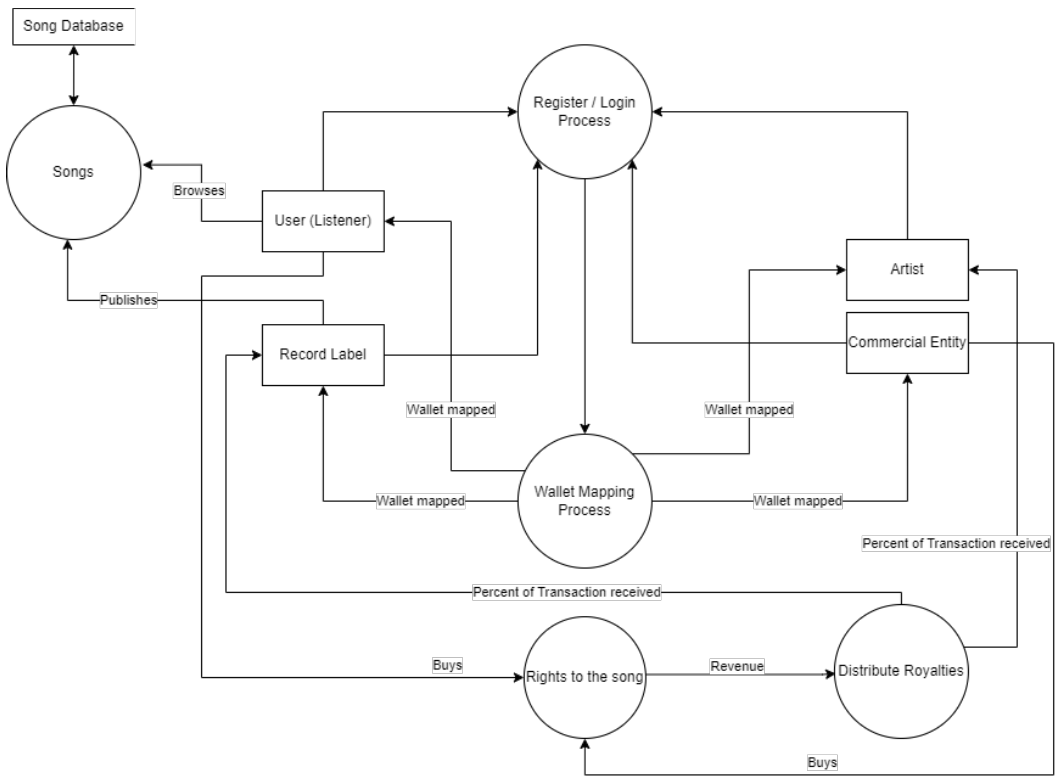


Figure 4.4: DFD Level 1 of DRM management system

The DFD level 1 above,gives a more wider perspective of the interaction of User, Artist, Record Label and Commercial Entity.

4.4 TimeLine Chart

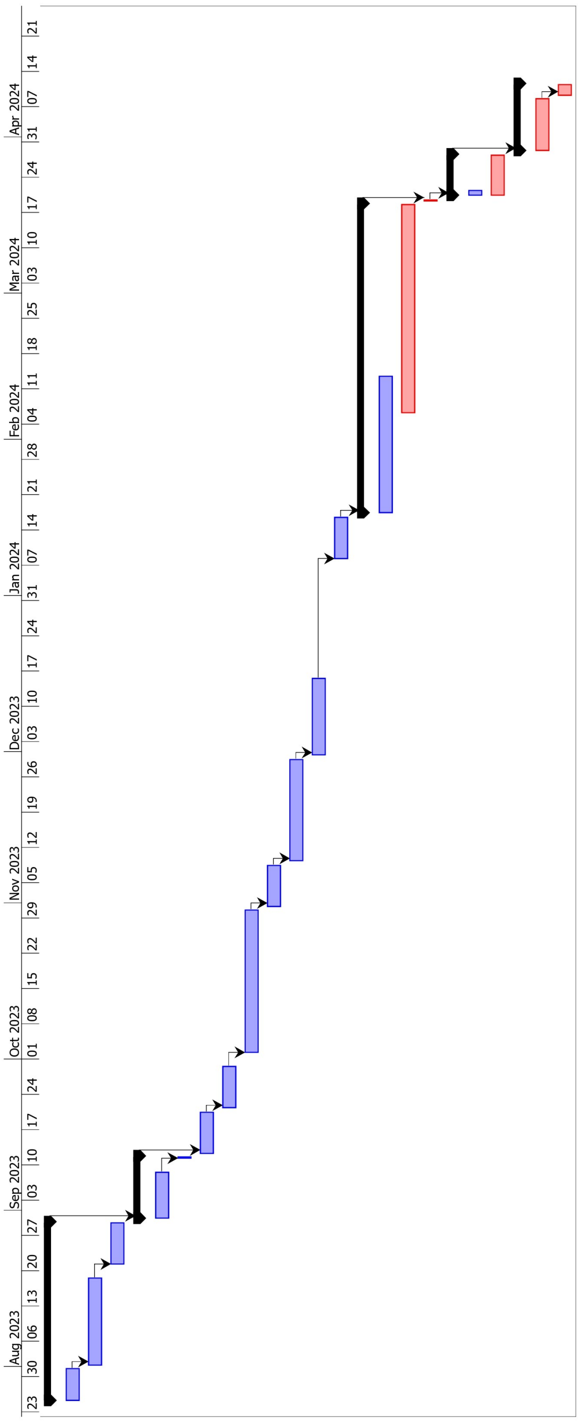


Figure 4.5: Gantt Chart From July 2023 - April 2024

Chapter 5

Design

5.1 Architectural Design

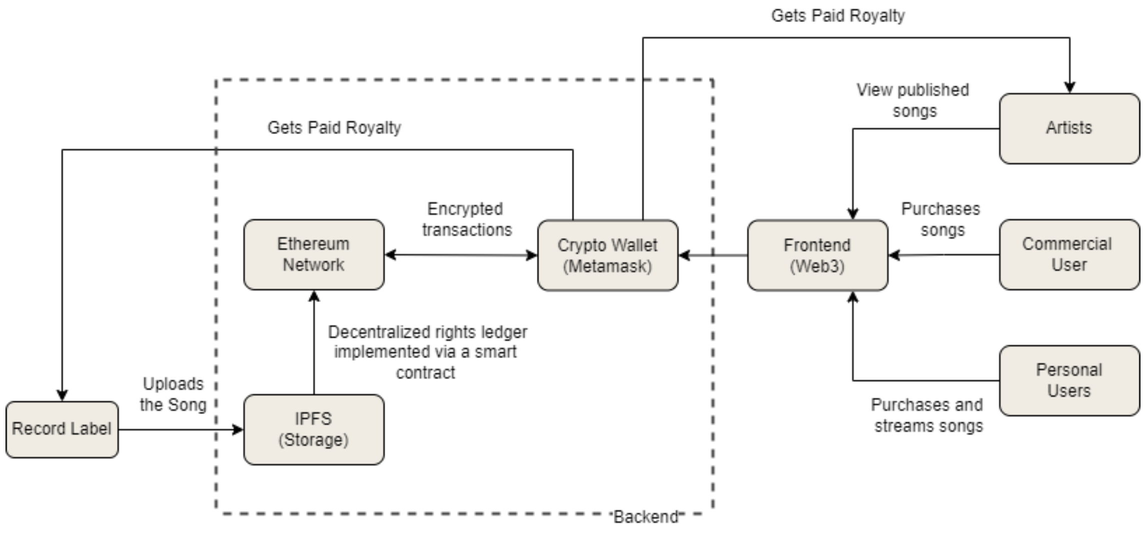


Figure 5.1: Block Diagram of DRM management system

The blockchain-based DRM management system has been designed with the following design principles in mind:

- **User Roles**: The system allows for different user roles with their own functionalities and privileges :
  - Personal Users: These users represent individual customers who buy the songs for personal use. They can buy songs and access details of their purchased songs.

