Blockstudio

Team Members

Adwit Singh Kochar

2018276

Anuneet Anand

2018022

Pankil Kalra

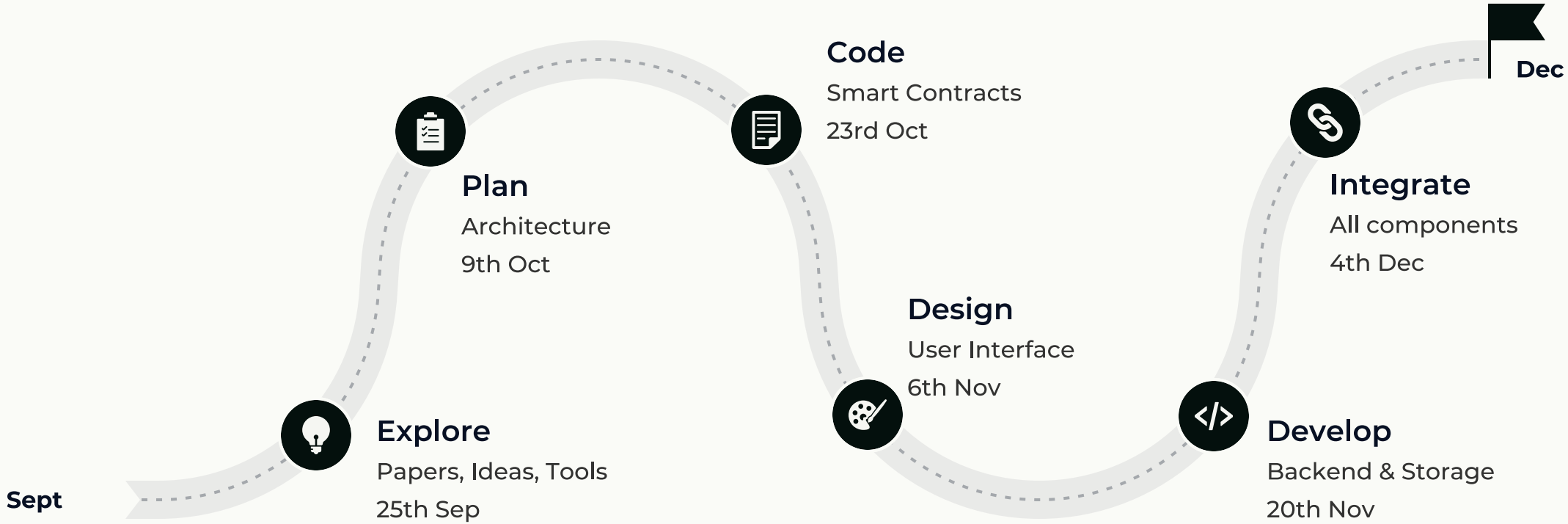
2018061

Motivation

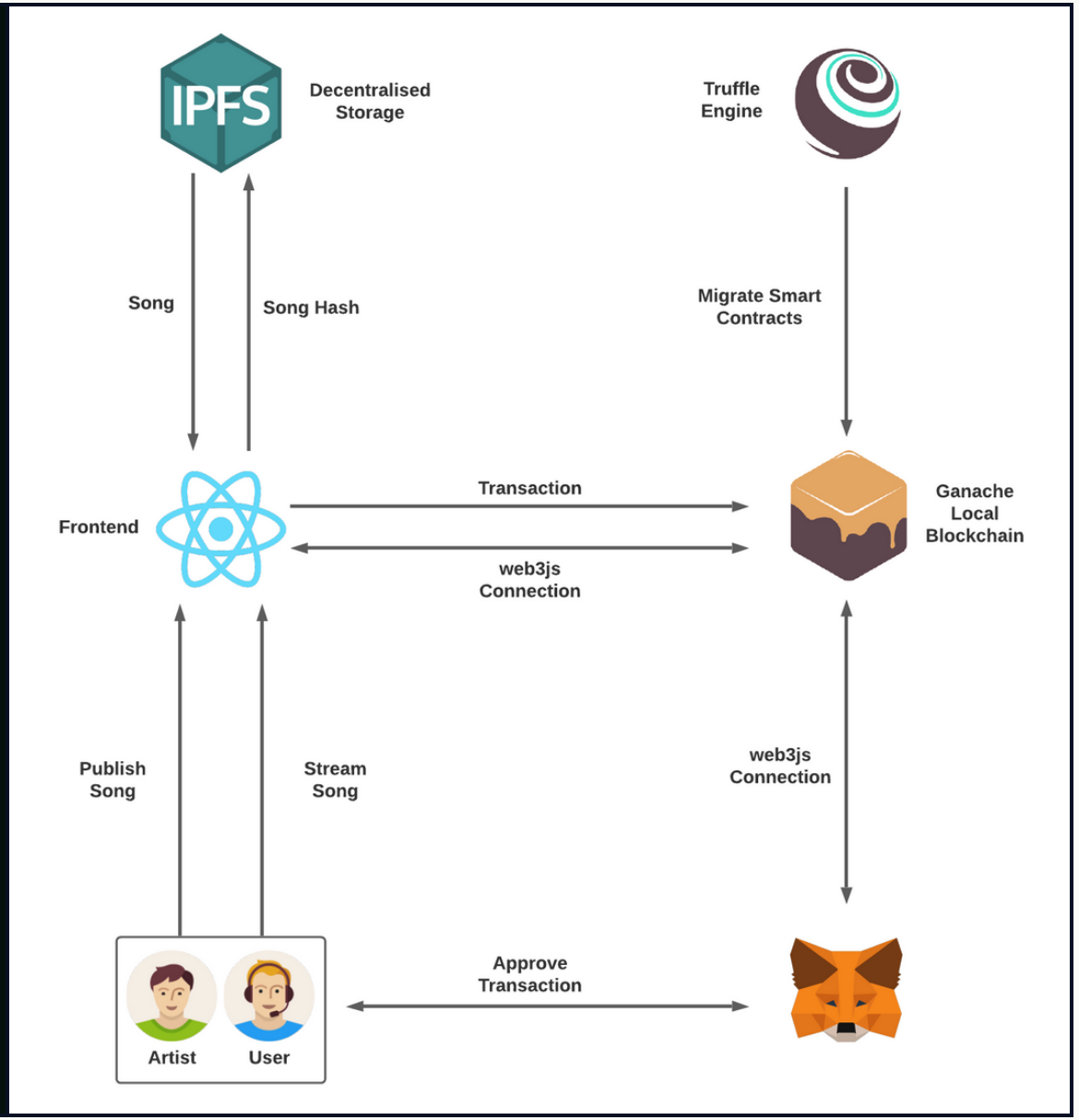
The music industry has grown exponentially in the past few years with the success of digital music streaming services. However, the major music streaming platforms are centralized, and as a result, the artists lack control over their content.

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.

Project Timeline



Architecture



Smart Contracts

Some snippets of the smart contracts used in the project.

```
contract blockstudio {
    uint256 audienceIDTracker;
    uint256 artistIDTracker;
    uint256 songIDTracker;

    enum UserType {
        UNDEFINED,
        ARTIST,
        AUDIENCE
    }

    struct Artist {
        string name;
        uint256 artistID;
        uint256[] songsPublished;
    }

    struct Audience {
        string name;
        uint256 audienceID;
        uint256[] songsPurchased;
        mapping(uint256 => bool) isSongPurchased;
    }
```

