



**COMP4801: Final Year Project**  
**Blockchain: Music Licensing**  
**Final Report**

**Supervisor:** Dr. Allen M. H. Au

**Date of Submission:** April 18, 2022

**FYP 21006**

**Author:** BAGRI, Siddhant (3035551785)

**Group Members:** ANAND, Mahima (3035550987) & VARGHESE, Bevan (3035552777)

## Abstract

The evolution of the music industry from physical stores selling records to a predominantly digital economy has facilitated an extensive global industry. A majority of the market share is held by streaming services which connect to artists via music publishers and record labels. Such a setup not only reduces the revenue of artists due to the commissions involved, but also fails to identify cases where royalties are not paid at all resulting in potentially thousands of unnoticed copyright infringements. This project aims to decentralize the setup of the music industry with the use of blockchain to securely and transparently manage file storage, rights distribution and royalty collections. The team has therefore developed [BeatChain](#), a streaming platform on the Polygon Matic network which uses a smart contract to automate rights validation and royalty payments to artists. Other features are implemented in a semi-decentralized architecture using industry-standard technologies. Given more time, the team has also found room for improvement in functionality and UX. Furthermore, an average case product feasibility analysis based on historical data reveals that BeatChain can generate higher revenues for artists at lower costs for listeners compared to industry leaders Spotify and Apple Music. Further improvement in blockchain protocols and lower gas fees will make such a product even more rewarding.

## **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 our supervisor Dr. Allen Au, whose constant support and guidance ensured that the project stayed on the right track. Next, we 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 Mr. Cezar Cazan, who provided sound advice on matters related to report-writing, presentations, and documentation. Without his help, this report would be lacking in terms of cohesion, completeness, and professionalism. I extend my gratitude to the University of Hong Kong for providing me with the opportunity to apply the knowledge I have harnessed over the course of our degree in a practical setting. Last but not least, my heartfelt gratitude to my group mates, Bevan and Mahima for their effort, support and camaraderie.

## Table of Contents

Abstract	i
Acknowledgement	ii
Table of Contents	iii
List of Figures	v
List of Tables	v
List of Abbreviations	vi
<b>1. Introduction</b>	<b>1</b>
1.1 Background	1
1.2 Problem Statement	2
1.3 Literature Review	3
1.4 Objectives & Deliverables	4
1.5 Report Organization	5
<b>2. Technical Background</b>	<b>6</b>
2.1. Blockchain	6
2.2 Layer 1 vs Layer 2 Networks	7
2.3 Smart Contracts	7
2.4 Cryptocurrency Wallets	8
2.5 Advanced Encryption Standard (AES)	8
<b>3. Methodology</b>	<b>10</b>
3.1 The Deliverables	10
3.2 The Underlying Blockchain Platform	11
3.3 Smart Contracts	12
3.4 The Frontend	13
3.5 Peer-to-Peer Data Storage	13
3.6 Centralized Database & Server	14
3.7 Post-Purchase Copyright Protection	15
<b>4. Final Product - BeatChain</b>	
4.1 Features	16
4.1.1 Login	16
4.1.2 Upload Music	17
4.1.3 Play Music	19
4.1.4 Album Artwork	19
4.1.5 Searching and Sorting	20
	iii

4.1.6 Playlists	20
4.2 Challenges & Constraints	21
4.2.1 Encryption & Protection against Piracy	21
4.2.2 Semi-Decentralized Implementation	22
4.3 Future Vision	22
4.4 Project Timeline	23
<b>5. Business Model</b>	<b>25</b>
5.1 Pay Per Stream vs Subscription	25
5.2 Cost Analysis & Market Comparison	26
<b>6. Conclusion</b>	<b>29</b>
<b>References</b>	<b>30</b>

## List of Figures

Figure 1	Revenues for the global music recording industry from 2001 to 2019	01
Figure 2	Gross income from music sources vs other sources across different age groups	03
Figure 3	AES Encryption in CBC Mode	09
Figure 4	System Architecture Diagram	10
Figure 5	Semi Decentralized Architecture Interaction	14
Figure 6	BeatChain Home Page	16
Figure 7	Metamask Wallet Connection Request	17
Figure 8	IPFS Console Responses	18
Figure 9	Metamask Transaction Confirmation Request	18
Figure 10	Ideation & Development Timeline	24

## List of Tables

Table 1	Revenues and commissions per 1000 streams on different streaming platforms	27
---------	--	----

## List of Abbreviations

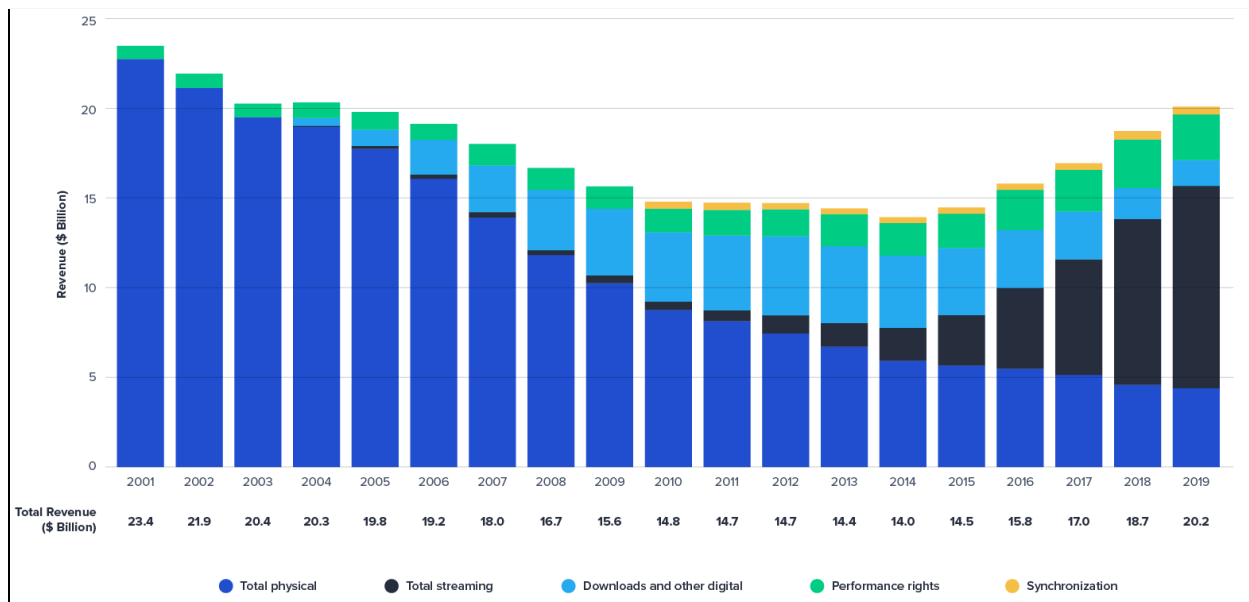
API	Application Programming Interface
B2C	Business-to-Consumer
COALA IP	Coalition Of Automated Legal Applications – Intellectual Property
EVM	Ethereum Virtual Machine
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IPFS	InterPlanetary File Storage
UI	User Interface
UX	User Experience
IV	Initialization Vector
XOR	Exclusive OR

