COMP4801: Final Year Project

Blockchain: Music Licensing

Final Report

Supervisor: Dr. Allen M. H. Au

Date of Submission: 18th April 2022

Submitted By: Bevan Varghese

Group 21006

ANAND, Mahima (3035550987)

BAGRI, Siddhant (3035551785)

VARGHESE, Bevan (3035552777)

Abstract

The music industry's growth in the last decade is largely attributed to the growth of streaming services during the period. However, lack of consistency in the mechanisms between different streaming platforms and the presence of several intermediaries between the artist and the consumer have led to two major problems, namely copyright infringement and unfair distribution of artists' royalties. The goal of this project is to build a blockchain-based web application that tackles these two issues. In order to do so, the application is characterized by three main features: a decentralized file storage system, a decentralized rights ledger, and integrated cryptocurrency wallets. These features are implemented through a combination of industry-standard technologies and frameworks on the server-side and the client-side. Background research on similar projects in the domain was conducted. The main takeaway is that there are gaps in the technical designs of these applications, which prevent them from addressing both of the aforementioned issues simultaneously. Using this information, the system's components and the relationships between each component were finalized. Solidity-powered smart contracts serve as the core component of the system. The application offers all the functionality that can be expected of a typical music-streaming service. Ultimately, the vision of this application is to protect the rights and royalties of artists in the music industry.

Acknowledgement

I would like to extend my gratitude to all the individuals without whom this project would not be possible. Firstly, I would like to thank my supervisor Dr. Allen Au, whose constant support and guidance has ensured that the project stays on the right track. Next, I would like to thank Dr. John Yuen, whose valuable feedback served as pointers for the project going forward. I would also like to thank my Technical English instructor Ms. Grace Chan, who provided sound advice on matters related to report-writing, presentations, and documentation. Without her help, this report would be lacking in terms of cohesion, completeness, and professionalism. Additionally, I would like to thank my friends and family, whose unrelenting support has helped me persevere towards delivering the project with high effort. Finally, I extend my gratitude to the University of Hong Kong, who have provided my group mates and I with the opportunity to apply the knowledge we have harnessed over the course of our degree in a practical setting.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AES	Advanced Encryption Standard
API	Application Programming Interface
B2C	Business-to-Consumer
CBC	Cipher Block Chaining
COALA IP	Coalition Of Automated Legal Applications — Intellectual Property
EVM	Ethereum Virtual Machine
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IPFS	InterPlanetary File Storage
UI	User Interface

1. Introduction

This section introduces the current state of affairs with regard to music distribution and licensing, and proceeds to introduce the aims and goals of this project – to build a blockchain-based music streaming and licensing platform.

1.1 Background

Technological advances in recent decades have led to the widespread availability of music in digital formats. The transition of the music industry from traditional brick-and-mortar stores to digitized distribution has boosted the growth of the industry. As reported by the World Bank in 2020, more than US$350 billion was collected in charges for the use of intellectual property, with music royalties dominating these numbers [1]. Streaming services are the frontrunners in the music industry, due to the combined effect of the convenience of streaming and the accessibility of smart devices [2]. The growth of the industry and streaming services is seen in Figure 1, which depicts the industry's revenues globally since the turn of the century.

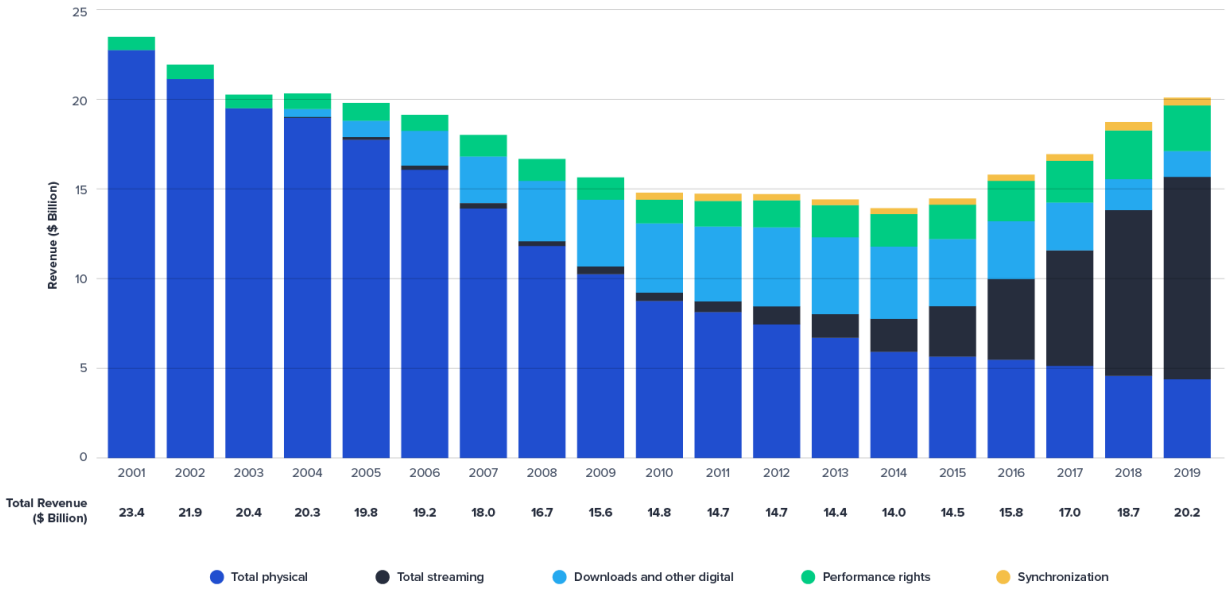

Figure 1: Revenues for the global music recording industry from 2001 to 2019 [2]

As is evident from Figure 1, the music industry's revenue growth in recent years coincides with that of streaming services. These services operate in tandem with artists through middlemen like music publishers and record labels. While publishers are in charge of musical composition copyright (the music and lyricism), record labels manage sound recording copyright (what the consumer finally hears). Additionally, several streaming platforms such as Spotify, Apple Music and Google Music adopt their own revenue models and file storage mechanisms. However, inconsistencies between the models and the presence of several middlemen have led to two main problems with music streaming: copyright infringement and the distribution of artists' due royalties [3].

1.2 Problem Statement

Copyright infringement refers to when consumers purchase licensed material and distribute it to other consumers for free or for their own personal gain. Since the music is protected by copyright law in the artist's name, public access to this material without due credit to the artist is a major problem in the Internet era.

The other issue relates to the protection of the artist's income. Some streaming services such as Spotify allow users to listen to their entire catalog of music on a subscription-basis, while others like iTunes only allow users to purchase singles/albums before they can listen to them. Due to these inconsistencies, unfair distribution of royalties among artists proves to be an issue. Moreover, the presence of the aforementioned middlemen like publishers and record labels further complicates the division of disbursements. The result of such a convoluted system is that artists, who make the music in the first place, are left feeling underpaid [4].

1.3 Literature Review

The problems mentioned above have generated keen interest in the research and start-up industries. This report discusses three such projects.

Ujo offers an Ethereum platform to eliminate confusion of music ownership and automate payments [5]. Easy license access is allowed through the COALA IP specification. IPFS is used to store the data, in order to maintain reliability and

decentralization. The caveat with Ujo, however, is the lack of a proper post-purchase copyright protection mechanism. Consequently, there are ways for users to illegally redistribute the music.

SingularDTV is a content distribution system on the Ethereum network [6]. Smart contracts power the platform, as they are used to democratize royalty collections and management of intellectual property. The platform's artists are given the power to create their own cryptocurrency tokens. By doing so, they can incentivize certain actions among their fanbase. As an artist becomes more popular, their tokens become more valuable and fans can also benefit from supporting newer talents. Much like Ujo, copyright protection is not a priority for SingularDTV. Moreover, artists cannot necessarily be expected to be aware of the economics involved in managing your own tokens.