  - Commercial Users: Commercial users in our context are entities such as businesses or organizations that purchase songs for commercial use, such as for the use of music at events, malls, and other commercial avenues.
  - Record labels: Record labels are the entities responsible for managing and distributing songs. They have the authority to add new songs to the marketplace and track earnings from song sales.
  - Artists: Artists are creators of the music available on the platform. They earn royalties from song sales and can access details about their earnings and associated songs.

- **Contract Structure**: The smart contract is structured based on the following key components:
  - User Structs: These structs define the properties of each user type, such as userID, wallet addresses, and earnings.
  - Song Structs: These contain information about each song, including its name, pricing for personal and commercial use, royalty percentage, and associated artists.
  - Purchase Counts: This struct keeps track of the number of times a song has been purchased, distinguishing between personal and commercial purchases.
  - Mappings: Various mappings connect users to their respective IDs and roles, songs to their record labels and artists, and track song purchase counts.
  - Modifiers and Functions: Modifiers ensure that only users with the appropriate roles can execute certain functions, such as adding songs or purchasing them. Functions handle user actions like adding users, songs, and purchasing songs, while also facilitating earnings distribution.

- **Transaction Processes**: The interaction between users and the contract follows a set of defined processes:
  - User Registration: Users register on the platform by specifying their role as personal, commercial, record label, or artist. Each user has a unique wallet address.
  - Song Addition: Record labels add new songs to the marketplace by providing details such as song name, pricing, royalties, and associated artists. Upon addition, songs become available for purchase.
  - Song Purchase: Personal and commercial users can buy songs by sending the required payment to the contract. The contract verifies the user's role and ensures they have sufficient funds before completing the purchase. Earnings are distributed to the record label and artists based on the specified royalty percentages.
  - Earnings Distribution: When a song is purchased, the contract calculates and distributes earnings to the record label and the song's associated artists based on the defined royalty percentages.
  - Accessing Details: Users are able to access various details such as their personal information, purchased songs, earnings, and the number of times a song has been purchased.

The blockchain-based DRM management system works as follows:

1. User Registration: Users register with specific roles (personal, commercial, record label, or artist) to access different features and permissions within the platform.

2. Song Upload: Record labels upload songs to the platform, providing details such as song name, pricing for personal and commercial use, royalty percentage for artists, and artist information.

3. Library Management: Uploaded songs are organized into a catalog, segmented by user type. Users can browse and access songs based on their registered role.

4. Song Purchase: Users initiate song purchases using cryptocurrency. The system deducts the appropriate amount from their wallet; based on their userType; and distributes revenue among record labels and artists according to predefined royalty percentages.

5. Revenue Distribution: Record labels receive a share of revenue from song purchases, while artists earn royalties based on their contributions to the song.

6. Sales Tracking: The system transparently tracks sales and revenue using blockchain, ensuring an immutable and auditable record of transactions.

7. User Access and Streaming: Users can access purchased songs for streaming, providing a seamless experience for enjoying music within the platform.

Chapter 6

Implementation

6.1 Functions Used

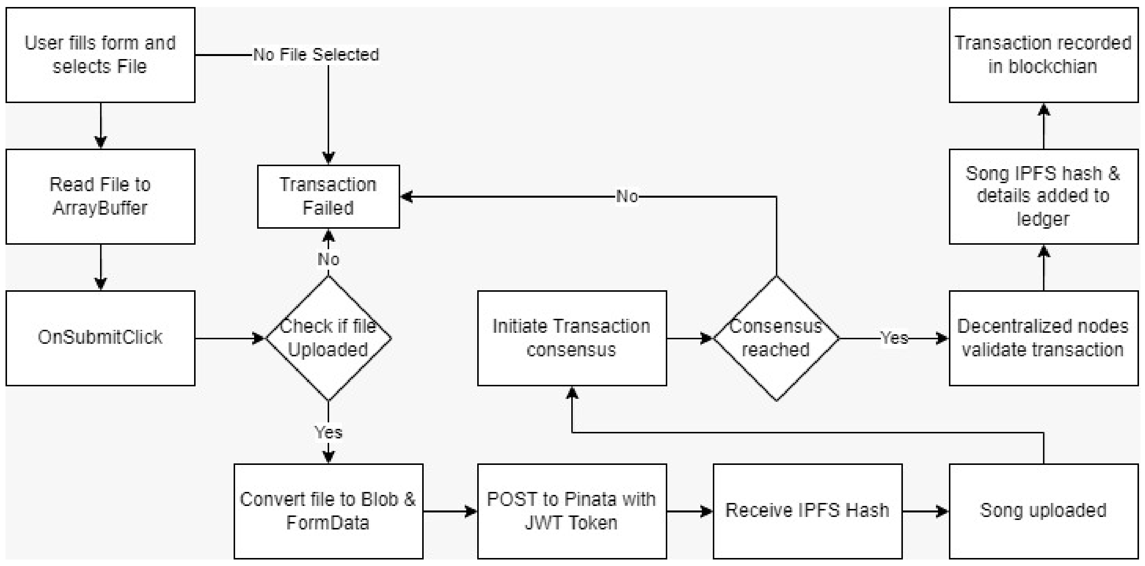

User fills form and selects File
No File Selected
Transaction recorded in blockchian
Read File to ArrayBuffer
Transaction Failed
No
Song IPFS hash & details added to ledger
No
OnSubmitClick
Check if file Uploaded
Initiate Transaction consensus
Consensus reached
Yes
Decentralized nodes validate transaction
Yes
Convert file to Blob & FormData
POST to Pinata with JWT Token
Receive IPFS Hash
Song uploaded

Figure 6.1: How Add Songs works

Working of AddSong Function:

1. **Song Add Button Clicked**: The user interacts with the AddSongCard React component, which is a form for adding a new song. The form includes fields for the song's name, genre, personal cost, commercial cost, royalty percent, artist IDs, and a file upload input for the song file.

2. **Song Details Captured**: The user fills in the form with the song's details. The user selects a song file to upload. The captureFile function is triggered

when the user selects a file. This function reads the file as an ArrayBuffer and stores it in the component's state.