# 1. Introduction

This section introduces the current state of the music industry in terms of music distribution and licensing. It then describes the aim 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 stores selling discs and records 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 portable and smart devices [2]. Figure 1 below describes the industry’s revenues globally since the turn of the century. The streaming revenues (in dark blue) have become the largest part of the market share over the last few years and this number is projected to increase further.



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

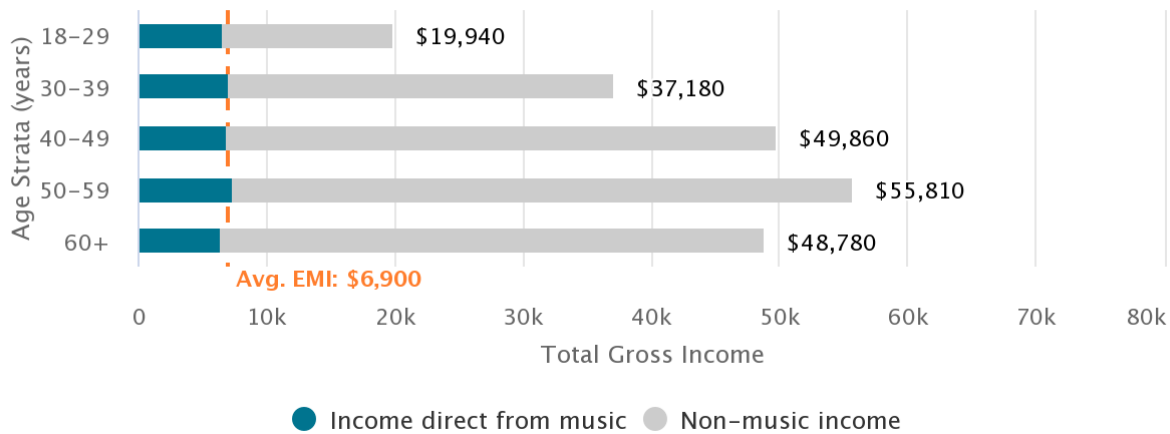


It is also important to understand the structure of the music industry from a streaming perspective. The music industry comprises artists, listeners, publishers and record labels. Artists write and create songs which the listeners then access through streaming services. The licensing and distribution of songs from artists to listeners is managed by publishers and record labels. A song consists of two parts: the musical composition which includes the music and lyrics created by the artist, and a recorded performance of the song as a single or album. Publishers are responsible for the musical composition copyright and help artists claim ownership of the song and manage allowed usages. Record labels manage the recorded performance, enforce ownership and help collect licensing fees for artists. When a streaming service wants to provide a song on their platform, they pay the royalties to the record label and can then add it to their database.

## **1.2 Problem Statement**

The presence of middlemen in the form of publishers and record labels have led to two main problems with music streaming: unfair distribution of artists' due royalties and poor copyright protection [3].

The distribution effort in the industry is expensive and therefore reduces revenue for artists who pay a considerable portion of their income to these middlemen [4]. Smaller or new artists find it extremely difficult to have a feasible career in the music industry. Streaming services also add to this problem because they hold a majority of the market share and can dictate how much they pay artists. In fact, in a survey conducted among over 5000 US-based musicians with a low to medium income stream, it was observed that non-music income was their primary source of income as shown in Fig. 2 below [5]. This was observed across age groups and demonstrates the need for a cost-effective solution for these artists.



*Figure 2: Gross income from music vs other sources across different age groups [5]*

The copyright protection that the current setup offers is also inefficient. Copyright infringement refers to the use of licensed material and for personal monetary 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 with rampant piracy. It is very difficult to monitor when a song is played without paying due royalties. This results in thousands of unnoticed copyright infringements which further reduces artists’ revenues.

### 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 [6]. It allows easy license access using the COALA IP specification. To provide reliability and decentralized storage, IPFS is used to store data. However, Ujo lacks a proper post-purchase copyright protection mechanism. Users can potentially distribute purchased music illegally.

SingularDTV is a content distribution system on the Ethereum network [7]. It uses smart contracts to democratize royalty collections and management of intellectual property. In fact, artists on this platform can create their own token and incentivize certain kinds of actions from fans. As an artist becomes more popular, their tokens become more

valuable and their fans can also benefit from supporting newer talents. Much like Ujo, copyright protection is not a priority for this project and it therefore does not address the problem properly. Moreover, it is difficult for artists to understand the economics of token management in such a model.

Vezt is an alternate business model and acts as a rights marketplace [8]. Artists are funded by fans directly who purchase partial ownership of the royalty rights. Essentially, songs are like mini-corporations and royalty rights can be purchased like shares in a stock market. This not only incentivizes fans to support the artist's growth but also allows artists to generate large funding. This model, however, is impossible for artists to manage without knowledge of financial markets. Moreover, in sharing the royalty rights with investors, artists will lose their autonomy in decision making and require voting for key decisions.

#### 1.4 Objectives & Deliverables

As discussed in Section 1.2, smaller artists struggle to make their careers financially viable. The objective of the project is to make the music industry a fair playing field for all artists and for artists to be recognized for the merits of their work. The project also improves upon the current copyright protection mechanism. The elimination of costs associated with middlemen in our implementation leads to upto 400% more revenue-per-stream for artists. Lost royalties as a consequence of unnoticed copyright infringement are also minimized using a smart contract streaming and rights protection. Files uploaded to the system are protected by encryption and therefore cannot be leaked from the storage node.

The deliverable of this project is **BeatChain** - a blockchain-based web application for music streaming. The platform has three main components:

- (i) Decentralized file storage: Stores the music which artists upload and users will stream 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.