Vezt is an alternate business model and acts as a rights marketplace [7]. Fans purchase partial ownership of the royalty rights and thereby directly fund the artists. Songs are essentially perceived as mini-corporations. Royalty rights may be purchased in a similar fashion to shares in a stock market. This not only incentivizes fans to support the artist's growth but also allows artists to generate large funding. Similar to SingularDTV however, this business model would be very difficult for artists to manage, without sufficient knowledge of financial markets. Additionally, artists lose some amount of autonomy when sharing the royalty rights with investors.

1.4 Objectives & Scope of Deliverables

The goal of this project is to build a blockchain-based web application that addresses the two problems of copyright infringement and unfair royalty distribution, as mentioned in Section 1.2. The platform has three main features:

(i) Decentralized file storage: This allows artists to upload their musical compositions and recordings, which can be accessed by anyone on the platform for a fee.

(ii) Decentralized rights ledger: This ensures validation of rights and licensing such that a track can be played by the user only after the transaction is confirmed.

(iii) Integrated cryptocurrency wallets: This enforces payments from the users for purchases and direct payments to the artists for royalties.

Blockchain's three fundamental characteristics make it ideal for the protection of copyright and licensing information. The three characteristics are as follows:

1. **Decentralization**: There is no single governing authority or person presiding over the platform. Instead, it is controlled by a distributed network of members.
2. **Immutability**: Data stored on the blockchain ledger cannot be altered or changed.
3. **Transparency**: Anyone can join the blockchain network and view the information stored.

Given the competition in the music industry, smaller artists struggle to make their careers financially viable. Figure 2 illustrates the dependency of these artists on alternate sources of income due to their limited earnings from music.

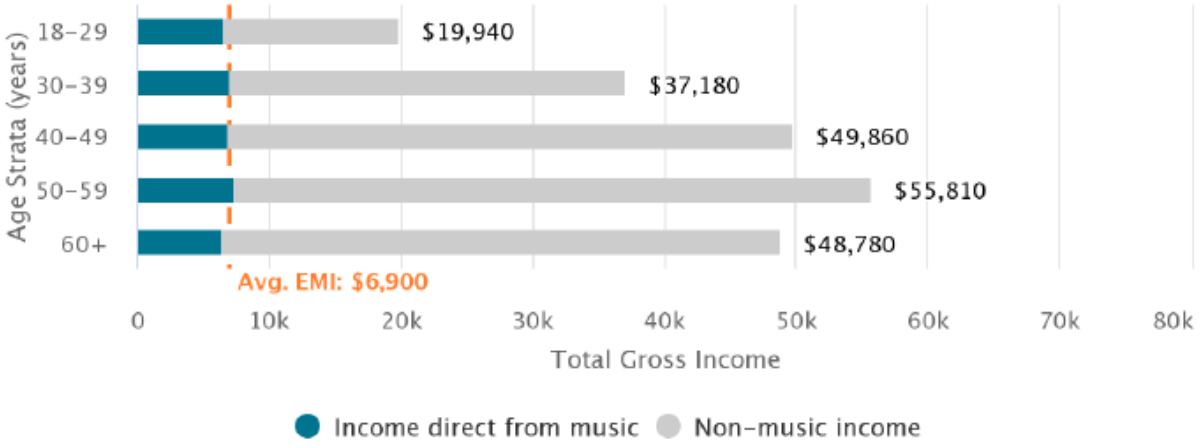

Figure 2: Gross income from music sources vs other sources across different age groups [8]

The elimination of costs associated with middlemen in our project's implementation is expected to lead to higher revenue-per-stream for all artists. Lost royalties as a consequence of unnoticed copyright infringement will also be minimized, owing to the decentralized setups for file storage and rights' information.

1.5 Report Organization

The remainder of this report proceeds as follows: first, to offer technical background and insight, Section 2 elaborates on the core technologies driving the platform. More specifically, blockchain, smart contracts, layer 1 and layer 2 implementations, and semi-decentralized applications are discussed. Section 3 elaborates on the methodology behind the implementation, and how different elements are incorporated to interact in a functional manner. Section 4 further discusses the features offered by the platform and the implementation details for each feature. Section 5 lays down the business model that the platform will operate on, so as to meet costs while delivering streaming services to the user. Section 6 discusses some of the challenges and limitations encountered over the course of development, as well as some of the solutions to address these difficulties. The project timeline is highlighted in Section 7. Further, Section 8 elaborates on some of the future enhancements of the platform and how they can be implemented. Finally, Section 9 recapitulates the project's goals and concludes with the future direction for applications in this domain.

2. Technical Background

This section outlines the key technologies and their suitable characteristics that will form the backbone of our project, namely - blockchain and smart contracts. It then describes why their application in solving the problem of music licensing presents a desirable solution.

2.1 Blockchain

A blockchain is defined as a digital public ledger for recording transactions [9]. This essentially serves as a database which stores information as a growing chain of 'blocks'. The suitability of a blockchain arises from its properties of being decentralized, transparent and immutable.

Through its decentralized network architecture, each member has access to an identical copy of the data stored on the blockchain ledger which updates in real time. New data is verified and added via a consensus algorithm and any compromise to a member's ledger would be rejected by majority members on the network [9]. This ensures data transparency and keeps the blockchain reliable. Moreover, without regulation from a centralized authority, peer to peer transactions can take place instantly eliminating the need to pay an intermediary fee.

Once created, it is impossible to alter the chain, rendering it permanent and immutable. This is achieved by cryptographic hashing. Each new block comprises of the following – (i) the recorded data, (ii) the hash value of the previous block linking it to the rest of the chain, and (ii) a new hash value generated from the contents of the current block and the previous block's hash which allows for linkage to the next block. Thus, any modification to a block would also alter its hash value and conflict with the existing block and would subsequently require every other block to be modified.

2.2 Smart Contracts

Smart contracts are programs stored on the blockchain that execute when predefined conditions, or "trigger-events", are met [10]. Hence, the execution of an agreement or another event can be automated without the effort or time of any intermediaries. Much like regular contracts, they define rules in the form of code which are automatically enforceable by specified function calls. Smart contracts are most commonly used for executing virtual currency transactions [10]. They possess the

benefit of being trackable since any interaction with a smart contract is irreversible. Each contract requires some gas fees to execute.

2.3 Layer 1 & Layer 2 Networks

The increasing popularity of cryptocurrencies has increased the need for blockchain layers for improved transaction rates, network security and recordkeeping. When the demand on the network is high, it gets clogged. This causes the pooling up of pending transactions. As a result, the processing and execution times are increased. In order to handle this, miners on the network begin to prioritize transactions with higher gas prices. Consequently, the minimum cost of executing a transaction increases to the point where the gas fees skyrockets unreasonably.

This is where layering alleviates the issue. Where layer 1 is the native underlying blockchain, layer 2 is a third-party integration that operates on this layer so as to improve its efficiency [11]. The dual-layered structure allows for an increased number of nodes, which further results in a higher transaction throughput [11]. The layer 2 blockchain offloads a portion of layer 1's transactional burden onto another system architecture for managing the processing load. The processed load is then reported to layer 1 for finalization. This reduces congestion and improves network scalability. Layer 2 solutions utilize smart contracts to make these transactions [11].

