

# Introduction to Blockchain and CryptoCurrency



## Project Progress Report

[GitHub Link](#)

Adwit Singh Kochar (2018276)

Anuneet Anand (2018022)

Pankil Kalra (2018061)

# About

Music has always been an essential part of our lives, and it is a way for many artists to express themselves. The music industry has grown exponentially in the past few years with the success of digital music streaming services. People can now access music from various parts of the world using their computers and smartphones.

However, the major music streaming platforms are centralized, and as a result, the artists lack control over their content and receive only a small fraction of the revenue generated by these platforms. The current system requires artists to depend on intermediaries like studio companies, record labels, and streaming service companies for managing and marketing their work. A technology that can improve transparency and provide more control to the artists over the money they make can revolutionize the industry. The direct interaction between consumers and artists would also eliminate the need for intermediaries from the revenue stream.

We propose **blockstudio** - a blockchain-based decentralized music streaming platform to connect music enthusiasts directly to independent music artists. The artists can use this platform to share their music with greater freedom while ensuring ownership and avoiding duplication of their music. People can listen to their favourite songs and support the artists by making micropayments. The platform can be hosted as a DApp on the Ethereum blockchain.

# Week - 1

We studied various technologies that are used in making DApps from various online resources like blogs, tutorials and videos. Since most of these technologies were new to us, we spent a good amount of time getting familiar with them. We also read about some concepts related to blockchain and cryptocurrency in general.

## Frontend

1. **React:** It is an open-source Javascript library used for frontend development. We use it because it is easy, fast and has a lot of support in the form of tutorials and forums. Many DApps we explored use React to develop their frontend and thus we follow the same.

## Backend

1. **Web3.js:** It is an extensive collection of libraries that will help our web app to communicate with an Ethereum blockchain using JSON based Remote Procedure Calls.
2. **IPFS (InterPlanetary File System):** It is a distributed storage system that utilizes Peer-to-Peer (P2P) networking. Unlike traditional “location-based” storage systems, IPFS is “content-based” as it maintains unique global hashes of all the files. This gives it many advantages like independence from a single controlling entity and preventing duplicate files, thus being perfect for implementing our DApp.

## **Blockchain**

1. **DApp:** A decentralized application uses a decentralized system to perform different tasks. DApps help the user interact with the smart contract programs stored on a blockchain.
2. **Solidity:** It is the high-level language used to write smart contracts that can be stored on the blockchain. This will define the logic behind our blockchain, dictating the behaviour of all the accounts.
3. **Ganache:** It is used to create a personal local Ethereum blockchain, whose dummy accounts can be accessed by a cryptocurrency wallet and used in a DApp. This helps us run our DApp like it is connected to the actual Ethereum network without long transaction times.
4. **MetaMask:** It is a crypto-wallet that can be accessed through a browser extension. We can use this to interact with our local blockchain created using Ganache.
5. **Truffle:** It is the development environment that will be used to develop smart contracts and DApps. It can be used for the compilation, testing as well as deployment of smart contracts.

## **Week - 2**

We explored projects on GitHub which have a similar use case as ours and obtained insights into the features which we should implement and also technologies that we can use. Short descriptions of some of them:

### **Resound**

- A project by Julien Tregoeat from the USA.
- It is a decentralized music marketplace for independent artists and labels.
- It uses Ethereum Virtual Machine (EVM) for transaction logic and as a database for primitive data types.
- IPFS (InterPlanetary File System) is used as a location for blob storage.
- Limitations: No check for piracy, Weak user authentication, Conversion errors due to direct use of ETH for transactions.

### **musicDAPP**

- A project by Raj Chakravarthy from India.
- It is a decentralized concert ticketing system.
- It has different user roles like Organisers who can create and deploy concerts using smart contracts and Users who can buy tickets to the deployed concerts.
- Images of the concert are stored on IPFS.
- Blockchain properties help to ensure that each user can buy only one ticket to each concert and no user can use some other user's ticket.
- The developer has tried to achieve minimum usage of Truffle to prevent code breaks.
- The developer has achieved variable-based routing using Next.js

## jelly-beats

- Music player built on blockchain.
- It is now archived and developers have moved on to [Hound.fm](https://hound.fm).
- Does not have a lot of documentation.
- It is a desktop app built using Node.js and React with Electron on top.
- The most interesting part was that it used LBRY (LBRY is an open-source protocol providing distribution, discovery, and purchase of digital content via a decentralized network).
- The app currently does not work when installed through code (hound.fm works though). It's most probably due to the current version of LBRY not being supported.
- Code is comparatively more complex than other projects that have been explored.