### **1.5 Report Organization**

The report is structured as follows. Section 2 elaborates on the core technologies driving the platform to provide a foundation of the technologies involved to the reader. Section 3 elaborates on the methodology behind the implementation, and the need for the different elements. Section 4 discusses the final product, giving a clearer picture of what the system looks like, the features and the constraints. Section 5 analyzes the feasibility of the project as a business with a cost analysis in comparison to current streaming platforms. Finally, Section 6 summarizes the report.

## 2. Technical Background

This section outlines the key technologies that form the backbone of the project to provide the reader with base knowledge for future sections. It explains blockchain and the relevant layers that exist in the Ethereum network. It then describes smart contracts and the use of cryptocurrency wallets. Finally, it briefly explains AES encryption which is used to encrypt audio files in our system.

### 2.1. Blockchain

A blockchain can be defined as a digital public ledger for recording transactions [9]. It essentially serves as a database which stores information as a growing chain of 'blocks'. 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.
3. **Transparency:** Anyone can join the blockchain network and view the information stored.

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. 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 of 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 Layer 1 vs Layer 2 Networks**

The increasing popularity of cryptocurrencies in everyday life has necessitated the creation of blockchain layers for improved transaction rates, network security and recordkeeping. When the demand on the network is high, it gets clogged leading pending transactions to pool up thereby taking more time to process and execute them. 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 skyrocket unreasonably. Ethereum mainnet gas fees can be as high as HK\$120 per transaction, making it infeasible for a platform which requires many transactions in a short period of time [28].

Using a layer 2 network 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 [10]. The two layer structure allows for an increased number of nodes which results in a higher transaction throughput [10]. The layer 2 blockchain offloads a portion of layer 1's transactional burden onto another system architecture which manages the processing load. The processed load is then reported to Layer 1 for finalization. This reduces congestion and makes the blockchain network more scalable. The layer 2 solutions utilize smart contracts to make these transactions [10].

## **2.3 Smart Contracts**

Smart contracts are programs stored on the blockchain that execute when predefined conditions, or "trigger-events", are met [11]. 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

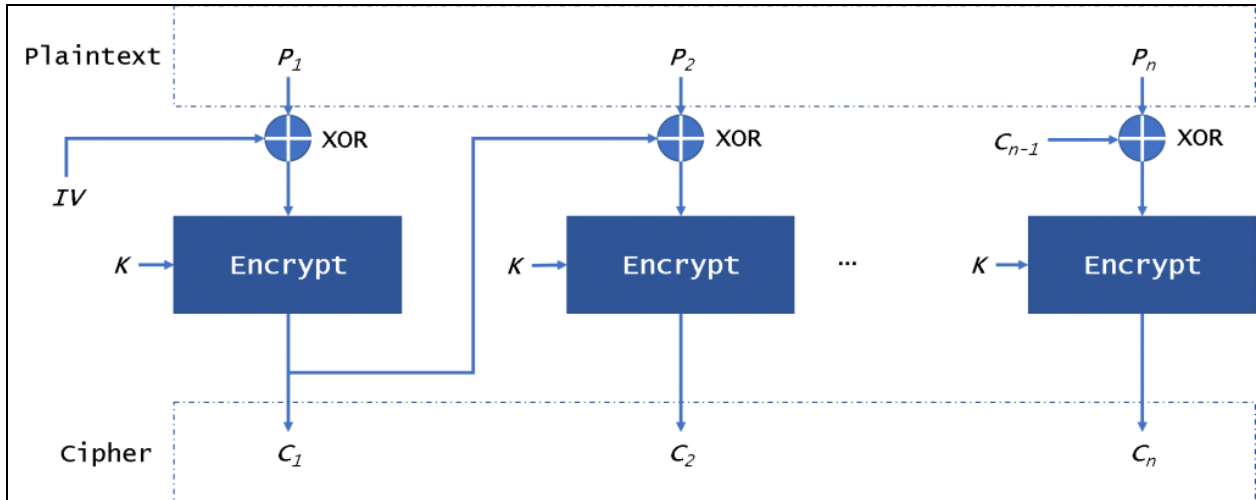
enforceable by specified function calls. Smart contracts are most commonly used for executing virtual currency transactions [11]. They possess the benefit of being trackable since any interaction with a smart contract is irreversible. Each contract requires some gas (fees) to execute.

## **2.4 Cryptocurrency Wallets**

A cryptocurrency wallet is a digital storage of decentralized tokens. It uses public and private keys to facilitate transactions and keep the tokens secure. When using a website or mobile application on a blockchain network, the application can be connected to the wallet and transactions can be made. An example of a widely used wallet on the Ethereum network is Metamask. It allows users to transact on the mainnet as well as testnets. It works as a browser extension which injects the Ethereum web3 API into a website's Javascript context [12].

## **2.5 Advanced Encryption Standard (AES)**

It is a widely used block cipher technique which performs the same encryption operation multiple times to finally generate an encrypted message. The key length can be 128, 192 or 256 bits and there are 5 different modes in which AES can be used [13]. The various modes differ in the inputs that they require and are slightly different in the way the encryption is performed. For example, AES-256-CBC means AES with a 256 bit key in Cipher Block Chaining mode. CBC requires a randomly generated number called the Initialization Vector (IV). As shown in Figure 3 below, the input message is divided into blocks and XORed with the IV. The output of one encryption operation becomes the input of the next block, hence the name CBC. This operation shown below repeats 14 times for a 256 bit key to finally produce the encrypted message.