2.4 Semi-decentralization

The defining characteristic of blockchain-based applications is decentralization. However, as quoted by Deloitte Insights, "blockchain-based systems are comparatively slow" [12]. While several organizations are moving towards the adoption of blockchain technology, the sluggish transaction speed acts as a bottleneck for these systems. The time taken for creation of new blocks, processing gas payments, and verifying transactions is quite significant, especially when dealing with the user experience of an application. In the context of a music-streaming platform, performing operations such as searching and sorting, in addition to uploading, encrypting, and playing, could prove

to be very computationally expensive. A simple solution to combat this feasibility issue is a semi-decentralized architecture: the combination of a backend API with a blockchain solution, to extract the best features from both [13]. The integrity of information on the blockchain is preserved while the efficiency of performance is maintained with the help of the backend.

2.5 Inferences

The qualities of the aforementioned technologies can be leveraged to build a decentralized music-sharing platform. Using blockchain technology would ensure a timely, transparent, and traceable payment system. Moreover, decentralization would allow for the development of a uniform, universal database. Payments can be made via cryptocurrencies. Smart contracts can automatically distribute the revenue among owners in pre-decided proportions as per the copyright agreement. Additionally, the layer 2 scaling solutions prove beneficial to building a decentralized application (DApp) on the blockchain network. The hybridization of the application into a semi-decentralized application would allow all these features to be delivered while still maintaining a high standard of user experience.

3. Methodology

This section explains the technology and engineering choices for the system, their advantages and the expected difficulties from these choices. The architecture diagram in Section 3.1 represents the architecture of the system and an explanation of its components follows in later subsections.

3.1 The Deliverables

As mentioned in the introduction, the goal is to develop a decentralized music platform that handles file storage, royalty collections, and rights & licensing. A web-based platform powered by blockchain technology will be developed by the end of this project. More specifically, the platform combines three main features:

- Decentralized file storage to host the music files and serve them to listeners upon request. This is implemented using IPFS and Infura. All uploaded files are visible to every user on the network. However, their access is restricted to those who stream the songs.
- Decentralized rights ledger implemented via smart contracts to manage and validate copyright, licensing, and availability of media on the platform. A listener is only able to play a track upon a confirmed transaction with payment redirected to the wallets of copyright owners listed in the contract.
- Integrated Cryptocurrency Wallets using Metamask for automatic payments from listeners and to artists for quick, efficient transfers.

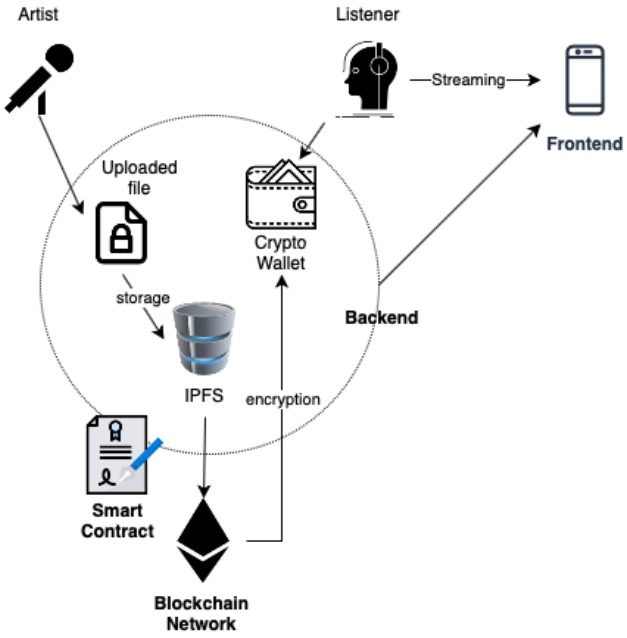

Figure 3: System architecture diagram

The above features and their implementations will also encompass a proper post-purchase copyright protection mechanism. This will ensure that the rights of a song are protected even after it is accessible to a user. The details of the same are discussed in Section 3.8. The application will be built through a combination of industry-standard backend and client-side frameworks, which will be discussed in later sub-sections.

3.2 The Underlying Blockchain Platform

When it comes to building blockchain-based applications, Ethereum and Hyperledger Fabric are the two most popular blockchain platforms [14]. While both platforms come with their own sets of advantages and disadvantages, Ethereum combined with Polygon as a secondary scaling solution was found to be most suitable for the development of this project.

Being an open-source platform backed by a highly active community, Ethereum offers the fundamental functionality that any blockchain has, from transaction initialization to transaction validation, and so on. Moreover, it is heavily tested and thoroughly documented, thus making the development process smoother. Ethereum supports public as well as private platforms, making it suitable for Business-to-Consumer (B2C) transactions. Given that the project's target audience is the general public, Ethereum is preferred to its counterpart Hyperledger Fabric, which only grants blockchain access to a set of predefined users [14].

A potential security concern in choosing Ethereum over Hyperledger Fabric is that the former posts all transactions to the public ledger, making it visible to all users. In contrast, Hyperledger Fabric provides transaction privacy, wherein visibility and accessibility can be specified by the developers [14]. However, it is worth noting that through the use of an appropriate encryption mechanism, it can be ensured that only by using both the public and the private keys can a user access their own transaction information on the blockchain.

The main piece of functionality that comes with Ethereum is the support for smart contracts. This makes the blockchain programmable since smart contracts run pieces of code upon certain trigger-events. Details and implementation of the same will be discussed in the next sub-section.

However, the Ethereum network by itself has some limitations as a blockchain development platform in terms of low throughput, risk of clogging and a

non-customizable technology stack. To deal with this, the team is using Polygon along with MATIC - its associated cryptocurrency. Polygon is a framework which serves the purpose of developing and connecting to networks that are Ethereum compatible [15]. It also provides the benefits of higher security alongside lower gas fees for processing transactions and faster speeds [15]. It has adaptor modules and a protocol to facilitate the exchange of messages with Ethereum [15].

3.3 Smart Contracts

Smart contracts serve as the project's centerpiece. To develop the smart contracts, the programming language Solidity is being used, because the language was specifically designed to write smart contracts, as well as to target the Ethereum Virtual Machine (EVM) [16].

When an artist uploads a single, it is captured on the blockchain in a smart contract. This smart contract contains all relevant information in terms of ownership, such as the artist's name, the song's title, the recording of the track, the rights to the track, information on the availability of the track, and the artist's crypto wallet address for payments [16]. The contract also serves the purpose of automating transactions. When listeners stream the single on the platform, a "play" event is triggered, and as a result, the artist gets paid.

For developing, prototyping, and testing the smart contracts, Truffle is used as the blockchain pipeline. Truffle is a development environment for blockchains using the EVM, which should facilitate the implementation of smart contracts as intended. MetaMask is currently being used for testing these contracts via the Polygon testnet. MetaMask is a cryptocurrency wallet software which allows users to interact with DApps on the Ethereum blockchain.

3.4 The Frontend

The client-side of the project presents the users with a GUI for interaction and engagement. Some functions such as the uploading of tracks interact with the blockchain through the backend API. Other features such as the music player and crypto wallets are presented on the frontend. ReactJS is used to develop the frontend, as it is the industry-standard framework used to design UI components. To help with styling, libraries such as MaterialUI and Bootstrap are used.

3.5 The Backend

The backend provides an API to support important additional functionality such as IPFS uploading, encryption and decryption, playlist support, etc. To design the API, Express is used as the web framework on top of Node.js. Following the use of JavaScript to build the server-side, Web3.js will be used to connect the Node.js server to the chain. Web3.js is essentially a collection of JavaScript libraries that facilitate interaction with a local or remote Ethereum node using HTTP requests [17]. The API endpoints offered are `/upload`, `/play`, `/fetch-all-playlists`, and `/add-song-to-playlist`. The functionality of each endpoint is discussed further in Section 4.