## Week - 3

We brainstormed and listed potential features of the DApp which we aim to implement.

### **1. Two types of users:**

- **Artists**

- Can post songs and track activity on their songs.
- Can assign fees for listening to songs after the free-plays have been exhausted.
- Can also choose to keep the song free for all.

- **Audience**

- Can browse artists and songs.
- Can get access to a song by paying the fees as defined by the artist.

**2. P2P Storage System:** All data will be stored on a distributed database like IPFS and songs will be tested for duplication.

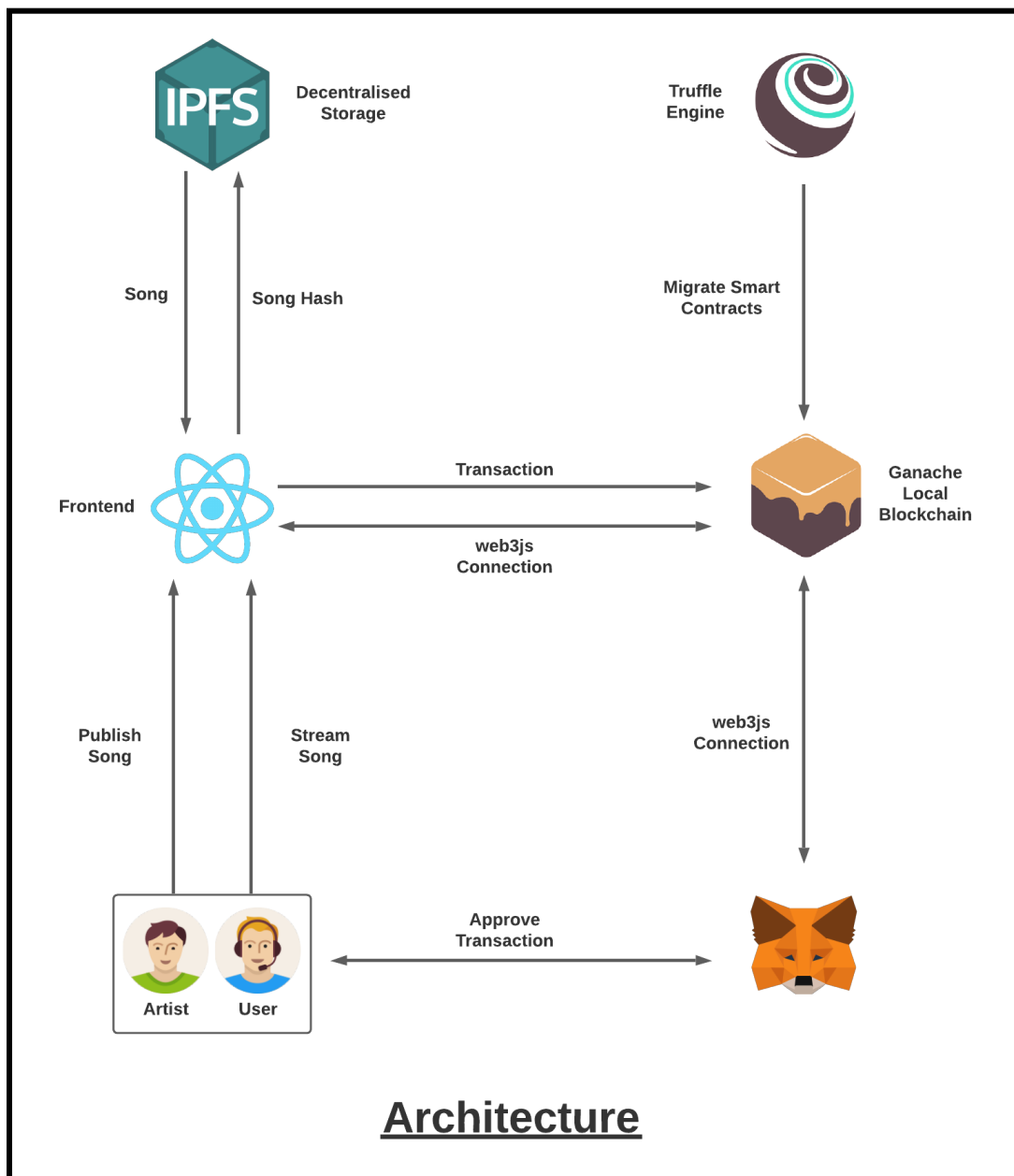
**3. Cryptocurrency:** All payments will be made in milliwei using Metamask Wallet. The wallet can be refilled using ETH funds.