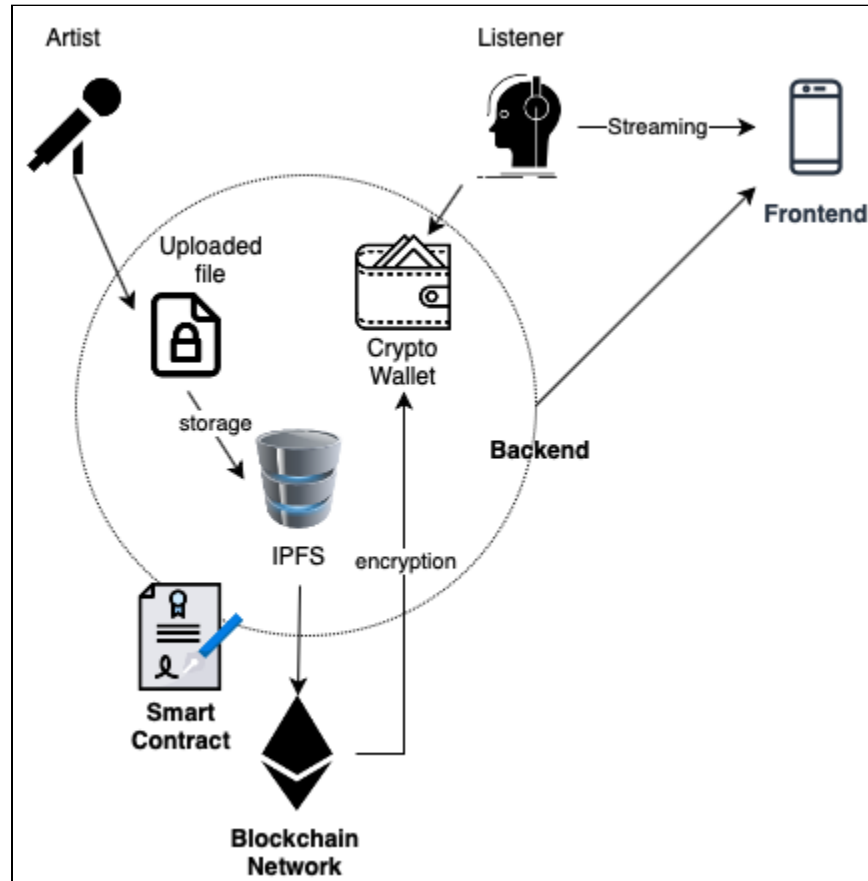
*Figure 3: AES Encryption in CBC Mode [13]*

The decryption operation also runs for 14 rounds with the key and IV as inputs (for CBC mode). If the key and IV are the same as the one used to encrypt the message, the original message is obtained. Thus, the key and IV used to encrypt a file need to be securely stored.



### 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 below in Figure 4 represents the architecture of the system and an explanation of its components follows in the subsections.



*Figure 4: System Architecture Diagram*

#### 3.1 The Deliverables

As mentioned in Section 1.4, the system is a decentralized music platform that handles file storage, royalty collections, and copyright licensing. The major components of the blockchain are:

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

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 3.7. 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 at the moment [14]. Ethereum combined with Polygon Matic as a layer 2 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 useful 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 Section 3.3.

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 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 developing on the Polygon Matic network. 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 are the most essential component of BeatChain. Solidity is the programming language of choice because it was specifically designed to write smart contracts as well as to target the Ethereum Virtual Machine (EVM) [16].

When an artist uploads a song, it is captured on the blockchain as 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 album artwork, and the artist's crypto wallet address for payments [16]. Another smart contract serves the

purpose of automating transactions. When listeners buy or stream the single on the platform, a “play-or-purchase” event is triggered, and as a result, the artist gets paid.

For developing, prototyping, and testing the smart contracts, Truffle is being 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 used as a cryptocurrency wallet.

### **3.4 The Frontend**

The frontend runs on the browser and allows users to upload, stream, search, sort and create playlists. Some functions such as playlists interact with the backend API, while others such as the uploading of tracks interact with the blockchain through the backend API. Some other features such as the music player and crypto wallets are also presented on the frontend. ReactJS is used to develop the frontend, as it is the industry-standard framework used to design UI components. Bootstrap is used for default components and easy styling.

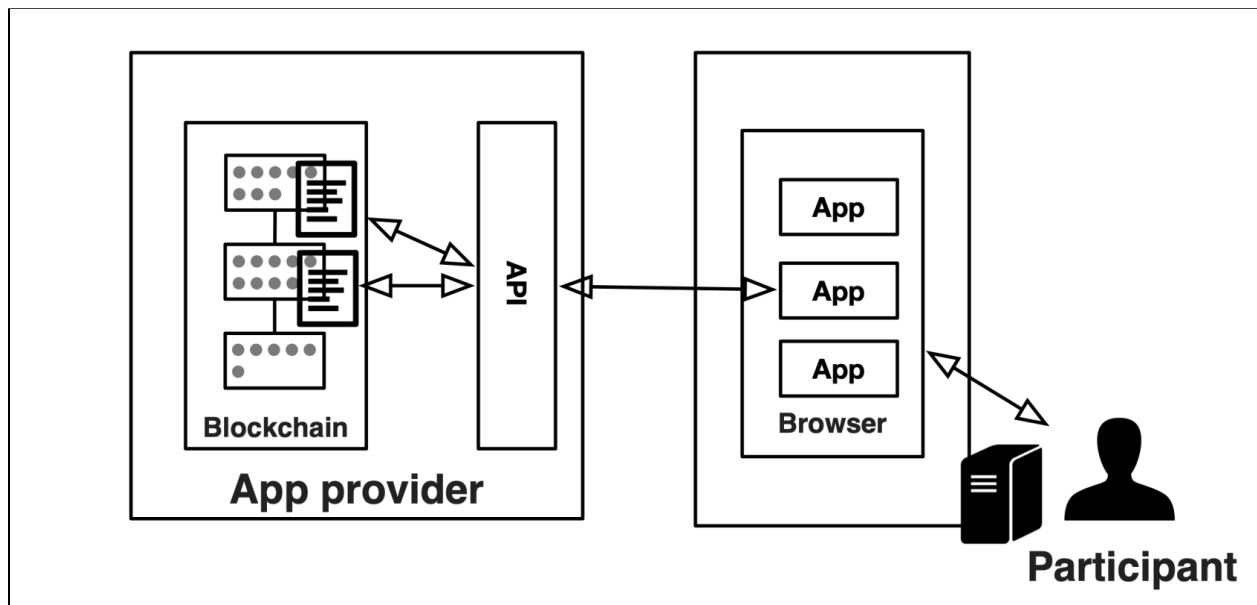
### **3.5 Peer-to-Peer Data Storage**

From a development perspective, storing large amounts of data in the form of files on Ethereum blockchain is going to grow considerably expensive. To manage this, the chain is linked to the InterPlanetary File System (IPFS) which is a distributed peer to peer storage system [17]. 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 service that makes access to the blockchain faster by connecting instantly to the Ethereum and IPFS networks [18]. 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.

### 3.6 Centralized Database & Server

It is a common practice to have semi-decentralized architectures in blockchain applications [19]. Such an architecture involves the use of smart contracts and blockchain for the transactions and storage of files in the decentralized setup while the centralized database provides for more functionality and UX with greater efficiency. Figure 5 below shows the semi-decentralized architecture and the interactions between the system components.



*Figure 5: Semi Decentralized Architecture Interaction [19]*

In BeatChain, the API currently supports encryption, playlists and additional UX features. 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 is used to connect the Node.js server to the chain. Web3.js is a collection of JavaScript libraries that facilitate interaction with a local or remote Ethereum node using HTTP requests [20]. In the future, however, with more streaming features and UX improvements, the backend server will make extensive use of the centralized database. It is important to note here that all the important transaction and copyright protection related work is on the chain with the centralized database playing a supporting role.