3. **Song File Uploaded to IPFS**: When the user submits the form, the onSubmitClick function is triggered. The function first checks if a file has been uploaded. If so, it converts the file to a Blob and appends it to a FormData object. The FormData object is then sent to Pinata's API using an HTTP POST request. The request includes the file and an authorization header with a JWT token for authentication. Pinata's API responds with the IPFS hash of the uploaded file.

4. **Song Added to Ledger**: The onSubmitClick function then interacts with the MusicMarketplace smart contract. It calls the addSong function of the smart contract, passing in the song's details (name, genre, personal cost, commercial cost, royalty percent, artist IDs, IPFS hash, and genre). The smart contract adds the song to its ledger, including the song's details and the IPFS hash.

5. **Song Added to IPFS**: The song file has already been uploaded to IPFS in step 3. The IPFS hash of the uploaded file is stored in the smart contract's ledger, allowing users to access the song file directly from IPFS.

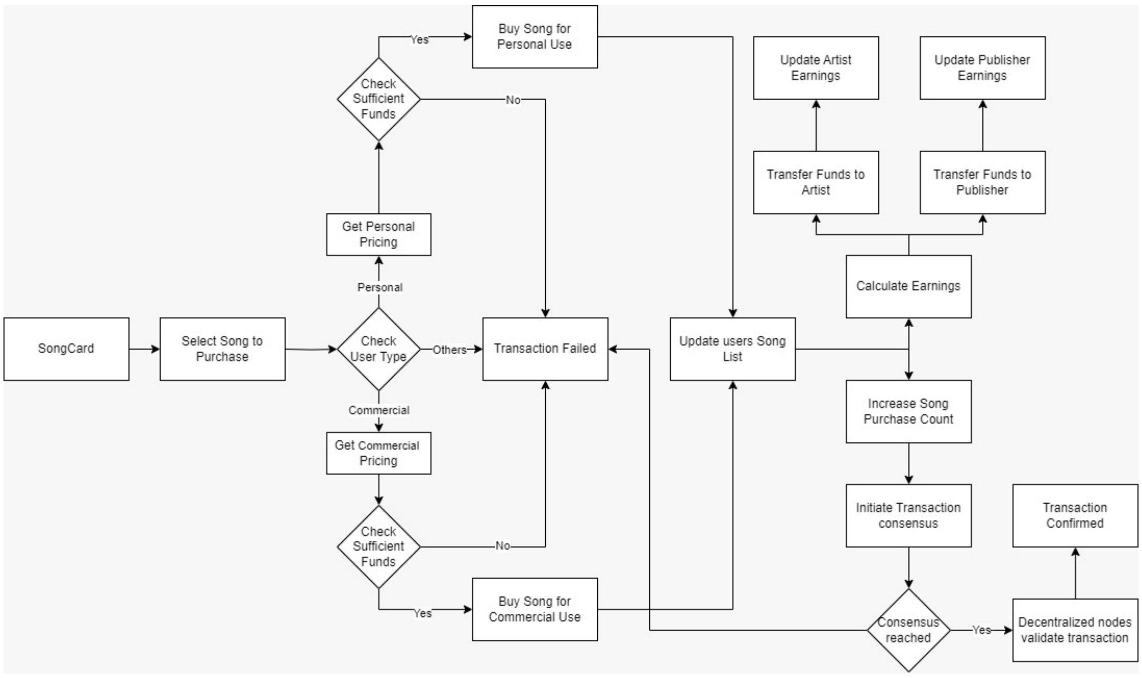


Figure 6.2: How Buy Song works

Working of BuySong Function:

1. **Buy Song Button Clicked**: The user interacts with the SongCard React component, which displays the details of a song available for purchase. The component includes a ”Buy Song” button.

2. **Ownership Check**: Upon clicking the ”Buy Song” button, the frontend checks whether the song is already owned by the user. If the song is already owned, the button display may change to indicate that the song is already purchased.

3. **Metamask Confirmation**: After confirming the purchase by clicking the ”Buy Song” button, Metamask prompts the user to confirm the transaction.

4. **Transaction Execution**: Once the user confirms the transaction through Metamask, the frontend sends a request to the smart contract function buySong with the song ID and the appropriate value for the purchase. The smart contract executes the transaction, deducting the payment from the user’s account and transferring it to the publisher and artists according to the defined royalty percentages.

5. **Update User's Song Ledger**: After a successful transaction, the song ID is added to the user's list of owned songs in the ledger stored in the respective user struct (Personal or Commercial).

6. **Confirmation and Update**: Upon successful completion of the transaction, the frontend displays a confirmation message to the user indicating that the song has been successfully purchased and added to their collection. Any relevant UI elements are updated to reflect the new ownership status of the song.

6.2 Working of the project

MusicMarketplace.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract MusicMarketplace {
    enum UserType {
        Unknown,
        Personal,
        Publisher,
        Artist,
        Commercial
    }

    struct Personal {
        uint256 personalID;
        address walletAddress;
        string personalName;
        uint256[] songIDs; // Array to store IDs of purchased songs
    }

    struct Commercial {
        uint256 commercialID;
        address walletAddress;
        string commercialName;
        uint256[] songIDs; // Array to store IDs of purchased songs
    }

    struct Publisher {
        uint256 publisherID;
        address walletAddress;
        string publisherName;
        uint256 totalEarnings;
        uint256[] songsPublished; // Array to store IDs of songs
            published by the publisher
    }

    struct Artist {
        uint256 artistID;
        address walletAddress;
        string artistName;
        UserType userType;
        uint256 totalEarnings; // Total earnings for the artist
        uint256[] songsArtist; // Array to store IDs of songs
            associated with the artist
    }

    struct Song {
        uint256 songID;
        uint256 publisherID;
        string songName;
        uint256 personalPricing; // Renamed from songPrice to
            personalPricing
        uint256 commercialPricing; // Additional pricing for
            commercial use
        uint256 royaltyPercent;
        string songHash;
        string genre;
        uint256[] artistIDs; // Array to store artist IDs
    }

    struct PurchaseCounts {
        uint256 personalCount;
        uint256 commercialCount;
    }

    uint256 public nextPersonalID = 1;
    uint256 public nextCommercialID = 1;
    uint256 public nextPublisherID = 1;
    uint256 public nextArtistID = 1;
    uint256 public nextSongID = 1;

    mapping(uint256 => Personal) public personals;
    mapping(uint256 => Commercial) public commercials;
    mapping(uint256 => Publisher) public publishers;
    mapping(uint256 => Artist) public artists;
    mapping(uint256 => Song) public songs;
    mapping(address => UserType) public userTypes;
    mapping(address => uint256) public publisherIDsByWallet;
    mapping(address => uint256) public artistIDsByWallet; // Mapping
       for artists
    mapping(uint256 => PurchaseCounts) public timesSongPurchased;

    uint256 public songIDTracker = 0;

    modifier onlyPersonal() {
        require(
            userTypes[msg.sender] == UserType.Personal,
            "Only personal users can perform this action"
        );
        _;
    }