**4. CrowdFunding:** Support favourite artists by funding their albums and tracks.

**5. Survey System:** Popularity scores would be generated for the various artists using the platform.

## Week - 4

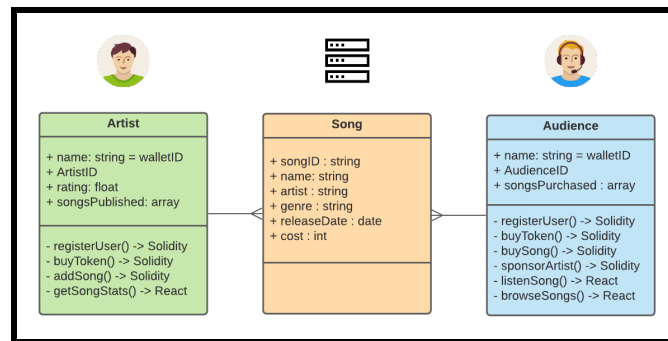
We designed a tentative architecture of our DApp and set up the code repository with the starter code.





## Week - 5

We brainstormed over the data structures and functions we would need to implement in Solidity to support the various proposed features of our DApp. This involved defining the Artists (people who generate music), Audiences (people who listen to music generated by Artists), and Songs along with their associated functions.



We were able to produce the first iteration of the smart contract for our application. The smart contract was tested on our local Ganache blockchain using Truffle.

```
struct Artist {
    string name;
    uint256 ArtistID;
    uint256 rating;
    uint256[] songsPublished;
}
```

```
struct Audience {
    string name;
    uint256 AudienceID;
    uint256[] songsPurchased;
}
```

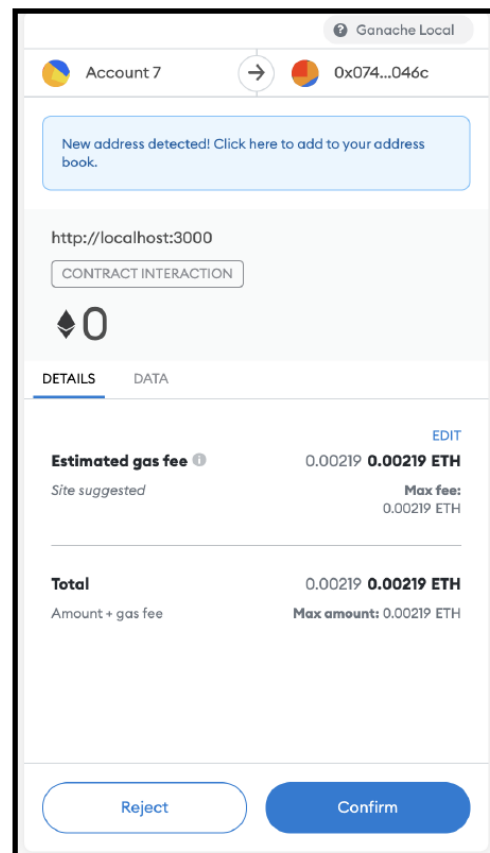
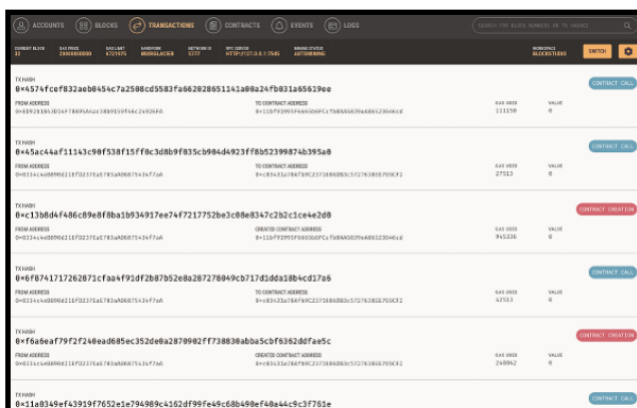
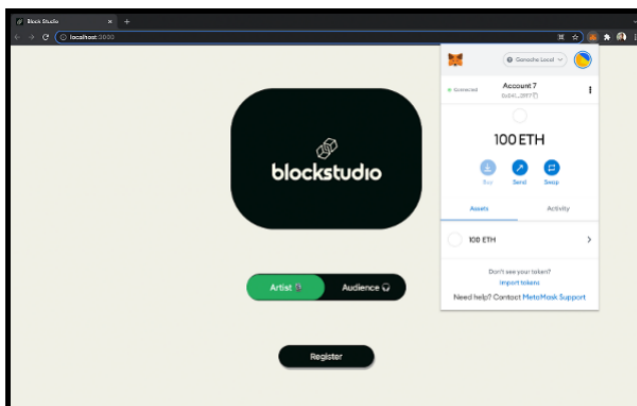
```
function addNewArtist(string memory _name) public {
    artistIDTracker += 1;

    Artist memory newArtist;
    newArtist.name = _name;
    newArtist.ArtistID = artistIDTracker;
    newArtist.rating = 0;

    allArtists[msg.sender] = newArtist;
    identifyUser[msg.sender] = UserType.ARTIST;
}
```