### 3.7 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 which protects the audio file from leaking. 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.

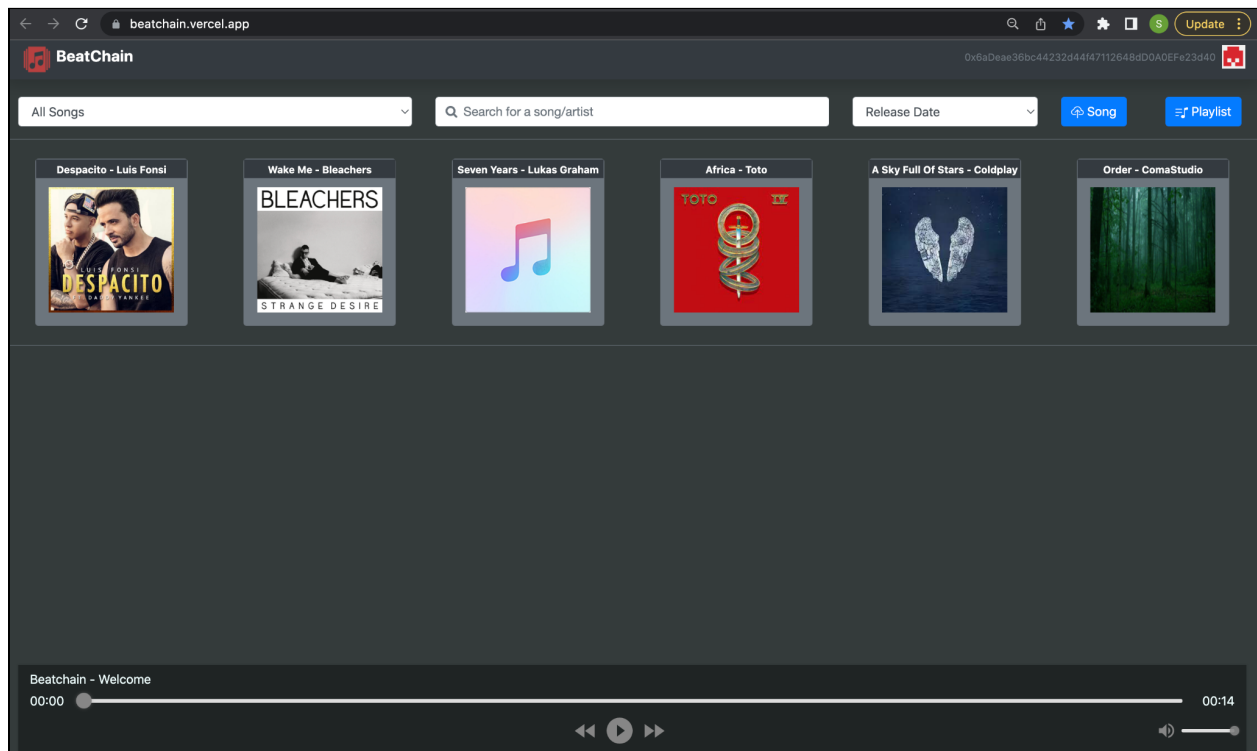
**Storage Layer** - IPFS is the storage of choice. It is a peer-to-peer protocol which allows for reliable decentralized storage [16]. A file uploaded to IPFS can only be accessed by a unique hash which is generated on file upload. These hashes are saved on smart contracts along with the song title and copyright owner. The usage of smart contracts ensures the immutability and traceability of data. Moreover, to protect the file in case of IPFS hash leaks, the file on IPFS is encrypted and the encryption key is stored in the centralized database.

**Service Layer** - On stream request, the encrypted file is retrieved from IPFS, decrypted on the backend server and sent to the frontend audio player. This need to store the decrypted file on the user's system creates an industry-wide security vulnerability which is discussed in detail in Section 4.2.1.

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

## 4. Final Product - BeatChain

This section describes the system's implementation, giving a more detailed description of its functionality, usage and constraints. It also discusses the difficulties encountered in the development process, the solutions to these problems and a future vision for the system. The system has been deployed and can be accessed at <https://beatchain.vercel.app/>. Figure 6 below shows the homepage with all the features described in detail in Section 4.1.

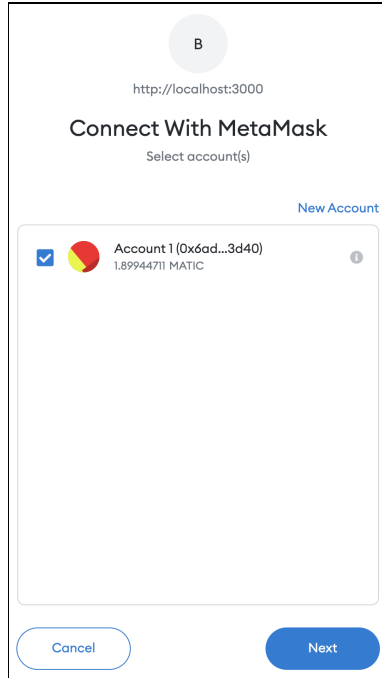


*Figure 6: BeatChain Home Page*

### 4.1 Features

#### 4.1.1 Login

Once the user launches the website, the user's browser connects to the chain through Web3.js. The user is asked to log into Metamask. Using this information, the user's wallet address is displayed in the navigation bar and he/she is now granted access to the platform. Figure 7 below shows a screenshot of the initial metamask connection popup for login.



*Figure 7: Metamask wallet connection request*

#### 4.1.2 Upload Music

The system allows artists to upload music which is then instantly available for streaming. The overall flow and implementation is as follows:

1. User inputs audio file, title, cost per stream and album artwork
2. The audio file is converted to a base64 string and then a buffer for IPFS storage
3. The buffer object is sent to the backend via a post request
4. The backend encrypts the buffer with AES-256 in CBC mode
5. The encrypted buffer is uploaded to IPFS by the backend
6. IPFS returns the hash of the uploaded file which can be used later to retrieve it
7. The backend saves the encryption key, IV and IPFS hash on the centralized database
8. The backend then returns the IPFS hash to the frontend
9. The frontend saves the song to the smart contract with this hash and other inputs from step 1

Figure 8 shows the console logs by IPFS, including the hash on successful upload.