    modifier onlyCommercial() {
        require(
            userTypes[msg.sender] == UserType.Commercial,
            "Only commercial users can perform this action"
        );
        _;
    }

    function addUserType(UserType _userType) internal {
        userTypes[msg.sender] = _userType;
    }

    function addPersonal(string memory _personalName) public {
        personals[nextPersonalID] = Personal(
            nextPersonalID,
            msg.sender,
            _personalName,
            new uint256[](0)
        );
        addUserType(UserType.Personal);
        nextPersonalID++;
    }

    function addCommercial(string memory _commercialName) public {
        commercials[nextCommercialID] = Commercial(
            nextCommercialID,
            msg.sender,
            _commercialName,
            new uint256[](0)
        );
        addUserType(UserType.Commercial);
        nextCommercialID++;
    }

    function addPublisher(string memory _publisherName) public {
        uint256 newPublisherID = nextPublisherID++;
        publishers[newPublisherID] = Publisher(
            newPublisherID,
            msg.sender,
            _publisherName,
            0,
            new uint256[](0)
        );

        // Store the mapping between the wallet address and the publisher ID
        publisherIDsByWallet[msg.sender] = newPublisherID;

        addUserType(UserType.Publisher);
    }

    function addArtist(string memory _artistName) public {
        artists[nextArtistID] = Artist(
            nextArtistID,
            msg.sender,
            _artistName,
            UserType.Artist,
            0,
            new uint256[](0)
        );
        addUserType(UserType.Artist);
        artistIDsByWallet[msg.sender] = nextArtistID; // Update mapping for artists
        nextArtistID++;
    }

    function addSong(
        string memory _songName,
        uint256 _personalPricing,
        uint256 _commercialPricing,
        uint256[] memory _artistIDs,
        uint256 _royaltyPercent,
        string memory _songHash,
        string memory _genre
    ) public {
        require(
            userTypes[msg.sender] == UserType.Publisher,
            "Only publishers can add songs"
        );
        uint256 publisherID = publisherIDsByWallet[msg.sender];
        require(publisherID > 0, "Publisher not found"); // Ensure the publisher exists
        //          // Check if the song hash has been used before
        // require(!songHashes[_songHash], "Song hash already used");

        uint256 newSongID = nextSongID++;
        songs[newSongID] = Song(
            newSongID,
            publisherID,
            _songName,
            _personalPricing,
            _commercialPricing,
            _royaltyPercent,
            _songHash,
            _genre,
            _artistIDs
        );
        //          // Mark the song hash as used
        // songHashes[_songHash] = true;

        // Update songsPublished array for the publisher
        publishers[publisherID].songsPublished.push(newSongID);

        // Update songsArtist array for each artist associated with the song
        for (uint256 i = 0; i < _artistIDs.length; i++) {
            artists[_artistIDs[i]].songsArtist.push(newSongID);
        }

        // Initialize the purchase count for the new song to 0
        timesSongPurchased[newSongID] = PurchaseCounts({
            personalCount: 0,
            commercialCount: 0
        });

        songIDTracker++; // Increment songIDTracker when a song is added
    }

    function checkUserType() public view returns (UserType) {
        return userTypes[msg.sender];
    }

    function getPublisherEarnings(
        uint256 _publisherID
    ) public view returns (uint256) {
        return publishers[_publisherID].totalEarnings;
    }

    function getArtistEarnings(
        uint256 _artistID
    ) public view returns (uint256) {
        return artists[_artistID].totalEarnings;
    }

    function buySong(uint256 _songID) public payable {
        require(_songID < nextSongID, "Invalid song ID");
        Song storage song = songs[_songID];

        uint256 userID;
        bool isPersonal;
        if (userTypes[msg.sender] == UserType.Personal) {
            require(
                msg.value >= song.personalPricing,
                "Insufficient funds to purchase the song."
            );
            userID = getPersonalIDByWallet(msg.sender);
            require(userID > 0, "Personal user not found");
            isPersonal = true;
        } else if (userTypes[msg.sender] == UserType.Commercial) {
            require(
                msg.value >= song.commercialPricing,
                "Insufficient funds for commercial use"
            );
            userID = getCommercialIDByWallet(msg.sender);
            require(userID > 0, "Commercial user not found");
            isPersonal = false;
        } else {
            revert("Only personal and commercial users can buy songs"
                );
        }

        // Check if the song is already purchased by the user
        bool songAlreadyPurchased = false;
        for (
            uint256 i = 0;
            i <
            (
                isPersonal
                    ? personals[userID].songIDs.length
                    : commercials[userID].songIDs.length
            );
            i++
        ) {
            if (
                isPersonal
                    ? personals[userID].songIDs[i] == _songID
                    : commercials[userID].songIDs[i] == _songID
            ) {
                songAlreadyPurchased = true;
                break;
            }
        }

        require(!songAlreadyPurchased, "Song already purchased by the
            user");

        // If not purchased, add the SongID to the SongIDs array of
           the user
        if (isPersonal) {
            personals[userID].songIDs.push(_songID);
        } else {
            commercials[userID].songIDs.push(_songID);
        }

        // Calculate earnings and perform transfers
        uint256 price = isPersonal
            ? song.personalPricing
            : song.commercialPricing;
        uint256 publisherShare = (price * (100 - song.royaltyPercent)
           ) / 100;
        uint256 artistShare = price - publisherShare;
        uint256 artistSharePerArtist = artistShare / song.artistIDs.
           length;

        // Update earnings for the publisher
        publishers[song.publisherID].totalEarnings += publisherShare;
        // Update earnings for the artists
        for (uint256 i = 0; i < song.artistIDs.length; i++) {
            artists[song.artistIDs[i]].totalEarnings +=
               artistSharePerArtist;
        }
        // Transfer funds to publisher
        (bool publisherTransferResult, ) = payable(
            publishers[song.publisherID].walletAddress
        ).call{value: publisherShare}("");
        require(publisherTransferResult, "Transfer to publisher
           failed");

        // Transfer funds to artists
        for (uint256 i = 0; i < song.artistIDs.length; i++) {
            (bool artistTransferResult, ) = payable(
                artists[song.artistIDs[i]].walletAddress
            ).call{value: artistSharePerArtist}("");
            require(artistTransferResult, "Transfer to artist failed"
               );
        }