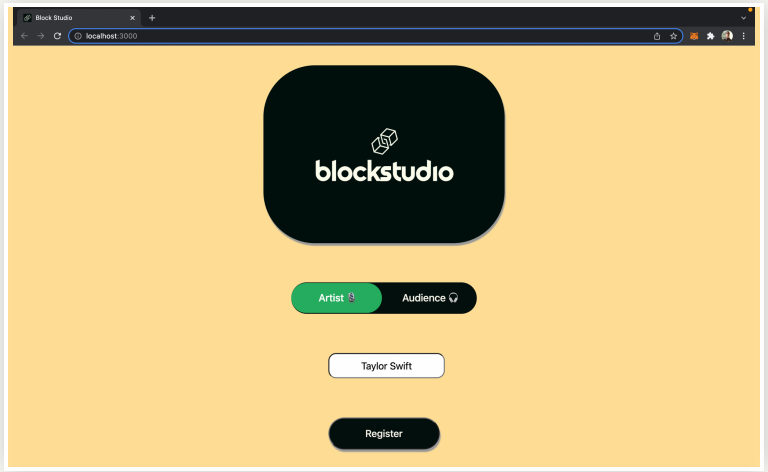
```
function getSongDetails(uint256 _songID)
    public
    view
    returns (
        string memory,
        string memory,
        string memory,
        string memory,
        uint256,
        uint256
    )
{
    return (
        allSongs[_songID].songName,
        allSongs[_songID].artistName,
        allSongs[_songID].genre,
        allSongs[_songID].hash,
        allSongs[_songID].price,
        timesSongPurchased[_songID]
    );
}
```

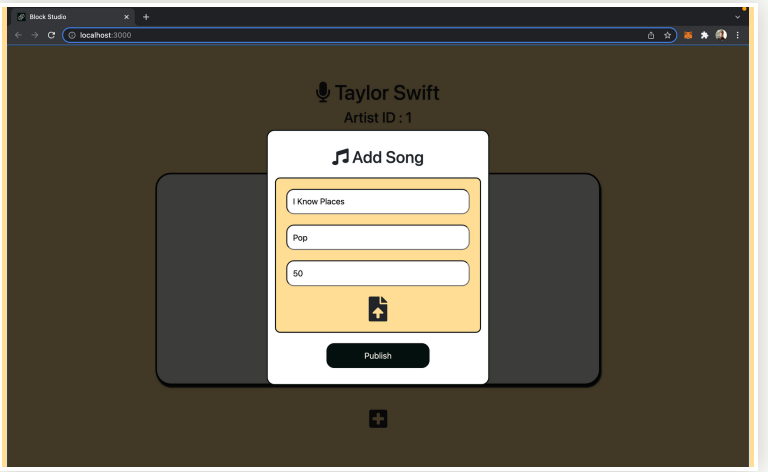
```
struct Song {
    string songName;
    string artistName;
    string genre;
    string hash;
    uint256 songID;
    uint256 price;
    address payable artistAddress;
}

mapping(address => UserType) identifyUser;
mapping(address => Artist) allArtists;
mapping(address => Audience) allAudience;
mapping(uint256 => Song) allSongs;
mapping(uint256 => uint256) timesSongPurchased;
mapping(string => bool) songHashUsed;
mapping(string => address payable) getArtistAddress;

constructor() {
    audienceIDTracker = 0;
    artistIDTracker = 0;
    songIDTracker = 0;
}
```

```
function addSong(
    string memory _name,
    string memory _genre,
    string memory _hash,
    uint256 _price
) public {
    require(identifyUser[msg.sender] == UserType.ARTIST, "Not an artist.");
    require(
        !songHashUsed[_hash],
        "Duplicate detected. Song hash already in use."
    );

    songIDTracker += 1;

    Song memory newSong;
    newSong.songID = songIDTracker;
    newSong.songName = _name;
    newSong.artistName = allArtists[msg.sender].name;
    newSong.genre = _genre;
    newSong.hash = _hash;
    newSong.price = _price;
    newSong.artistAddress = payable(msg.sender);

    timesSongPurchased[songIDTracker] = 0;

    allSongs[songIDTracker] = newSong;
    allArtists[msg.sender].songsPublished.push(songIDTracker);
    songHashUsed[_hash] = true;

    emit songAdded(
        newSong.songID,
        newSong.songName,
        newSong.artistName,
        newSong.price
    );
}
```