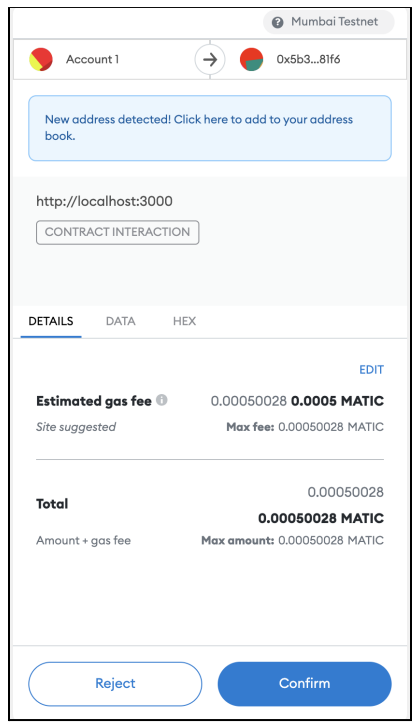
```
buffer App.js:73
  Uint8Array(2924042) [255, 251, 208, 68, 0, 9, 133, 36, 12
7, 60, 65, 230, 66, 240, 165, 47, 183, 133, 60, 200, 126,
21, 65, 250, 252, 44, 24, 192, 2, 130, 63, 32, 36, 244, 1
52, 185, 0, 0, 128, 32, 10, 197, 246, 118, 53, 123, 248,
▶ 108, 106, 53, 98, 177, 88, 129, 233, 152, 135, 136, 136,
49, 64, 224, 50, 100, 33, 140, 143, 100, 232, 154, 118, 7
6, 154, 123, 236, 130, 16, 64, 129, 4, 11, 79, 17, 18, 18
0, 74, 147, 247, 30, 143, 122, 36, 188, 10, 24, 133, 138,
86, 238, 247, ...]

Submitting file to IPFS... App.js:79

IPFS result App.js:82
▼ [{...}] ⓘ
  ▶ 0: {path: 'Qma3iR3gqsk6SUedH9y9q7m91h8Csmff6vsYv3PB3ViV...
length: 1
  ▶ [[Prototype]]: Array(0)
```

*Figure 8: IPFS console responses*

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. Figure 9 below shows the popup from Metamask displaying the gas fees and asking for confirmation.



*Figure 9: Metamask transaction confirmation request*

### 4.1.3 Play Music

The homepage contains all uploaded songs and the user simply clicks on any song they would like to stream. Once the metamask transaction is processed, the song begins to play. The overall flow and implementation is as follows:

1. User clicks on a song and the frontend retrieves the ID and IPFS hash from the smart contract
2. A Metamask transaction is initiated with the cost equal to the cost per stream set by the artist + gas fees of approximately 0.0001 MATIC (~HK\$0.001)
3. On successful transaction, the frontend sends the ID and IPFS hash to the backend
4. The backend uses the ID to get the encryption key and IV from the centralized database
5. The backend fetches the encrypted audio buffer from IPFS using the hash
6. The backend decrypts the file with the information from step 4 and sends it as a base64 string to the frontend
7. The frontend converts this base64 string to a Blob object
8. The frontend also creates a temporary unique URL using `createObjectURL()` for this Blob which expires if the window is closed or the track is changed. The `URL.createObjectURL()` is a static method which generates a `DOMString` containing a URL representing the Blob object passed as parameter [29].
9. The audio player takes this URL as the src input and plays the song.

### 4.1.4 Album Artwork

An important component to recognizing a song or album is the album art of that song or album. BeatChain allows artists to upload album art along with the audio file. The following steps explain how album art is managed and displayed on the system:

1. Artists may upload artwork along with the audio file
2. If no artwork is uploaded, a default generic one is used for all songs without an album art

3. If uploaded, the artwork is sent to backend along with other data as explained in the upload feature
4. The backend uploads the artwork to IPFS and stores the hash to the centralized database
5. The backend also returns this hash to the frontend which then saves it on the contract
6. The artwork is then accessed using the IPFS hash

#### **4.1.5 Searching and Sorting**

BeatChain provides a search bar to search for particular songs. It also has a sorting feature which can sort by release date, artist and title. This is implemented on the frontend alone using React's state object. The array of songs is saved as a state object in React. On rendering, the array elements are filtered to match the search query so that only those songs are displayed. For sorting, based on the filter applied, the output array is rendered.

#### **4.1.6 Playlists**

Playlists are a vital feature of any music streaming service and BeatChain currently provides the option to create public playlists which all other users can access. The functionality is basic and similar to other streaming services. The implementation and flow for different use cases is explained below.

- Create playlist
  1. User clicks on the Playlist button on the frontend and sets a title
  2. Frontends sends a post request to the backend to create an empty playlist
  3. The backend saves a new playlist title on the centralized database and initializes the song list to an empty array
  4. The frontend then shows the new playlist in the list of playlists.
- Fetch all playlists
  1. When the frontend loads, it sends a get request to the backend
  2. Backend gets all the playlists in the database and returns

3. Frontend displays playlist titles in a dropdown and has the details of which playlist contains which songs in a state object
- Select a playlist
    1. After the frontend fetches all playlists, users can select one from the dropdown
    2. The Javascript filter function is used to get those songs with song IDs stored in the playlist array
    3. These songs are then displayed instead of all songs
  - Add song to a playlist
    1. After selecting a playlist from the dropdown, the user clicks on Add Song
    2. The frontend displays a list of all songs that are not already in the playlist
    3. User selects a song from the above list
    4. The frontend sends a post request to the backend with the song id
    5. The backend finds the playlist by id and adds the song to that playlist the database
    6. The backend responds with the updated playlist to the frontend

## **4.2 Challenges & Constraints**

### **4.2.1 Encryption & Protection against Piracy**

The audio file is encrypted and stored in IPFS with the encryption keys stored in the centralized database. The goal of the encryption mechanism is to ensure that the audio file is protected even if the IPFS hash is leaked. It also protects the file from being accessible to the storage node. Therefore, the file is secure in storage and will be safe even in case of leaks.