        // Increment the appropriate count in the timesSongPurchased
           mapping
        if (isPersonal) {
            timesSongPurchased[_songID].personalCount += 1;
        } else {
            timesSongPurchased[_songID].commercialCount += 1;
        }
    }

    function getPersonalDetails()
        public
        view
        onlyPersonal
        returns (
            uint256 personalID,
            address walletAddress,
            string memory personalName,
            uint256[] memory songIDs
        )
    {
        uint256 foundPersonalID = getPersonalIDByWallet(msg.sender);
        Personal storage personal = personals[foundPersonalID];

        return (
            foundPersonalID,
            personal.walletAddress,
            personal.personalName,
            personal.songIDs
        );
    }

    function getPersonalIDByWallet(
        address _walletAddress
    ) internal view returns (uint256) {
        return
            userTypes[_walletAddress] == UserType.Personal
                ? _getPersonalID(_walletAddress)
                : 0;
    }

    function getCommercialIDByWallet(
        address _walletAddress
    ) internal view returns (uint256) {
        return
            userTypes[_walletAddress] == UserType.Commercial
                ? _getCommercialID(_walletAddress)
                : 0;
    }

    function _getCommercialID(
        address _walletAddress
    ) internal view returns (uint256) {
        for (uint256 i = 1; i < nextCommercialID; i++) {
            if (commercials[i].walletAddress == _walletAddress) {
                return i;
            }
        }
        return 0;
    }

    function _getPersonalID(
        address _walletAddress
    ) internal view returns (uint256) {
        for (uint256 i = 1; i < nextPersonalID; i++) {
            if (personals[i].walletAddress == _walletAddress) {
                return i;
            }
        }
        return 0;
    }

    function getCommercialDetails()
        public
        view
        onlyCommercial
        returns (
            uint256 commercialID,
            address walletAddress,
            string memory commercialName,
            uint256[] memory songIDs
        )
    {
        uint256 foundCommercialID = getCommercialIDByWallet(msg.
            sender);
        Commercial storage commercial = commercials[foundCommercialID
            ];

        return (
            foundCommercialID,
            commercial.walletAddress,
            commercial.commercialName,
            commercial.songIDs
        );
    }

    function getSongDetails(
        uint256 _songID
    )
        public
        view
        returns (
            uint256 songID,
            uint256 publisherID,
            string memory songName,
            uint256 personalPricing,
            uint256 commercialPricing,
            uint256 royaltyPercent,
            string memory songHash,
            string memory genre,
            uint256[] memory artistIDs
        )
    {
        require(_songID < nextSongID, "Invalid song ID");
        Song storage song = songs[_songID];
        return (
            song.songID,
            song.publisherID,
            song.songName,
            song.personalPricing,
            song.commercialPricing,
            song.royaltyPercent,
            song.songHash,
            song.genre,
            song.artistIDs
        );
    }

    function getSongPurchaseCounts(
        uint256 _songID
    ) public view returns (uint256[] memory) {
        require(_songID < nextSongID, "Invalid song ID");
        PurchaseCounts memory counts = timesSongPurchased[_songID];
        uint256[] memory purchaseCountArray = new uint256[](2);
        purchaseCountArray[0] = counts.personalCount;
        purchaseCountArray[1] = counts.commercialCount;
        return purchaseCountArray;
    }

    function getPublisherDetails()
        public
        view
        returns (
            uint256 publisherID,
            address walletAddress,
            string memory publisherName,
            uint256[] memory songsPublished
        )
    {
        // require(
        //     userTypes[msg.sender] == UserType.Publisher ,
        //     "Only publishers can access their details"
        // );

        uint256 foundPublisherID = publisherIDsByWallet[msg.sender];
        Publisher storage publisher = publishers[foundPublisherID];

        return (
            publisher.publisherID,
            publisher.walletAddress,
            publisher.publisherName,
            publisher.songsPublished
        );
    }

    function getArtistDetails()
        public
        view
        returns (
            uint256 artistID,
            address walletAddress,
            string memory artistName,
            uint256[] memory songsArtist
        )
    {
        require(
            userTypes[msg.sender] == UserType.Artist,
            "Only artists can access their details"
        );

        uint256 foundArtistID = artistIDsByWallet[msg.sender];
        Artist storage artist = artists[foundArtistID];

        return (
            artist.artistID,
            artist.walletAddress,
            artist.artistName,
            artist.songsArtist
        );
    }

    function getNumSongs() public view returns (uint256) {
        return songIDTracker - 1;
    }

    function getPublisherNameByID(
        uint256 _publisherID
    ) public view returns (string memory publisherName) {
        // Check if the publisherID exists in the publishers mapping
        require(
            publishers[_publisherID].publisherID != 0,
            "Publisher ID does not exist"
        );
        // Retrieve the publisher details using the publisherID
        Publisher storage publisher = publishers[_publisherID];
        return publisher.publisherName;
    }

    function getArtistNameByID(
        uint256 _artistID
    ) public view returns (string memory artistName) {
        // Check if the artistID exists in the artists mapping
        require(artists[_artistID].artistID != 0, "Artist ID does not
            exist");
        // Retrieve the artist details using the artistID
        Artist storage artist = artists[_artistID];
        return artist.artistName;
    }