3.6 The Database

The backend is supplemented by MongoDB. MongoDB is a NoSQL database wherein data is organized in the form of documents within collections (similar to rows within tables). The database will be used to store collections of songs and playlists. The structure of songs and playlists can be seen in Figure 4.

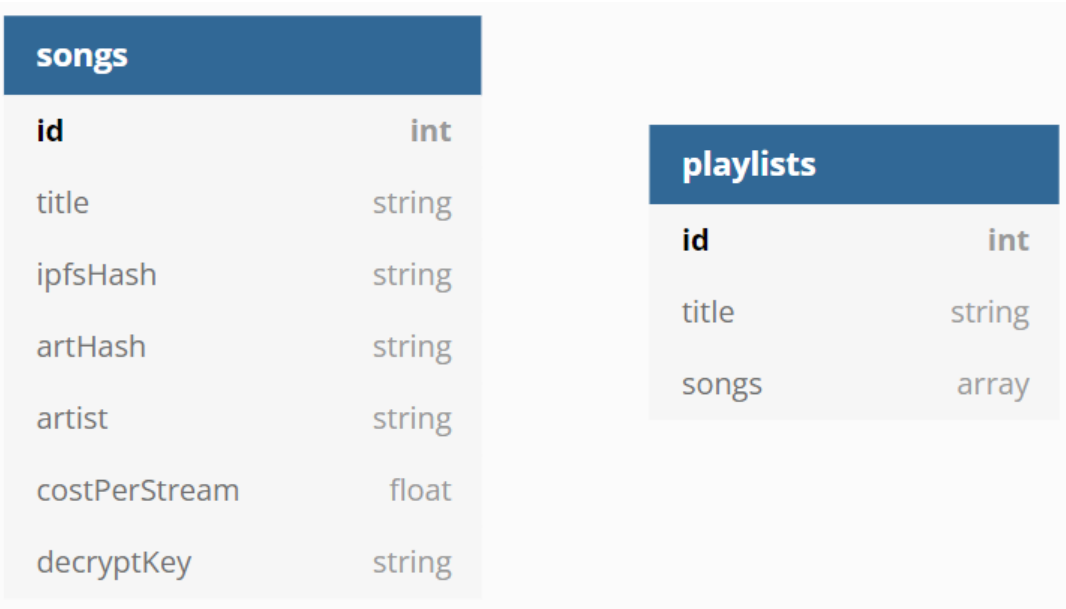

Figure 4: Data structures for songs and playlists

3.7 Peer-to-Peer Data Storage

From a development perspective, storing large amounts of data in the form of files on Ethereum blockchain would be considerably expensive. To manage this, the platform's chain is linked to the InterPlanetary File System (IPFS) - a distributed peer-to-peer storage system [18]. It serves the purpose of storing and accessing files via their cryptographic hashes stored on the blockchain.

However, if the IPFS node is down for some reason, the audio file may become temporarily unavailable. Infura is used to solve this problem. Infura is a Web3 infrastructure that makes access to the blockchain faster by connecting instantly to the Ethereum and IPFS networks [19]. Infura will pin uploaded IPFS files and keep them available to the network in a reliable manner. It will also accommodate scaling as the number of files grows.

3.8 Post-Purchase Copyright Protection

An important goal of the project is to protect the music copyright. Unlike the start-ups discussed in Section 1.3, the system implements a post-purchase copyright protection mechanism. There are different implementations for the network, storage, service and view layers.

Network Layer: Every peer is connected via the blockchain network. A user must be synchronized with all the information on the chain to avail of any services. This ensures that no rights are tampered with and all peers have the same version of the data.

Storage Layer: IPFS is the storage of choice. It is a peer-to-peer protocol which allows for reliable decentralized storage [18]. A file uploaded to IPFS can only be accessed by a unique hash, which is generated during file-upload. These hashes are saved on the smart contract, along with other details such as the song title and contributing artists. The usage of smart contracts ensures the immutability and traceability of data.

Service Layer: This layer manages offline ownership enforcement, by using watermarking on audio files. A watermark on an audio file refers to adding a short sound effect or voice-over into the audio file to claim ownership [20]. This implementation also allows the system to recognize illegal or inorganic uploads. The service layer will also provide an encryption function to protect the IPFS hash and audio file in case the IPFS hash is leaked. The specifics of the encryption feature are discussed in Section 3.9 and Section 4.3.

View Layer: Access to audio files is limited to the system itself, i.e, users can only use the frontend to pay for and play music. This ensures that no one gains access to the files outside the system to distribute them.

3.9 Encryption Algorithm

The audio file for each song will be encrypted by the backend, before being uploaded to IPFS. The encryption algorithm used is Advanced Encryption Standard

(AES). AES is one of the most popularly used block cipher mechanisms. By repeatedly executing the same encryption operations several times, the encrypted file is generated. The key length for AES can vary between one of 128, 192, and 256 bits. Moreover, there are six different modes in which AES can be used: authenticated encryption with additional data (AEAD), electronic codebook (ECB), cipher block chaining (CBC), cipher feedback (CFB), output feedback (OFB), and counter (CTR). For this project, AES-256-CBC is used, which indicates AES with a 256-bit key, using cipher block chaining.

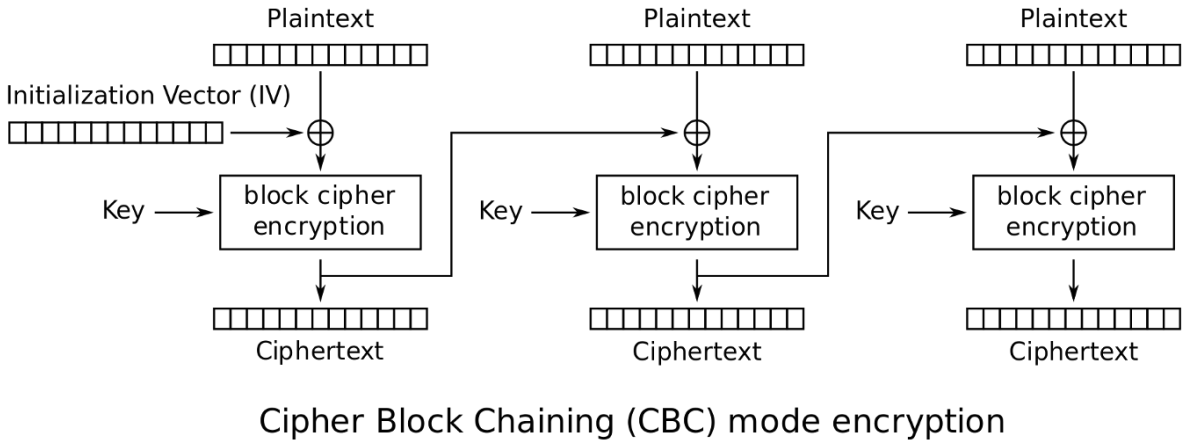

Figure 5: Cipher Block Chaining (CBC)

To perform CBC, a randomly generated number is required. This number is referred to as the initialization vector (IV). The input message is divided into blocks. An XOR operation is performed on each block with the defined IV. The name CBC is derived from the fact that the output of each encryption operation is fed in as the input for the next block. This encryption operation is performed a certain number of times to produce the final encrypted message. For decryption, the key, IV, and encrypted message are fed in as inputs. This operation is performed the same number of times as specified during encryption. If the key and IV used are the same as the original ones used for encryption, then the original message is retrieved and the file is decrypted.

3.10 Software Engineering Practices

The crux of software engineering in today's world is to follow an effective workflow to productively address complex problems and deliver products of the highest quality. Agile methodologies have been adopted by the team over the course of development, with the Scrum framework particularly serving as a guideline.