## Week - 6

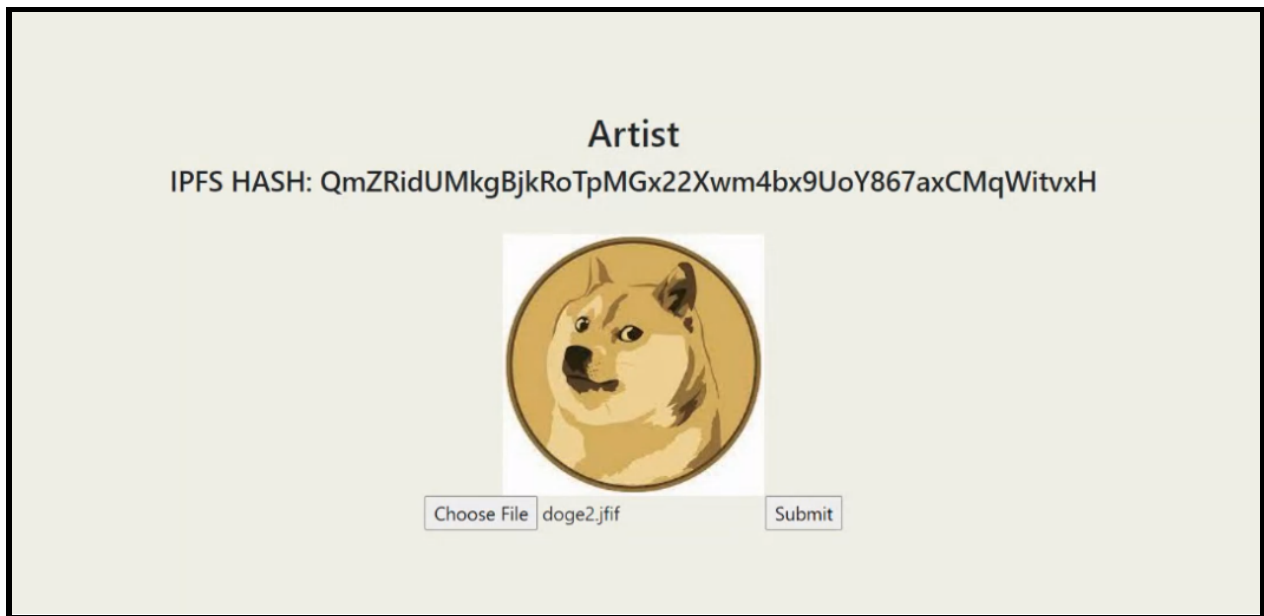
We designed a basic frontend to interact with our DApp and test the functionality. We further worked on integrating our smart contract with our React frontend. The login page (as shown below) allows a new user to register as Artists or Audiences and a transaction is made for their registration on the local Ganache Blockchain. The DApp will automatically detect his/her profile type on future login to make sign-in easier. This segregation is also useful to split the Artists and Audience functionalities.