    function getAllArtists()
        public
        view
        returns (uint256[] memory, string[] memory)
    {
        uint256[] memory artistIDs = new uint256[](nextArtistID - 1);
        string[] memory artistNames = new string[](nextArtistID - 1);
        // Iterate through all artists and populate the arrays
        for (uint256 i = 1; i < nextArtistID; i++) {
                artistIDs[i - 1] = i;
                artistNames[i - 1] = artists[i].artistName;
            }
            return (artistIDs, artistNames);
        }
}
```

6.3 Results and Discussions

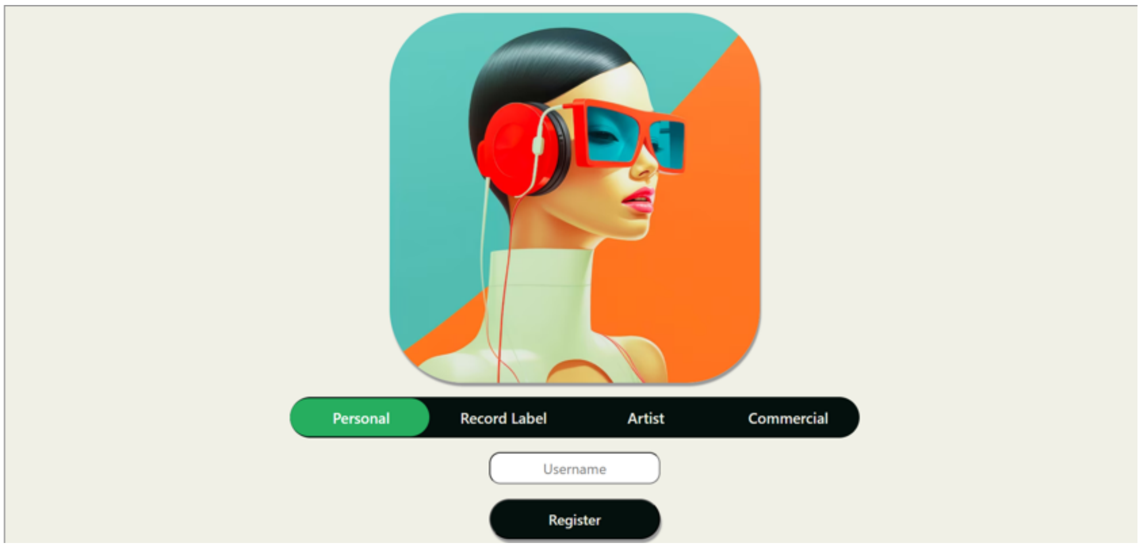
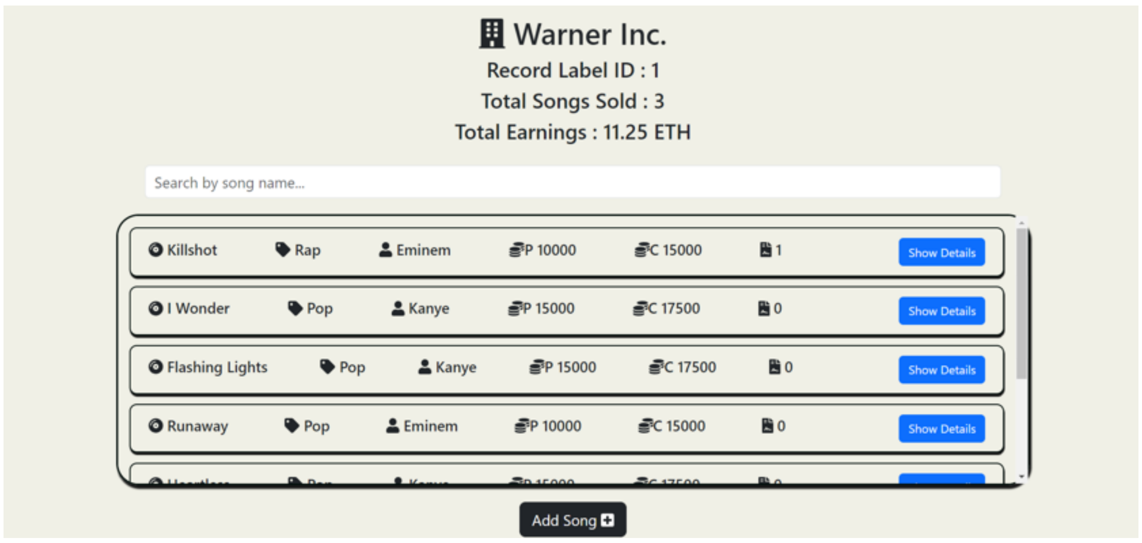


Figure 6.3: Login Page



Warner Inc.
Record Label ID : 1
Total Songs Sold : 3
Total Earnings : 11.25 ETH

Search by song name...

Killshot	Rap	Eminem	P 10000	C 15000	1	Show Details
I Wonder	Pop	Kanye	P 15000	C 17500	0	Show Details
Flashing Lights	Pop	Kanye	P 15000	C 17500	0	Show Details
Runaway	Pop	Eminem	P 10000	C 15000	0	Show Details

Add Song

Figure 6.4: Home Page for Record Label

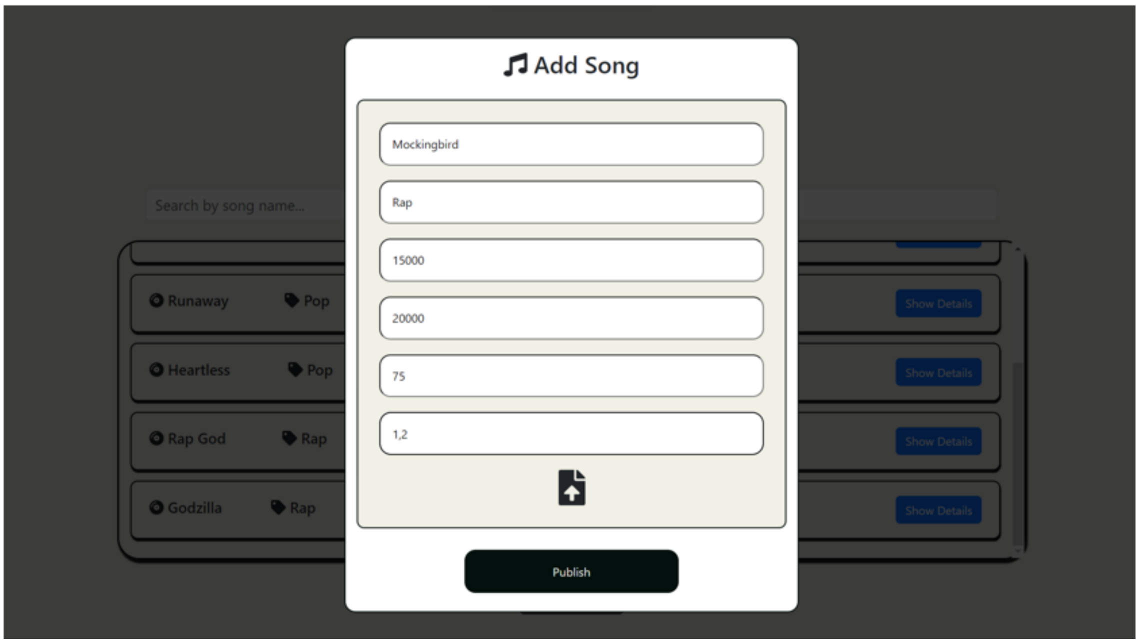
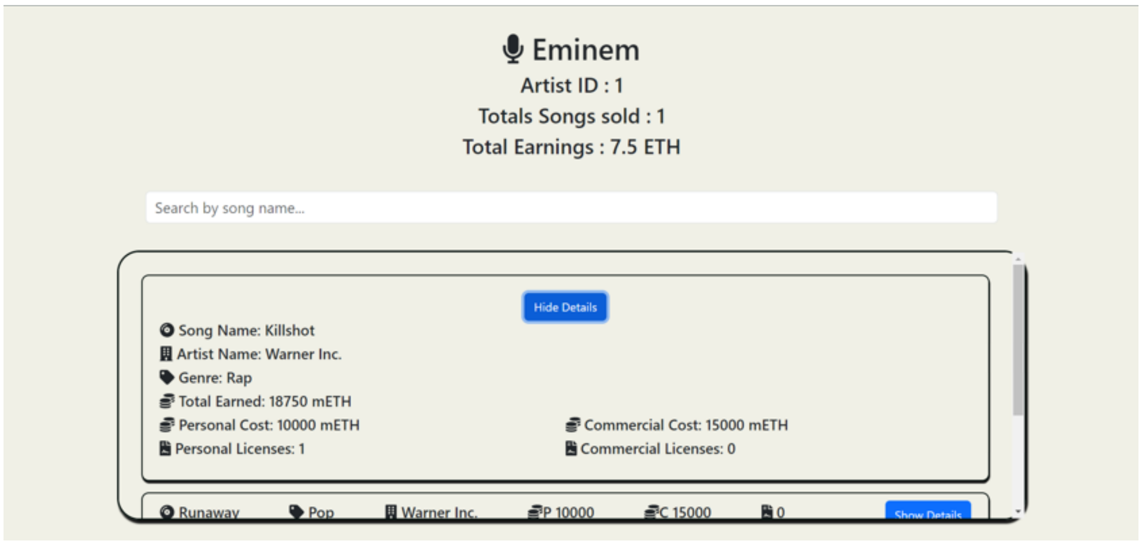


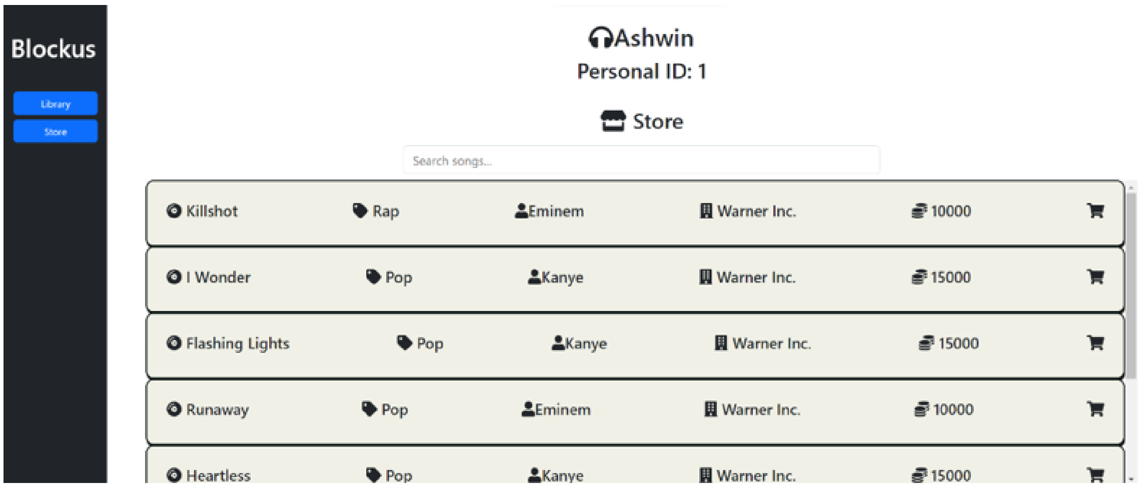
Figure 6.5: Adding Songs for Record Label


Eminem
Artist ID : 1
Totals Songs sold : 1
Total Earnings : 7.5 ETH

Song Name: Killshot
Artist Name: Warner Inc.
Genre: Rap
Total Earned: 18750 mETH
Personal Cost: 10000 mETH
Personal Licenses: 1
Commercial Cost: 15000 mETH
Commercial Licenses: 0

Runaway Pop Warner Inc. P 10000 C 15000 0

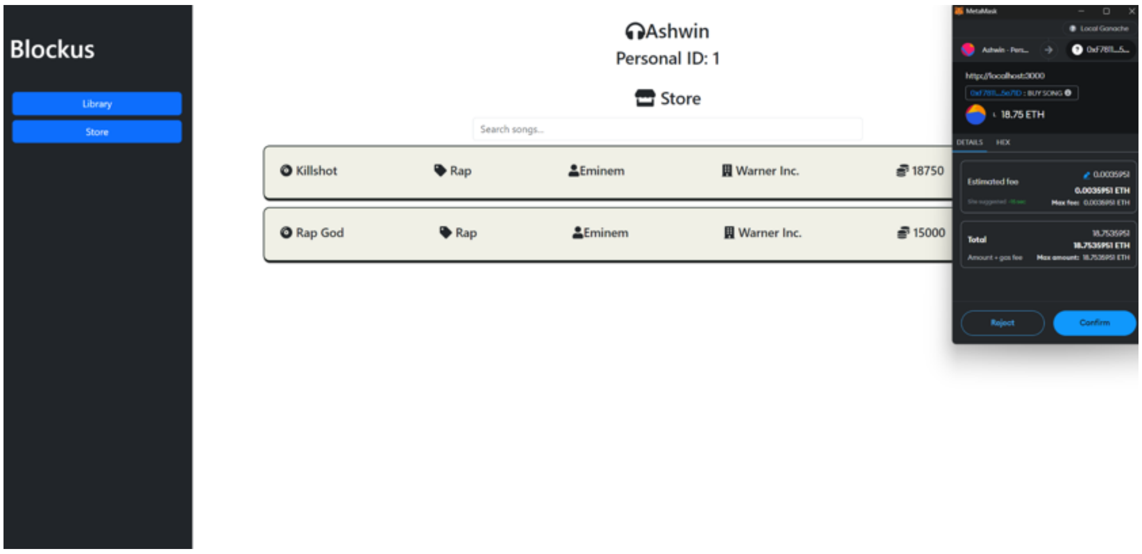
Figure 6.6: Home Page for Artist


Blockus

Ashwin
Personal ID: 1
Store

Killshot	Rap	Eminem	Warner Inc.	10000
I Wonder	Pop	Kanye	Warner Inc.	15000
Flashing Lights	Pop	Kanye	Warner Inc.	15000
Runaway	Pop	Eminem	Warner Inc.	10000
Heartless	Pop	Kanye	Warner Inc.	15000

Figure 6.7: Store Page for Personal Users


Blockus

Ashwin
Personal ID: 1
Store

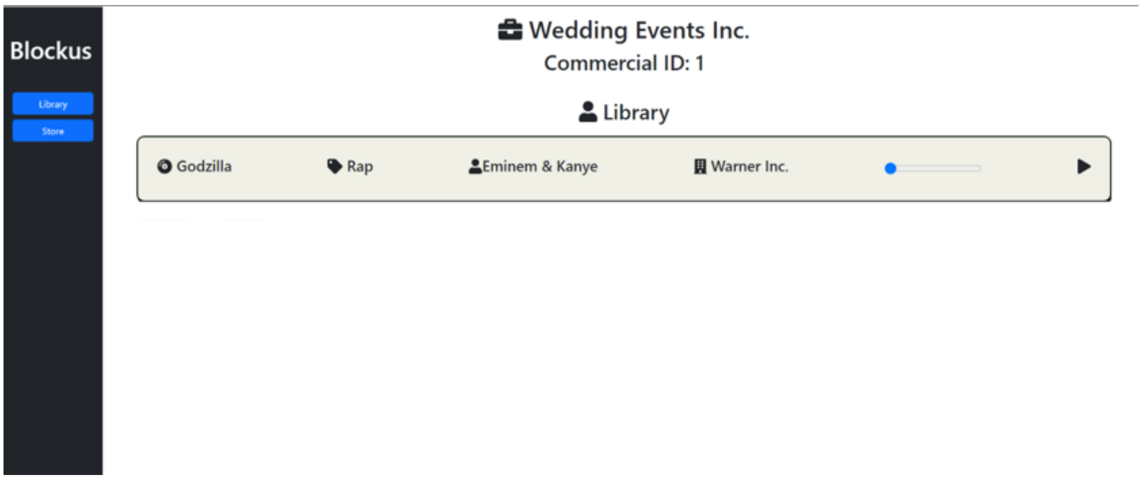
Killshot	Rap	Eminem	Warner Inc.	18750
Rap God	Rap	Eminem	Warner Inc.	15000

http://localhost:3000
0xF781I...5e7ID : BUY SONG
18.75 ETH

Estimated fee 0.0035951
0.0035951 ETH
Max fee: 0.0035951 ETH

Total 18.7535951
18.7535951 ETH
Amount + gas fee Max amount: 18.7535951 ETH

Figure 6.8: Buying Song


Blockus

Wedding Events Inc.
Commercial ID: 1
Library

Godzilla	Rap	Eminem & Kanye	Warner Inc.

Figure 6.9: Library Page for Commercial Users

Chapter 7

Testing

7.1 Test Cases

Black box testing is a software testing methodology that focuses on evaluating the functionality of a system without knowing its internal structure, design, or implementation details. In the context of automated tools for auditing smart contracts, black box testing can be used to assess the effectiveness and reliability of the tool without considering its underlying code or algorithms.

Table 7.1: Black box testing of the system

Test Name	Test Description	Test Step	Expected Result	Actual Result
User Registration Test	Verify that users can successfully register on the platform	Navigate to the registration page, Fill in the required user details, and submit.	User should be registered successfully and log in automatically	User was able to register and login successfully

Continued on next page

Test Name	Test Description	Test Step	Expected Result	Actual Result