However, there is a vulnerability in the service layer. When a user streams a song, the backend decrypts the file and sends it to the frontend. As described in Section 4.1.3, the frontend receives the decrypted file and creates a temporary URL for the audio player source. This causes the vulnerability. It is possible to get the decrypted file from the URL and distribute it. In fact, this issue is not limited to our platform. Popular streaming services like Netflix and Amazon Prime also face the issue of piracy with their content

openly available for free on websites like FMovies. Apart from this, there are widespread operations which pirate content from streaming platforms despite their use of licensed Digital Rights Management (DRM) technologies like Widevine. According to an article by Aditya Tiwari, pirates make use of the loophole that the decrypted file needs to be stored on the user's machine for streaming [21]. The author further discusses how these operations have been automated and any new releases on Netflix or Amazon Prime can be available on pirated platforms in a matter of hours. Netflix reportedly loses around US\$200 million every month due to piracy. This constraint, therefore, seems to be industry-wide and there is a big room for future improvement on this front. A step forward in solving this issue will be a large benefit for the entire streaming industry.

#### **4.2.2 Semi-Decentralized Implementation**

Decentralized applications are still new with constraints on efficiency and costs. The time taken to mine blocks and verify transactions can make the user experience worse when there are many transactions in a short time [22]. Moreover, operations such as searching and sorting on blockchain are computationally expensive. As a result, it is difficult to implement a completely decentralized architecture to support all functionality of a streaming platform. As mentioned in Section 3.6, it is currently a common practice to have semi-decentralized architectures in blockchain applications. In BeatChain, the important transaction and copyright protection related work is on the chain with the centralized database and server playing supporting roles to allow more functionality and UX improvements. With future improvements in Ethereum and IPFS, the architecture can be made a fully decentralized one.

#### **4.3 Future Vision**

BeatChain is currently a blockchain based music streaming platform with the features as described in Section 4.1. However, given more time, there are various improvements the team would like to implement. These features were not implemented on priority because they were out of scope of the project and the team's focus was on developing

an end to end fully functioning system. Mentioned below are some of the features that the team considers as important future improvements.


- Shared distribution of royalties - BeatChain supports only single artist uploads but many songs are created by many artists. The smart contract can be updated to divide royalties on a pre-decided proportion between all these artists.
- Support payment via multiple cryptocurrencies - This is the web2 equivalent of having different payment mode options. To make it more convenient for users, the system can accept multiple cryptocurrencies as stream or upload payments.
- Song recommendations - An important feature of a music streaming service is to provide song recommendations or play similar songs to the one already playing. This would require extensive use of Machine Learning and can be an entire project in itself.
- Private playlists - The playlists created right now are visible to every user. Users might want to create playlists that others cannot view. A simple way to implement this would be to assign a boolean tag to each playlist to represent whether it is private or public and render based on the creator id.
- Mobile application - Streaming via a mobile app is an important part of user experience. It is inconvenient for users to use a web application to stream music, especially on mobile devices. The Metamask mobile app can be used for blockchain transactions on mobile devices.
- Blockchain network with even lower gas fees - New network protocols and improvements to current ones are constantly being researched. With lower gas fees, transaction costs are further reduced thereby making streaming more cost effective.


#### **4.4 Project Timeline**

Figure 10 below outlines the timeline and development schedule that the team followed.



*Figure 10: Ideation & Development Timeline*

 = Semester 1

 = Semester 2

## 5. Business Model

BeatChain adopts a pay-per-stream business model and charges 15-25% commission per stream. The remaining amount goes completely to the artist. A pay-per-stream model entails that there is no fixed one-time subscription fee but rather a very small amount paid per stream of a song. This section will discuss the pay-per-stream model in comparison to the subscription model. It will also do a cost analysis and market comparison with streaming giants Spotify and Apple Music.

### 5.1 Pay Per Stream vs Subscription

While pay per stream might be uncommon as a business model, it provides considerable benefits in the music streaming industry and is preferred over the freemium subscription model. The primary benefit in the pay per stream model is the flexibility. Users need not commit to an up-front cost for a month or year and therefore can choose to not use the service for any period without the feeling of wasting money. Moreover, charging a fixed subscription cost makes it difficult for the platform to distribute royalties fairly. For example, according to their website, Spotify does not pay artists on a per-stream basis but rather based on their deals with record labels [23]. This means Spotify pays some artists even if their music is not streamed, therefore making revenue distribution unfair. In the pay per stream model, the revenue from users is not fixed and therefore there is no need for the platform to negotiate a fixed fee with the artist. The artist's revenue simply depends on the number of streams. The pay per stream model also allows for an ad-free listening experience at far lower costs as we will see in Section 5.2.

There are, of course, benefits to the subscription model and problems with the pay per stream model. The subscription model helps increase user stickiness on the platform because they control the fickle-minded nature of users. Once a user pays the subscription for a period of time, they will usually use the subscription till expiry even if they find a better alternative. It is important to note here that a longer term subscription is usually cheaper, thereby incentivizing users to stay on the platform for a longer period. This also makes users get used to the platform and increases re-subscription



rates. The streaming platform can also use data driven analytics to make users resubscribe. The pay per stream model lacks this quality of customer retention because shifting to another platform is costless. Moreover, revenues are unpredictable in such a setup. Finally, because the users are forced to pay per stream, they might think it is more expensive and feel uncomfortable using such a system. The marketing campaign to inform users and artists of the monetary benefits will have to be very convincing. Overall, when the number of users is higher, the benefits in the subscription model seem to be of incentive to the business only while the pay per stream model benefits the artist, user and business.

## 5.2 Cost Analysis & Market Comparison

This subsection conducts an average case cost analysis based on historical data. It then compares the user costs and artist revenues on BeatChain with those on Spotify and Apple Music, the two biggest music streaming services right now.

As discussed in Section 1.2, publishers and record labels charge high commissions and reduce artist revenues. On average, publishers charge 50% as commission for their services [24]. Record label commission differs by the kind of deal the artist makes with them and the services they provide to the artist but it usually ranges between 10 and 35% [25]. This means artists make only about 15 to 40% of streaming revenues, with the best case scenario implying a very basic record label deal which might not provide them much utility. In contrast, BeatChain charges an overall average commission of 20% with the remaining 80% going directly to artists. This is possible because of the elimination of middlemen and automation of all processes.

Table 1 below contextualizes the above information **in HK\$ per 1000 streams**.

	Spotify [26]	Apple Music [27]	BeatChain
Avg pay	29	60	20
Publisher Commission	14.5	30	0
Avg Record Label Commission	6.525	13.5	0
Avg Total Commission	21.025	43.5	4
Avg Artist income	7.975	16.5	16

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

As demonstrated in the table above, at a cost of HK\$0.02 per stream, BeatChain can give artists revenues of more than 200% than Spotify and about the same as Apple Music. Let us now analyze what the cost per stream can be on BeatChain based on average usage. According to Business of Apps, average global monthly Spotify usage is 3540 mins per user [26]. For Apple Music, this number is slightly higher at 3960 mins per user [27]. For the simplicity of the analysis, let us take the average per user listening time as 4000 mins and the average song duration to be 4 mins equalling a 1000 streams on average per user per month. Both Spotify and Apple Music have regular monthly subscription costs of HK\$58. On BeatChain, for a cost per stream of HK\$0.02 (considered in the table above), an average user would pay only HK\$20 for the same amount of usage and better or equal artist revenues. Thus, at HK\$0.02 per stream, using BeatChain cannot be worse for artists and is almost 1/3rd the cost for users.