## Week - 7

We continued work on smart contracts and frontend. The IPFS storage system was integrated with the React frontend to enable storage of Artist and Song metadata (images, audio files, etc.). Artists will be allowed to upload new songs and images to the storage system, which then can be browsed by the audience.

In the example shown here, we upload a test image to IPFS storage and get back its hash values. This process will be used to store album art, artist photos and audio files.

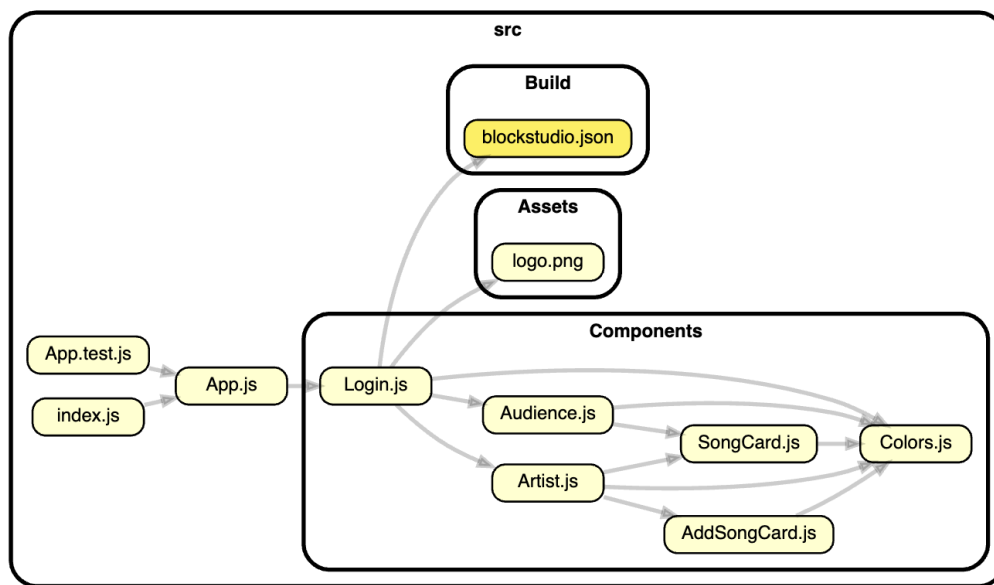


### Image Link

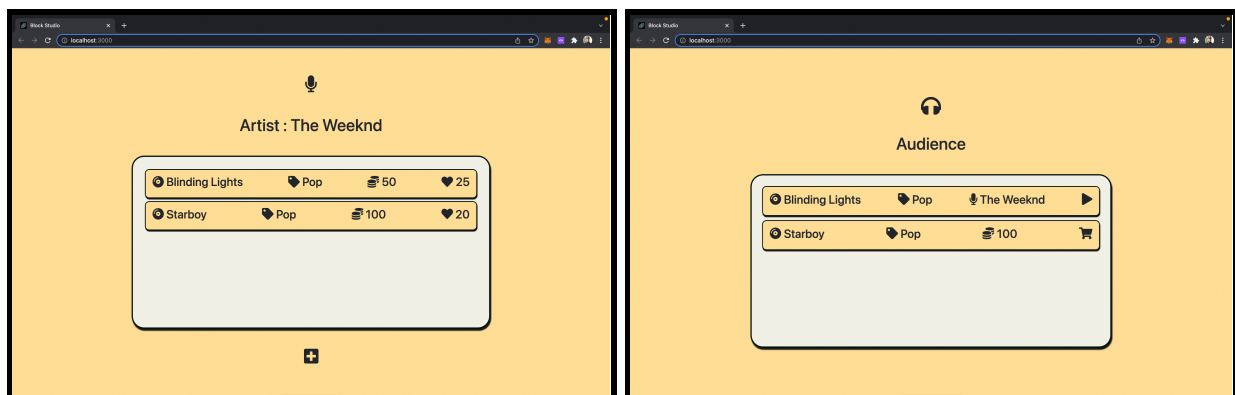
<https://ipfs.infura.io/ipfs/QmZRidUMkgBjkRoTpMGx22Xwm4bx9UoY867axCMqWitvxH>

## Week - 8

We focused on the frontend layout & design and accordingly created react components for our DApp. We also implemented the associated logic for our react components and tested the functionality. The following diagram provides an overview of our component hierarchy.