Adding Song Test	Verify that Record Labels can add songs	The Record Label once logged-in clicks on ’Add Song’ and, enters details like song name, royalty payout, genere, artists, song cost	Song should be added successfully and visible in the published songs list and to users to purchase	Song was added successfully
Buying Song Test	Verify that Users can buy songs	The user (commercial or personal) once logged-in navigates to store and clicks on cart icon to purchase song	Song should be added successfully and visible in the users library	Song was bought successfully

Continued on next page

Test Name	Test Description	Test Step	Expected Result	Actual Result
Compatibility Testing	Verify that the application is compatible with different browsers and operating systems	Access the application using different browsers and operating systems	Application should function correctly across all tested platforms	Application functioned correctly across all tested platforms

Chapter 8
Conclusions

8.1 Conclusion

In conclusion, the Blockchain-Based DRM Management System is a significant step forward in addressing the complex challenges of the modern music industry. By harnessing the power of blockchain technology, this system offers transparency, security, and efficiency for managing digital rights, royalties, and music licensing. Through careful design and analysis, we've developed a comprehensive solution that caters to a wide range of functionalities and stakeholders. At its core, this system aims to empower artists, record labels, users, and commercial entities by providing a secure and fair platform for music licensing and distribution. The use of blockchain technology ensures transparency and reliability in the licensing process, backed by smart contracts that automate royalty payments. This guarantees that everyone involved in the music ecosystem receives just compensation for their contributions. The provided user interface designs give a glimpse of the system's user-friendly and intuitive experience. Artists and record labels can effortlessly manage their catalog, define contract terms, and keep an eye on earnings. Users (listeners) can purchase music with lifetime ownership, while commercial entities can obtain music licences. The personalized dashboards offer quick access to relevant information for all user roles. As the digital music industry evolves, the demand for a transparent, efficient, and secure platform for DRM, licensing, and royalty management grows. This project represents a significant stride towards creating a more equitable and prosperous music ecosystem.

8.2 Future Scope

The future of blockchain-based DRM management systems looks incredibly promising, with several exciting opportunities on the horizon. One noteworthy aspect is the potential for more efficient and transparent royalty payments. Blockchain technology has the power to automate payments to artists and copyright holders, ensuring accuracy and timeliness while also cutting down on administrative costs.

Our research has identified the shortcomings that arise in transparency, efficiency, and accessibility of copyright information. We proposed a decentralized application (dApp) built on the Ethereum blockchain to address these issues. The dApp streamlines royalty payments, increases transparency through public transaction records, and establishes a public registry for copyright information.