Jira is used as the project management tool of choice, wherein job functions are organized into sprints. Each sprint is set for a duration of two weeks. Every task, bug and user story is recorded in the Jira backlog and moved to the active sprint board in order of priority and precedence. Work is divided internally within the team as equitably as possible at bi-weekly meetings.

4. Feature Implementation

This section describes the project's main features. It goes over the implementation details and the approach used to effectively deliver the proposed features. The overall UI of the application is presented below in Figure 6.

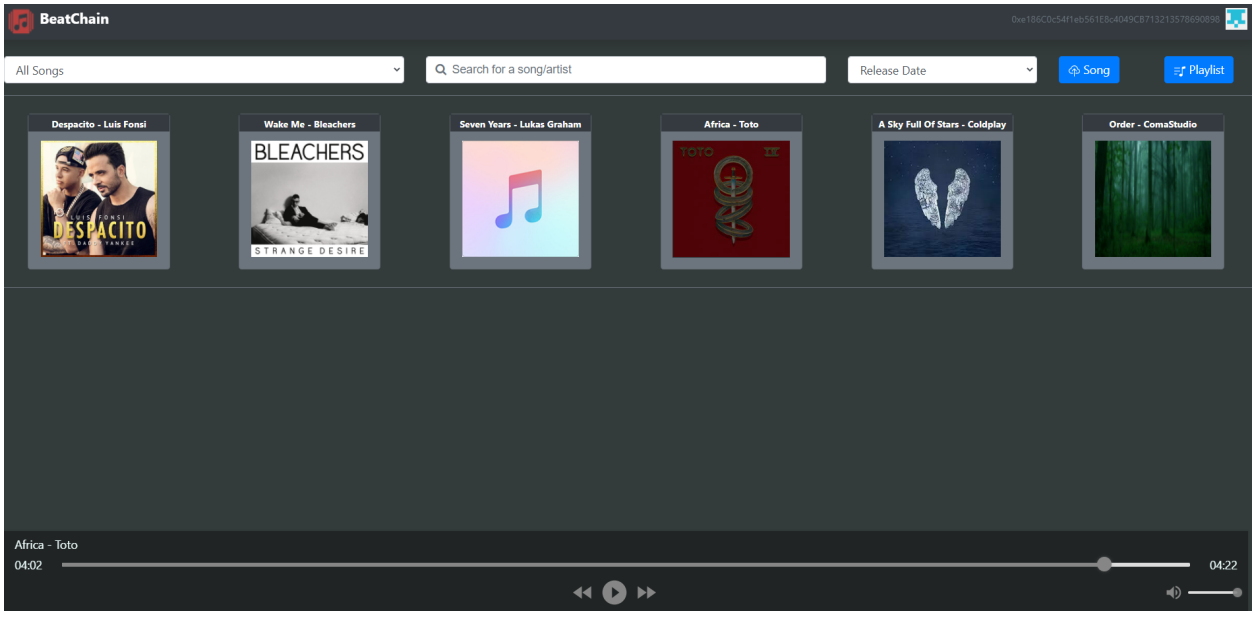

Figure 6: UI of the application

At the top of the page is the navigation bar, which presents the user's wallet address on the right side. Below that is the toolbar, where the user can choose the playlist using the dropdown menu, search for songs using the search-bar, sort the music according to recency/artist/title, upload new songs, and create playlists. Below that is the feed where all the music is rendered. Finally, the audio player is embedded at the bottom of the page.

4.1 Smart Contract

The platform is powered by a smart contract which is defined as follows:

1. **`struct Song:`** Indicates each song, with the following details
    a. **`uint id:`** Identifies the song.
    b. **`string hash:`** IPFS hash of the audio file.
    c. **`string title:`** Title of the song.
    d. **`string costPerStream:`** The cost for each stream of the song in MATIC.
    e. **`string artHash:`** IPFS hash of the song's artwork.
    f. **`string artist:`** Artist's name.
    g. **`address author:`** Wallet address of the uploader.
2. **`function uploadSong():`** A function to upload the song to the smart contract, using the details provided.
3. **`function stream():`** A function to perform the payment from the user to the artist (via the contract), whenever a user requests to stream a song. The payment is equivalent to the cost-per-stream specified during upload.
4. **`event SongUploaded:`** An event which is emitted when a song is added successfully.
5. **`event PaidForStream:`** An event which is emitted when a successful payment is made.

4.2 User Login

When the platform is launched, the user's browser connects to the chain through Web3.js. The user is asked to log into their Metamask account. Upon successful login, the user's wallet address is displayed in the navigation bar and he/she is now granted access to the platform. The screenshot below shows the initial metamask connection popup for login.

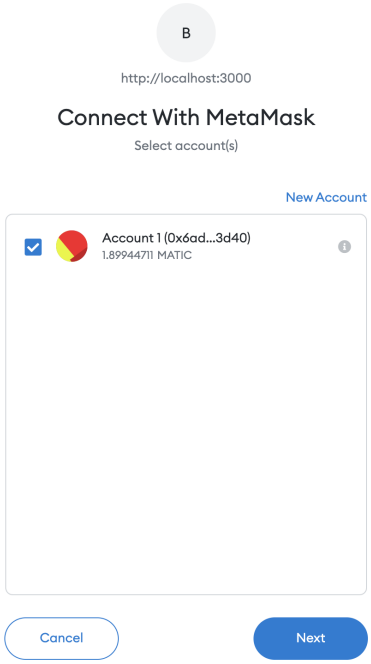

Figure 7: Metamask wallet connection request

4.3 Upload Music

Users are allowed to upload music to the platform. The files are stored using IPFS, as mentioned in earlier sections. The files are encrypted so as to prevent illegal redistribution and violation of copyright. The flow for uploading music is described as follows:

1. **User fills out the form:** The user presses the "New Song" button and a modal containing the form for the new song is shown, with fields for the audio file, the song's title, the artist's name, the cost-per-stream (in MATIC), and an optional field for the album artwork. The user then presses the "Submit" button.

2. **Frontend processes the data:** The audio file is captured and converted into a base64 string (`"data:audio/mpeg;base64,/+MYxAAEaAIEeUAQ…"`) which is further converted into a file-buffer object. Validation is performed on the remaining fields, such as limiting the cost-per-stream to 0.007 MATIC, in accordance with the business model proposed in Section 6.
3. **Frontend sends the data to the backend:** The frontend makes a POST request to the API endpoint `/upload`, including all the song's data in the request body. The frontend then waits for the backend's response.
4. **Backend encrypts the file:** The backend generates a unique key (and initialization vector) and encrypts the received file-buffer using the AES-256-CBC algorithm.
5. **Backend uploads the file to IPFS:** The backend then uploads the encrypted file to IPFS and saves the IPFS hash that is received as a response.
6. **Backend saves information to the database:** After receiving the IPFS hash of the file, the backend saves all the information about the song, including the unique encryption key, to the database.

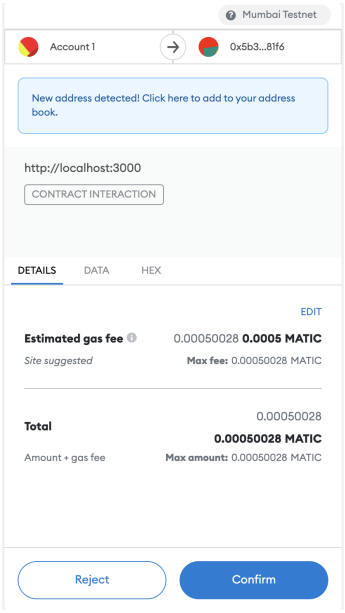

Figure 8: Metamask transaction confirmation request

7. **Backend returns IPFS hash to the frontend:** The backend now returns the IPFS hash of the file to the frontend.