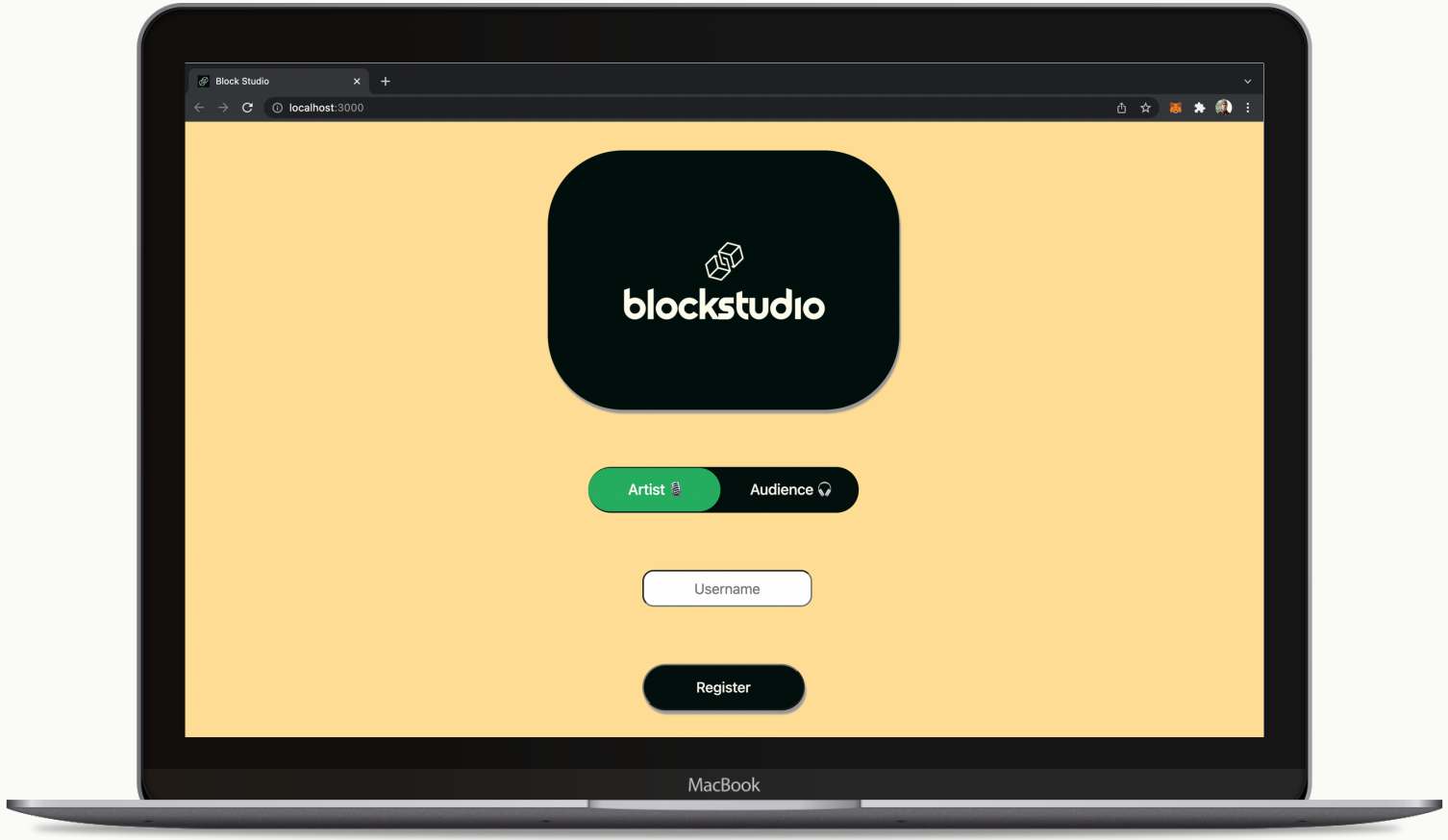
```
function buySong(uint256 _songID) public payable {
    require(
        identifyUser[msg.sender] == UserType.AUDIENCE,
        "Not an audience member."
    );
    require(
        !allAudience[msg.sender].isSongPurchased[_songID],
        "Cannot buy the song twice."
    );

    Song memory curSong = allSongs[_songID];

    require(
        msg.value == (curSong.price * 1 wei),
        "Amount payed does not match price of the song."
    );
    require(msg.sender.balance > msg.value, "Insufficient balance.");

    curSong.artistAddress.transfer(msg.value);
    timesSongPurchased[_songID] += 1;

    allAudience[msg.sender].songsPurchased.push(_songID);
    allAudience[msg.sender].isSongPurchased[_songID] = true;

    emit songPurchased(
        curSong.songID,
        curSong.songName,
        allAudience[msg.sender].name,
        msg.value
    );
}
```