However, our project also revealed some of the limitations of blockchain technology, such as scalability, processing times, and storage costs. Additionally, the legal implications of copyright enforcement through smart contracts remain unclear. While our dApp functioned well in a local environment, further testing is required to assess its efficacy in a real-world setting, with real record labels and artists, and their respective licensing agreements on the Ethereum Mainnet.

Future work should focus on these limitations and areas for improvement. Legal research is required to determine the validity of smart contract enforcement of copyright ownership. Additionally, a mechanism for user verification and copyright ownership needs to be developed to ensure the integrity of the system in order to verify if the uploader of the music is indeed the correct authorized user. Finally, ongoing code optimization and security audits are crucial for a robust and secure dApp.

References

[1] A. Garba, A. D. Dwivedi, M. Kamal and S. U. Khan, “A digital rights management system based on a scalable blockchain,” Peer-to-Peer Networking and Applications, vol. 14, no. 5, pp. 2665-2680, 2021.

[2] Ciriello, R. F. Ciriello, A. C. G. Torbensen, M. R. P. Hansen and J. Hedman, “Blockchain-based digital rights management systems: Design principles for the music industry,” Electronic Markets, vol. 33, no. 5, pp. 1-17, 2023.

[3] J. Shen, “Blockchain Technology and Its Applications in Digital Content Copyright Protection,” in Proceedings of the 4th International Conference on Economic Management and Green Development, C. Yuan, X. Li and J. Kent, Eds., Singapore: Springer, 2021, pp. 25-35.

[4] Maria Alice Bosseljon Roche. Innovating in the music industry: Blockchain, Streaming Revenue Capture. Diss., 2020.

[5] A. Qureshi and D. Megías Jiménez, “Blockchain-Based Multimedia Content Protection: Review and Open Challenges,” Applied Sciences, vol. 11, no. 1, p. 1, 2021.

[6] R. S. Ramani, S. V. Sri Vishva, L. Dua, A. Abrol and M. Karuppiah, “Blockchain for digital rights management,” in Hybrid Computational Intelligence for Pattern Analysis, S. K. Hafizul Islam, A. K. Pal, D. Samanta and S. Bhattacharyya, Eds. Academic Press, 2022, pp. 177-205.

[7] Adjei-Mensah, Isaac, et al. ”Securing music sharing platforms: A Blockchain-Based Approach.” arXiv preprint arXiv:2110.05949 , 2021.

Publications

Vendrell Mendonca, Ashwin Pillai, Elbin Thomas Abraham, Kyle Crasto and Annies Minu, ”Decentralized Music Marketplace: A Blockchain-Based Rights Management Approach”, International Journal For Multidisciplinary Research (IJFMR), 2024

Acknowledgements

We are thankful to a number of individuals who have contributed towards our final year project and without their help; it would not have been possible. First of all, we would like to express our gratitude to Ms. Annies Minu, our project guide, for their prompt suggestions, guidance and encouragement over the course of our entire project.

We sincerely appreciate Ms. Ankita Karia and Mr. Shamsuddin Khan, our project coordinator, for their tremendous assistance to our project. We also appreciate the support of the faculty in our department.

We express our sincere gratitude to our respected Director Bro. Shantilal Kujur, our Principal Dr. Sincy George and our Head of Department (CMPN) Dr. Kavita Sonawane for providing the facilities, encouragement as well as a conducive environment for learning.

Sincerely,

Vendrell

Vendrell Mendonca

[illegible]

Ashwin Pillai

[illegible]

Elbin Thomas Abraham

[illegible]

Kyle Crasto