The following screenshots show the custom-designed user interface of our DApp with a minimal and retro theme.



## Week - 9

We defined the structure for Song on our platform and created solidity functions for the functionality of adding and buying songs in our smart contract. They were defined with appropriate checks such as:

### 1. addSong

- Only artists can add a song.
- The IPFS hash of the song must not be already in use (to avoid duplicates).

### 2. buySong

- Only Audience members can buy a song.
- Audience members cannot buy a song twice.
- Enough funds should be available in the wallet to buy the song.
- The right amount should be transferred to the Artist of the song.

Transfer of funds takes place through the *buySong* function in the denomination of **Wei** (1/1e18 ether).

The associated solidity code snippets are given below.

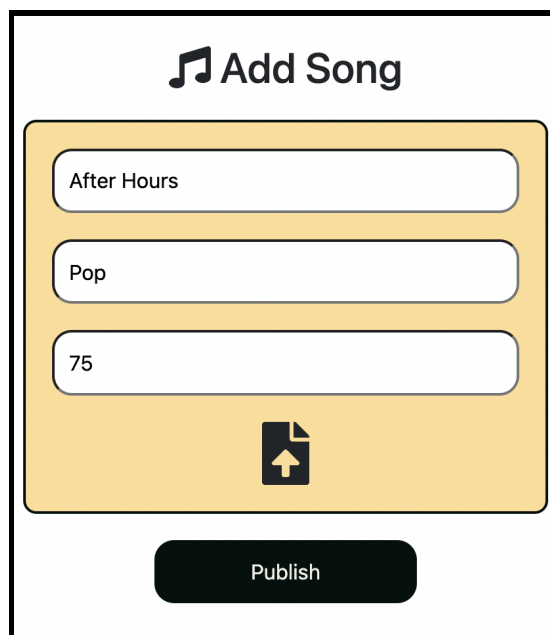
```
event songPurchased (
    uint songID,
    string songName,
    string audienceName,
    uint price
);
```

```
event songAdded (
    uint songID,
    string songName,
    string artistName,
    uint price
);
```

```
struct Song {
    string songName;
    string artistName;
    string genre;
    string hash;
    uint songID;
    uint price;
    address payable artistAddress;
}
```

## Week - 10

We integrated the solidity functions with the react frontend of our DApp. Each artist can upload a song file and add its name, genre and cost. The artist page shows a list of all the songs added by him/her along with their attributes and number of likes.




The image shows a web form titled "Add Song" with a musical note icon. The form has a yellow background and contains three input fields: "After Hours" for the song name, "Pop" for the genre, and "75" for the cost. Below these fields is a file upload icon (a document with an upward arrow). At the bottom of the form is a black "Publish" button.

The uploaded songs are stored on the IPFS Infura service. For each song a unique hash is generated which corresponds to its entry in the database and thus can be used to access the song. This hash is stored on the blockchain using the contract. A songID is also created to track the songs published by the artists and purchased by the audience. The buySong functionality is added to the audience along with the functionality to retrieve purchased songs from IPFS and play them.


## Week - 11

We added a feature for the audience using which they can make donations to their favourite artists to show their support. The audience needs to provide the artist username and donation amount in milliwei. The amount is directly transferred to the respective artist's wallet.