A Retro Approach to User Interface





Features



Publish Songs

Artists can publish their songs and track activity on them.

Stream Songs

Audience can purchase songs and add them to their library.

Availability & Security

The songs are stored on a P2P database and checked for duplicates.

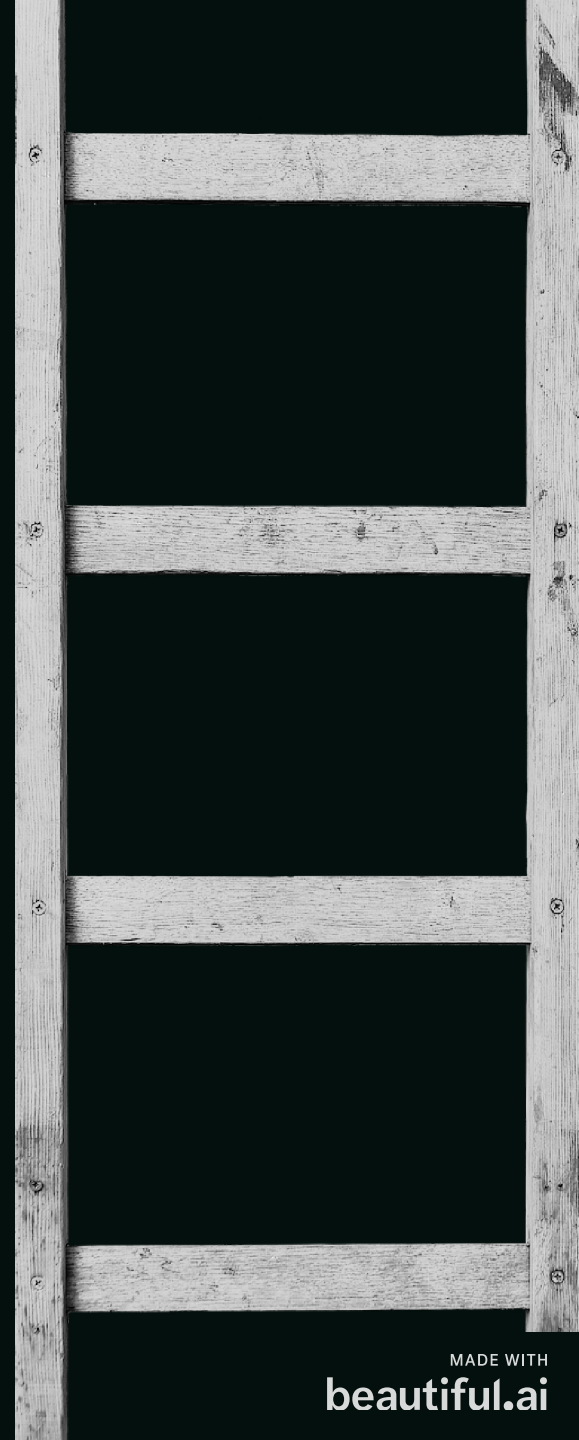
Future Prospects

“Taking the project further one block at a time”

Scale the System

Integrate a Recommender System

Build a Marketplace for Artists to Sell their NFTs



Thank You