8. **Frontend interacts with the contract:** The frontend now uploads the song's metadata (including the returned IPFS hash) through the contract's `uploadVideo` method. If the user's wallet has sufficient balance to pay off the gas fee, then the transaction goes through and the song is successfully uploaded. Metamask provides a confirmation on the success or failure of the transaction, as denoted in Figure 8.
9. **Frontend displays the new song:** Upon receiving the transaction hash from the contract, the frontend displays the uploaded song, which is now available for all users to play.

4.4 Play Music

Users are able to play music upon request from the music-feed provided. Upon launch, the feed brings the latest uploaded song into focus. Users are free to select any song from the uploaded set, so long as they have sufficient funds in their account. The flow for playing music is described as follows:

1. **User selects a song:** The user clicks on a song to play. The system identifies this song using the song ID provided by the contract.
2. **User makes payment:** A transaction dialog is triggered in Metamask, wherein the user confirms the payment. The amount is equal to the cost-per-stream specified by the uploader (in MATIC). This amount is paid to the artist via the smart contract.
3. **Frontend makes a request to the backend:** The frontend makes a POST request to the API endpoint `/play`, sending the song's ID and IPFS hash in the request body. The frontend awaits a response from the backend.
4. **Backend retrieves the song's encryption key:** Using the provided details, the backend fetches the song's document from the database and retrieves the encryption key.
5. **Backend fetches the audio file from IPFS:** The backend uses the hash to fetch the encrypted audio file (as a base64 string) from IPFS.

6. **Backend decrypts the audio file:** Using the encryption key, the backend decrypts the audio file and returns it to the frontend. Currently, the file is in a base64 string format.
7. **Frontend processes the response:** The frontend receives the decrypted audio file (as a base64 string). This string gets converted into a blob, which is a file-like object of immutable, raw data [21]. This file can be fed into the audio player.
8. **Frontend creates an access URL:** Since media files require an `src` attribute, the frontend uses the web API's `URL.createObjectURL()` method to create a unique URL for the blob. This URL is released when the window is closed or when the audio track changes.
9. **Audio player plays from URL:** Using the URL created, the audio player's `src` tag is updated. The song starts playing.

4.5 Searching

The user is allowed to search for a song through the search-bar provided. The functionality is implemented within the frontend itself, using **state management in React** [22]. After communicating with the chain during the initial load, the frontend (i.e., React) receives the updated list of songs. These songs are saved to the React component's "state", which is essentially a data structure that defines the UI. React uses the state's data to render the UI accordingly. Hence, the songs are saved to the state variable `songs`. To implement search functionality, a **state variable called `searchQuery`** is created. Whenever the value in the search-bar changes, the `searchQuery` updates accordingly. Hence, **only `songs` that match the `searchQuery`** in terms of title/artist are shown in the feed. Therefore, by using React state management, search functionality is enabled without needing back-and-forth communication with the backend or the chain.

4.6 Sorting

The user is allowed to change the sorting order of songs through the dropdown menu provided. Similar to search functionality, sorting is implemented using **React state**

management as well. In this case, the **`sortingFilter` state variable** is set to one of the available fields: “recent”, “title”, or “artist”. Based on the field that is specified, the **`songs` are sorted and displayed** accordingly. The specified sorting is preserved even when the `searchQuery` changes. Therefore, React state management facilitates sorting functionality without a helping hand from the backend or the chain.

4.7 Playlists

Users are allowed to create playlists, add songs to playlists, and select a playlist to listen to. The workflows for each function are described as follows:

(i) Creating a playlist: The user presses the “New Playlist” button, which triggers the modal to open. Here, the user enters the name of the playlist to be created. Upon confirmation, the frontend makes a POST request to the API endpoint `/create-playlist` along with the playlist title. The backend creates a new playlist in the database with the aforementioned title, playlist-ID, and an empty array of song-ID’s. This array will be used to denote the songs that are in the playlist. The backend then responds with the playlist-ID, following which the frontend displays the playlist.

(ii) Selecting a playlist: When the application is loaded initially, a GET request is made to the API endpoint `/fetch-all-playlists`. The backend retrieves all playlists from the database and returns it to the frontend. The array of playlists is saved to the React state. Using the dropdown menu, the user can select a playlist they would like to listen to. Based on the information about each playlist saved in the state, the frontend displays only those songs which are in the selected playlist. This is done by checking if each song’s ID is in the playlist’s array of song-ID’s.

(iii) Adding songs to a playlist: When a user selects a playlist, an “Add Song” button is visible. When this button is clicked, a table of songs not in the current playlist is shown. When the user selects a song to add to the playlist, the frontend makes a POST request to the API endpoint `/add-song-to-playlist`, along with the IDs of the new song and the corresponding playlist. Using this information, the backend finds the playlist’s document in the database and pushes the song-ID to the array of songs. The

backend then responds to the frontend with a successful message, and the frontend displays the added song in the playlist.

4.8 Album Art

Expanding upon the steps described in Section 4.3 for uploading music, an optional field for album artwork is presented to the user. If the user uploads an artwork image for the track, the frontend forwards the image buffer to the API endpoint `/upload`, along with all the other data. The backend uploads the image to a separate IPFS URL, and returns the artwork's hash to the frontend. The frontend uploads this hash to the contract while uploading all the other details of the song. When rendering each song, the image uses the artwork's IPFS hash as the `src` and displays the artwork accordingly.

4.9 Mobile Support

Considering the platform is a web-application, it supports mobile devices as well. The users will have to install the Metamask application from their respective app store and log in to their account. They can then open the platform on a browser and stream music on their mobile device.

5. Challenges & Difficulties

This section describes the main challenge faced by the platform as well as solutions to issues that were discussed in the interim report.

5.1 Piracy Concerns

A core objective of the project is to prevent copyright infringement, which is the most prominent issue faced by the media industry. The platform is designed in such a way that it would be difficult to pirate any uploaded content, given the encryption and costs associated with each song. The loophole with online media however, is that data needs to be stored on the user's machine ultimately. This is an industry-wide issue, faced by even the media streaming giants Netflix, Amazon Prime, and YouTube [23]. In

the case of this platform, the decrypted audio file is temporarily stored on the user's browser so that the HTML audio tag can access the file through the `src` attribute. Since the internet is currently designed in such a way that media needs to be locally stored in order to be accessed, it is difficult, but not impossible, to pirate any content posted on the platform.

5.2 Encryption Mechanism

The goal of the encryption mechanism is to ensure that the audio file is protected even if the IPFS hash is leaked. At the interim stage, the team proposed testing two approaches to solve the issue. The first approach involved encrypting the IPFS hashes in the backend such that users cannot see the decrypted hashes in the response. When a user plays a song, the backend will facilitate the IPFS call and only return an encrypted hash whose key is stored secretly. The second method involved encrypting the audio files directly. In this case, even if a user gets the IPFS hash, they can only access the encrypted audio file. The decryption key will be stored in the backend and it will be used to decrypt the audio file for users who have paid for the stream. Encryption of the file itself not only served as a more secure method but also did not show any significant decrease in server response time. Hence, after testing both approaches, the team settled on the latter approach.

5.3 Searching and Sorting

In terms of user experience and engagement, the platform should allow users to search for any song they wish to listen to, instead of scrolling through the music-feed. Similarly, users would like to sort the music feed in an order they desire, be it in terms of title, artists, or recency. Since all the data so far is stored either on IPFS or in the smart contract, performing searching or sorting directly on the chain would prove to be expensive. In order to tackle this problem, the plan was to develop a search-function either within the frontend or the backend. After performing further development, the team settled on using React state management on the frontend to facilitate searching and sorting.