Finally, let us analyze the possible usage and artist revenues if BeatChain cost the same as Spotify and Apple Music at HK\$58 / month. At HK\$0.02 per stream, a user can stream 2900 songs equalling 11,600 mins on average. This is 4 times that of Spotify and Apple Music at the same cost. If we fix the listening time to 1000 streams, for the same user cost, cost per stream can be as high as HK\$0.058 out of which the artist gets HK\$0.0464. This is more than 580% compared to Spotify and more than 280% compared to Apple Music. Therefore, in the range of HK\$0.02 and HK\$0.058 per stream,

artists generate higher revenues on average than Spotify and Apple Music while users pay less on average.

## 6. Conclusion

The music industry is currently expensive and inefficient at distribution and licensing. As a result, smaller artists find it difficult to make a feasible career in the industry. This project aims to democratize participation in the music industry and the team has developed a streaming service on the Polygon Matic network called [BeatChain](#). Using a smart contract, the system automates the royalty collection of streams of songs stored on IPFS. Current features on the system include song uploads and streams, decentralized storage of album art for every song, searching and filtering for songs, and public playlists. Audio files are also protected using encryption and a centralized database. This encryption is, however, not completely secure due to an industry-wide constraint of having to store decrypted files on user devices. Given more time, the team would like to implement features like song recommendations and a mobile app to improve user experience. The team has also conducted an average case business analysis to demonstrate the feasibility of such a product. The proposed model is a pay-per-stream setup which on average increases revenues for artists and reduces costs for users when compared to Spotify and Apple Music, the biggest streaming services currently available. Overall, this project demonstrates the need and practical possibility of a decentralized music streaming service which can benefit both artists and listeners. With improvements in blockchain networks everyday and better protocols with lower gas fees, such a service will become more lucrative in the future. The team hopes that our project inspires more research and development in this space.

## 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](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](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] "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](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). [Accessed: 30-Sep-2021].

- [6] 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>.
- [7] SingularDTV, "Introducing the Breaker Royalty Management Platform," *Medium*, 24-Jun-2021. [Online]. Available: <https://singulardtv.medium.com/introducing-the-breaker-royalty-management-platform-60a819b80c1e>
- [8] "Music fans share ownership with artists in their favorite songs," *Vezt*. [Online]. Available: <https://vezt.co/>
- [9] M. Nofer, P. Gomber, O. Hinz, and D. Schiereck, "Blockchain," *Bus. inf. syst. eng.*, vol. 59, no. 3, pp. 183–187, 2017.
- [10] 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/>.
- [11] 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
- [12] "The crypto wallet & gateway to web3 blockchain apps," *MetaMask*. [Online]. Available: <https://metamask.io/>. [Accessed: 30-Sep-2021].
- [13] Shawn Wang, "The difference in five modes in the AES encryption algorithm," *Highgo Software Inc*. [Online]. Available: <https://www.highgo.ca/2019/08/08/the-difference-in-five-modes-in-the-aes-encryption-algorithm/>. [Accessed: 18-Feb-2022].

- [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] "IPFS powers the distributed web," *IPFS Powers the Distributed Web*. [Online]. Available: <https://ipfs.io/>
- [18] "Ethereum API: Ipfs API & gateway: ETH Nodes as a service," *Infura*. [Online]. Available: <https://infura.io/>
- [19] "Semi decentralised applications (Semi-DApp)," *Blockchain Patterns*, 01-Sep-2021. [Online]. Available: <https://research.csiro.au/blockchainpatterns/general-patterns/deployment-patterns/semidapp/>. [Accessed: 17-Mar-2022].
- [20] "Web3.js - Ethereum javascript API," *web3.js - Ethereum JavaScript API - web3.js 1.0.0 documentation*. [Online]. Available: <https://web3js.readthedocs.io/en/v1.7.3/>. [Accessed: 29-Dec-2021].
- [21] A. Tiwari, "This is how your favorite netflix movies and shows are pirated," *Fossbytes*, 12-Jun-2021. [Online]. Available: <https://fossbytes.com/how-pirate-netflix-amazon-prime-movies-shows-piracy/>. [Accessed: 12-Apr-2022].
- [22] K. L., "The blockchain scalability problem & the race for visa-like transaction speed," *Medium*, 23-Jul-2019. [Online]. Available: <https://towardsdatascience.com/the-blockchain-scalability-problem-the-race-for-visa-like-transaction-speed-5cce48f9d44>. [Accessed: 28-Feb-2022].

- [23] "Help - Royalties – Spotify for artists," *Fans Make it Possible – Spotify for Artists*. [Online]. Available: <https://artists.spotify.com/en/help/article/royalties>.
- [24] "Music Publishing explained: How artists get paid for their songs," *Music Careers / Expert Advice - Careers In Music*, 11-Jan-2022. [Online]. Available: <https://www.careersinmusic.com/music-publishing/>. [Accessed: 10-Apr-2022].
- [25] Ucaya, "Market intelligence for the music industry," *Soundcharts*. [Online]. Available: <https://soundcharts.com/blog/splits-and-profits-record-deals-analysis>. [Accessed: 10-Apr-2022].
- [26] "Spotify revenue and Usage Statistics (2022)," *Business of Apps*, 19-Jan-2022. [Online]. Available: <https://www.businessofapps.com/data/spotify-statistics/>. [Accessed: 10-Apr-2022].
- [27] "Apple Music Revenue and Usage Statistics (2022)," *Business of Apps*, 14-Apr-2022. [Online]. Available: <https://www.businessofapps.com/data/apple-music-statistics/>. [Accessed: 10-Apr-2022].
- [28] "Get estimation of confirmation time," *Etherscan*. [Online]. Available: <https://docs.etherscan.io/api-endpoints/gas-tracker>. [Accessed: 17-Apr-2022].
- [29] "URL.createObjectURL() - web apis: MDN," *Web APIs / MDN*. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/API/URL/createObjectURL>. [Accessed: 18-Mar-2022].