  
**Sponsor  
Artist**

**Donate**

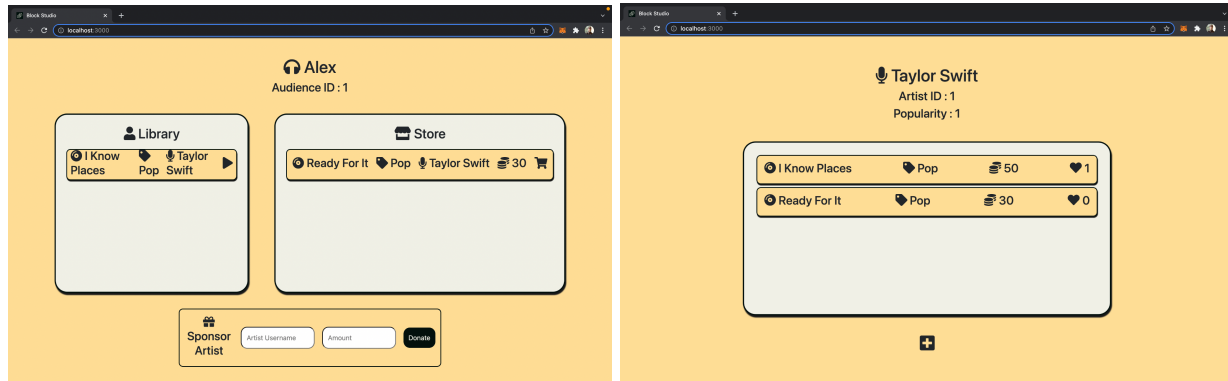
We also added a popularity measure for the artists which is calculated based on the number of purchases made by the audience on their songs. The artist can also view the number of purchases of individual songs to get an idea of how many people like his/her work.

 **Taylor Swift**  
**Artist ID : 1**  
**Popularity : 1**

To prevent duplication of media files, we have stored hashes for the song files added by the artist. If someone tries to add a song file which already exists in the system, the transaction would fail.

## Week - 12

We worked on improving the user interface of our DApp and making it consistent across different pages.



We also thoroughly tested each and every functionality and accordingly fixed the bugs encountered. The code was refactored and unnecessary code snippets were removed. We revisited our smart contract to make improvements wherever possible.

After wrapping up the project, we focused on completing the necessary documents and making a presentation. We also created a short demo video where we explain the working of our project.

In retrospect, undertaking this project was a great learning experience. We were introduced to various new technologies and were able to apply the learning of IBC course to our project.



# Work Distribution

- **Week 1:** All of us explored the basics of the technologies mentioned.
- **Week 2:** Anuneet explored the resound project, Adwit explored the musicDAPP project and Pankil explored the Jelly-Beats project.
- **Week 3:** All of us brainstormed together to come up with potential features. Adwit proposed we should use a custom token for our system to prevent the transaction of Eth in decimal values. Anuneet proposed we can add a crowdfunding feature where users can support their artists. Pankil proposed that we should also display some stats about songs and give users the ability to rate songs.
- **Week 4:** All of us collectively worked on the architecture diagram and setting up the repository.
- **Week 5:** All of us discussed the possible structure of the smart contract. Adwit came up with a basic contract in solidity which was further tested and improved by Anuneet & Pankil.
- **Week 6:** Anuneet worked on designing the user interface and linking the app with web3, while Pankil & Adwit focussed on deploying and integrating the smart contract.
- **Week 7:** Pankil worked on integrating the IPFS, while Anuneet and Adwit continued with developing smart contract and frontend.
- **Week 8:** Anuneet and Adwit brainstormed the required attributes and functions in react to support the DApp and developed the Frontend.
- **Week 9:** All of us discussed the structure of the new functions to be implemented in Solidity and wrote the code with subsequent testing.
- **Week 10:** Pankil focused on file handling, IPFS functionality and its integration while Anuneet and Adwit integrated the react code with the smart contract.
- **Week 11:** Anuneet focused on adding the support artist feature while Adwit looked at hash generation and Pankil focused on the popularity calculation.
- **Week 12:** All of us tested the DApp, refactored the code and prepared the necessary deliverables for final submission.

## References

- <https://reactjs.org>
- <https://web3js.readthedocs.io/en/v1.5.2/>
- <https://ipfs.io>
- <https://docs.soliditylang.org/en/v0.8.9/>
- <https://www.trufflesuite.com>
- <https://metamask.io>
- <https://ethereum.org/en/dapps/>
- <https://www.youtube.com/watch?v=nU0uqk0jLdw>