6. Business Model

In this section, the business model adopted for the project is discussed. An analysis of popular models in the industry is conducted, following which a market comparison is done to finalize the numbers.

6.1 Analysis of Different Models

6.1.1 Pay-Per-Stream Model

As the name suggests, the idea behind a pay-per-stream model is that users pay the platform a certain fee for each stream that they request [24].

Advantages: Such a model is both affordable and flexible for users. There is no obligation for the users to commit to the platform and use the services. Moreover, the platform owners can redistribute revenue only to the artists whose music has been streamed. Additionally, advertisements will not be required, since users pay as they use the platform. Advertisements can be very disruptive while listening to music. User experience is preserved in this manner.

Disadvantages: It is difficult for the platform to retain customers with such a model. Firstly, customers may find it easier to use a platform where they are not expected to make a payment for every stream. Secondly, since there is no obligation or additional incentive to keep using the platform, the users may find it easier to depart. Furthermore, pay-per-stream models can fluctuate a lot in terms of revenue, depending on the service usage.

6.1.2 Subscription Model

A subscription model is one wherein users pay a periodic fee (every month, for instance) as a subscription fee to the platform [25].

Advantages: Given that the users make a commitment to the platform when they pay the fee, there is a greater incentive for them to use the platform and get their money's worth. The familiarity they gain with the platform over time makes them more likely to stay as a subscriber, as opposed to leaving for a different service. The

predictability of revenue each month allows the platform shareholders to make business decisions with lesser risks.

Disadvantages: Subscription models come with higher upfront costs and longer commitments than their counterparts. For example, one-year plans tend to work out cheaper than one-month plans for most platforms. Since users make a long-term commitment, a feeling of guilt arises when they do not use the platform. Another issue with this model is the unfair distribution of royalties to artists. Artists receive a specific percentage of the revenue, even if they were not responsible for that percentage of streams during the time period [25].

6.1.3 The Verdict

For the subscription model, the benefits are more suited to the platform and not the artist. For the pay-per-stream model however, benefits are catered to both users as well as artists. The model also captures the essence of blockchain technology, considering it eliminates the need for centralization and gives freedom to peers. Hence, the pay-per-stream model is adopted for this project. As for commission, the platform will take 15% of all the payments made for streaming.

6.2 Market Comparison

With the pay-per-stream model chosen for this project, a market comparison was conducted with two giants of the music-streaming industry: Apple Music and Spotify.

Apple Music: Apple Music adopts a subscription model. On average, Apple Music users pay around HK$0.06 per stream globally (varies by country) [26]. Publishers take around 50% of the cut [27], while record labels can take anywhere between 10-35%, depending on the type of deal [28]. Artists end up receiving about between 15-40% approximately, which caps their revenue-per-stream at around HK$0.009 to HK$0.024 per stream.

Spotify: Spotify also adopts a subscription model. Spotify does not release their data on royalty distributions directly, but inferences can be made from some of their reports. [29] Some artists on the platform get paid, even if their songs are not streamed.

The average user pays around HK$0.029 per stream. Similar to Apple Music, publishers receive 50% of the cut [27] while record labels take upto 35% [28]. This leaves artists with around 15-40% of the revenue. Hence, their revenue-per-stream totals at about HK$0.009 to HK$0.024 per stream.

6.3 Cost Analysis

- The average cost-per-stream on our platform is 0.0183225 MATIC, which equates to around HK$0.02. Given the platform's rate of commission is 20%, the artists get around HK$0.016 on average for each stream. The revenue-per-stream falls in a similar bracket to the industry giants, but the artist gets around 80% of the cut here, owing to the reduced streaming costs.
- At this cost, the user pays approximately HK$20/month, assuming they listen to 1,000 streams each month. In comparison, Spotify and Apple Music users pay upwards of HK$58/month.
- For HK$58/month, a user can expect to listen to approximately 2,900 streams. With the average song running for four minutes in duration, that equates to around 11,600 minutes of streaming. In comparison, the average Spotify monthly usage is 3,540 minutes while the average Apple Music monthly usage is 3,960 minutes.
- To match the average monthly costs of Spotify and Apple Music, our platform can offer streams as high as HK$0.058. From this, artists would get HK$0.0464. This is nearly four times the revenue rate offered by Apple Music (average of HK$0.0165) and six times the revenue rate offered by Spotify (average of HK$0.007975).

Table 1 given below compares the numbers that have been mentioned across the three platforms. The feasibility of the proposed business model can be verified from the table.

	Spotify	Apple Music	BeatChain
Average Pay	29	60	20
Publisher Commission	14.5	30	0
Average Record Label Commission	6.525	13.5	0
Total Average Commission	21.025	43.5	4
Average Artist Income	7.975	16.5	16

Table 1: Revenues and commissions per 1000 streams on different streaming platforms (in HK$)

7. Schedule & Milestones

Table 2 below outlines the current status of development and the timeline of deliverables for the project:

Time Period	Tasks Planned/Deliverables	Status
August 2021 - September 2021	Project topic deliberation and background research	Complete
October 2021 - November 2021	➔ Phase 1 Deliverable: ➔ Project proposal report ➔ Project webpage ➔ Enhancing our domain knowledge and development capabilities regarding blockchain and smart contract development over the Ethereum Platform ➔ Finalizing the system design of our application	Complete

December 2021 - January 2022	➔ Commencing application development ➔ Phase 2 Deliverable: ➔ Interim Report ➔ First Project Presentation	Complete
February 2022 - March 2022	➔ Completion of application development ➔ Testing and debugging ➔ Code review, documentation and refinement	Complete
April 2022	➔ Phase 3 Deliverable: ➔ Final Project Presentation ➔ Final Report	Complete

Table 2: Project timeline

8. Future Enhancements

This section discusses potential enhancements and features that could be added to the platform during the next phase of development.

1. **Featured artists:** Currently, the platform only supports one artist per song. A common theme in the music industry is for multiple artists to perform on the same track. Such artists are known as featured artists. The contract's song-structure can be changed so that each song can have multiple artists, each identified by their wallet address. By allowing artists to describe the revenue-split during the upload process, artists can be paid their due percentage of the royalties automatically.
2. **Album playthroughs:** A feature that users would like is to listen to an album sequentially from the first track to the last track. Users can make playlists (to denote albums) and manually play each song of the playlist to resemble the experience. A future enhancement would be to allow users to pay the overall cost

of all the songs in the album with a single click, thereby allowing them to listen to the whole album seamlessly and without interruptions.

3. **Privacy toggle for playlists:** All playlists that are created on the platform are visible to all other users. By simply adding a boolean variable to indicate whether a playlist should be made private or public, the platform can allow users to make playlists for their own, personalized listening.
4. **Ratings and comments:** A simple feedback mechanism for each song would go a long way in polishing the user experience. Users would be able to identify music that they would like based on the feedback provided by their peers, in terms of ratings and comments. Since users are required to pay for a song before listening to it, they would appreciate knowing how other listeners perceive the song.
5. **Mobile and desktop applications:** The platform is currently developed as a web-application. While it is supported on both mobile and desktop platforms, a key development would be to build standalone applications for both platforms. By catering the UI to the user's device specifically, the platform would go a long way in terms of user experience.
6. **Song recommendations:** Similar to other music streaming platforms, a recommendation system could be built. The system should be able to recommend songs to each user based on their own preferences (content-based filtering), as well as the preferences of other users who enjoy similar genres of music (collaborative-based filtering).
7. **Blockchain development:** With the release of Ethereum 2.0 expected in around three to six months, the platform may see significant improvements in terms of efficiency and responsiveness [30]. Gas fees could be decreased as well, allowing both the platform to face lower costs as well as artists to pay lower fees while uploading music.

9. Conclusion

There are two main problems that plague the music industry today, namely copyright infringement and the unfair distribution of artists' royalties. This project explores the application of blockchain technology to solve these two issues. The aim of the project is to develop a blockchain-based web-application that is characterized by a decentralized rights ledger, decentralized file storage, and smart contracts that ensure fair distribution of royalties. The platform currently offers all the basic functionalities of a typical music player, but does so while staying true to its objectives. Artists are allowed to upload music and get paid for it, while listeners are allowed to pay for music they would like to listen to as well as make playlists.

The ultimate vision of this application is to protect the rights and dues of artists in today's music industry. The media-streaming industry still has its fair share of issues when it comes to online distribution and piracy. Nonetheless, the application of blockchain technology is a step in the right direction for the protection of copyright. In view of the expected release of Ethereum 2.0 within the next year, blockchain technology can and should be applied to solve more problems, both within the music industry and otherwise.

References

[1] “Charges for the use of intellectual property, receipts (BOP, current US$),” *The World Bank DataBank*. [Online]. Available: https://data.worldbank.org/indicator/BX.GSR.ROYL.CD?end=2020&most_recent_value_desc=true&start=1980&view=chart.

[2] J. Stone, “The state of the music industry in 2020,” Toptal Finance Blog, October 06, 2020. [Online]. Available: https://www.toptal.com/finance/market-research-analysts/state-of-music-industry.

[3] S. Zhao and D. O’Mahony. “BMCProtector: A blockchain and smart contract based application for music copyright protection,” Trinity College Dublin, Ireland, 2018. [Online]. Available: https://www.researchgate.net/publication/330891236_BMCProtector_A_Blockchain_and_Smart_Contract_Based_Application_for_Music_Copyright_Protection.

[4] B. Sisario. “Musicians say streaming doesn’t pay. Can the industry change?” The New York Times, May 07, 2021. [Online]. Available: https://www.nytimes.com/2021/05/07/arts/music/streaming-music-payments.html

[5] S. de la Rouviere, “Introducing ujo portal: Making musicians more money.,” *Medium*, 13-Dec-2018. [Online]. Available: https://blog.ujomusic.com/introducing-ujo-portal-making-musicians-more-money-9224d808a57a.

[6] SingularDTV, “Introducing the Breaker Royalty Management Platform,” *Medium*, 24-Jun-2021. [Online]. Available: https://singulardtv.medium.com/introducing-the-breaker-royalty-management-platform-60a819b80c1e

[7] “Music fans share ownership with artists in their favorite songs,” Vezt. [Online]. Available: https://vezt.co/

[8] “Money from music survey data portal,” *Future of Music Coalition's Artists Revenue Streams Project*. [Online]. Available: http://arsdata.futureofmusic.org/dashboard/show?utf8=%E2%9C%93&role_composer=true&role_recording=true&role_salaried=true&role_performer=true&role_session=true&role_teacher=true&mgenre=ALL&ft=false&trained=false&careerexp=ALL&gender_male=true&gender_female=true&gender_transgender=true&gender_unanswered=true&emigroup=1. .

[9] M. Nofer, P. Gomber, O. Hinz, and D. Schiereck, “Blockchain,” *Bus. inf. syst. eng.*, vol. 59, no. 3, pp. 183–187, 2017.

[10] S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han, and F.-Y. Wang, “Blockchain-enabled smart contracts: Architecture, applications, and future trends,” *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 49, no. 11, pp. 2266–2277, 2019

[11] Bybit Learn, “Blockchain layer 1 vs. layer 2: Things you must know,” *Bybit Learn*, 08-Oct-2021. [Online]. Available: https://learn.bybit.com/blockchain/blockchain-layer-1-vs-layer-2/.

[12] L. Kenny. “The blockchain scalability problem and the race for visa-like transaction speed,” *Medium,* 31-Jan-2019. [Online]. Available: https://towardsdatascience.com/the-blockchain-scalability-problem-the-race-for-visa-like-transaction-speed-5cce48f9d44

[13] CSIRO. “Semi decentralised applications (semi-dapp),” *Blockchain Patterns.* 2021. [Online]. Available: https://research.csiro.au/blockchainpatterns/general-patterns/deployment-patterns/semidapp/

[14] H. Anwar. “Hyperledger vs Ethereum,” 101 Blockchains, April 04 2019. [Online]. Available: https://101blockchains.com/hyperledger-vs-ethereum-2/

[15] “Ethereum's internet of blockchains,” *Polygon*, 15-Sep-2021. [Online]. Available: https://polygon.technology/

[16] “Solidity - Solidity 0.8.9 documentation,” Solidity, September 29 2021. [Online]. Available: https://docs.soliditylang.org/en/v0.8.9/

[17] “Blockchain music rights in (about) 3 minutes”, YouTube, Feb 16 2018. [Video file]. Available: https://www.youtube.com/watch?v=1LUKgbWihGU&ab_channel=BruceBalensiefer

[18] “IPFS powers the distributed web,” *IPFS Powers the Distributed Web*. [Online]. Available: https://ipfs.io/

[19] “Ethereum API: Ipfs API & gateway: ETH Nodes as a service,” *Infura*. [Online]. Available: https://infura.io/

[20] I. Team, “How to watermark audio files - all options explored,” *How to Watermark Audio Files - All Options Explored*. [Online]. Available: https://www.intrasonics.com/news/2020-10-30-how-to-watermark-audio-files-whats-your-best-option

[21] “Blob”, *Mozilla Developer Network*. 20-Feb-2022. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Blob

[22] V. Gupta. “React state management: What is it and why to use it?”, *Login Radius*. [Online]. Available: https://www.loginradius.com/blog/engineering/react-state-management/#:~:text=What%20is%20React%20State%20Management,for%20a%20JavaScript%20data%20structure.

[23] A. Tiwari. “This is how your favourite Netflix movies and shows are pirated,” *Fossbytes*, 10-July-2019. [Online]. Available: https://fossbytes.com/how-pirate-netflix-amazon-prime-movies-shows-piracy/

[24] “Subscription vs pay-per-use - which revenue model would work for your business?”, *Subscription Flow.* 2020. [Online]. Available: https://www.subscriptionflow.com/2020/07/subscription-vs-pay-per-use-which-revenue-model-would-work-for-your-business/

[25] A. Paul. “Introducing pay-per-use: Consumption-based subscription for cohesity service provider partners”, *Cohesity*, 16-Jun-2020. [Online]. Available: https://www.cohesity.com/blogs/introducing-pay-per-use-consumption-based-subscription-for-cohesity-service-provider-partners/

[26] D. Curry. “Apple Music revenue and usage statistics (2022)”, *Business of Apps*, 14-Apr-2022. [Online]. Available: https://www.businessofapps.com/data/apple-music-statistics/

[27] T. Stein. “Music publishing explained: How artists get paid for their songs”, *Careers in Music*, 11-Jan-2022. [Online]. Available: https://www.careersinmusic.com/music-publishing

[28] D. Pastukhov. “A hard look at how record companies make money: royalty splits, types of record deals and the label business model”, *Soundcharts*, 10-Feb-2020. [Online]. Available: https://soundcharts.com/blog/splits-and-profits-record-deals-analysis

[29] M. Iqbal. “Spotify revenue and usage statistics (2022)”, *Business of Apps,* 19-Jan-2022. [Online]. Available: https://www.businessofapps.com/data/spotify-statistics/

[30] D. McQuaid. “ETH 2.0: What’s happened so far and when is the next phase?”, *Currency*, 05-Apr-2022. [Online]. Available: https://currency.com/eth-2-0-what-s-happened-so-far-and-when-is-the-next-phase