CSE 426LEC Blockchain

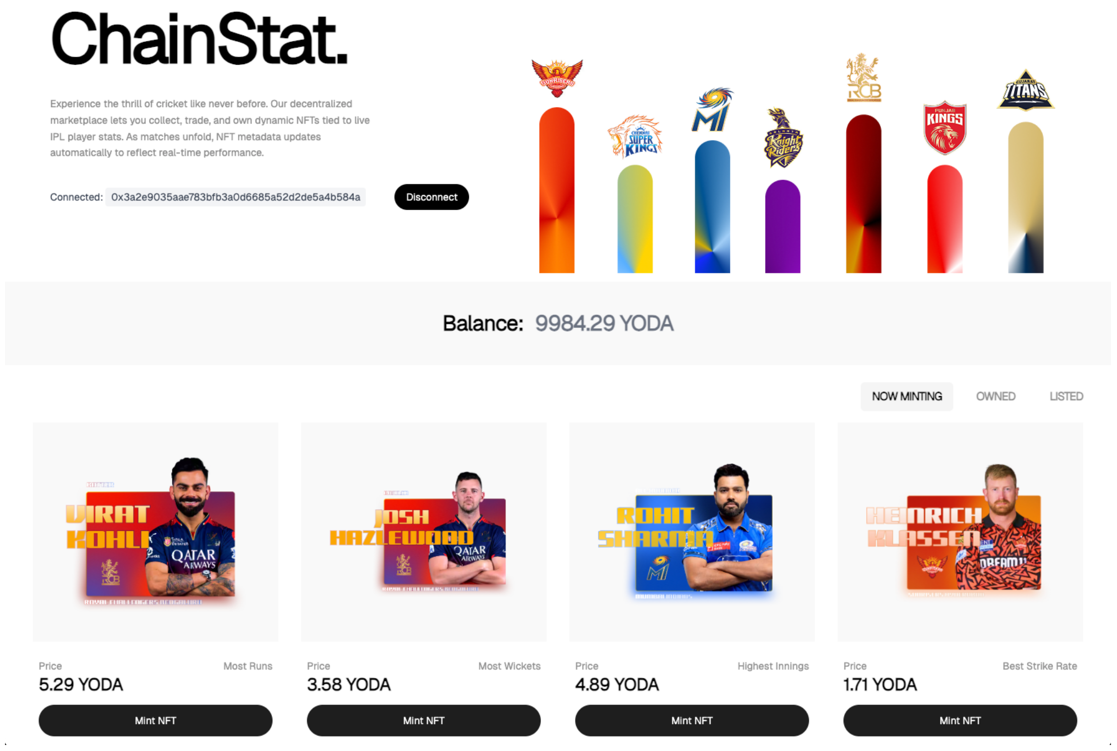
Term Project Report

CHAINSTAT- Decentralized IPL NFT Marketplace

Santosh Kota, UBID-50593968 **Karthik Sharma Madugula, UBID-50611293**

Abstract

ChainStat is a dynamic NFT marketplace built around real-time IPL 2025 cricket statistics. IPL stats—such as most runs, highest strike rate, or most wickets—are tokenized as NFTs using a pool-based architecture. Each stat category corresponds to a pool that generates NFTs with shared, dynamically updated metadata. Users can mint stat NFTs, list them for sale and purchase them using YODA tokens. Metadata updates for all NFTs are automatically triggered every 24 hours based on the latest match statistics—ensuring that if any player surpasses the current leader, the NFT metadata reflects the new top player in that category, showcasing real-time dynamism. Built with React, TailwindCSS, and ethers.js, the platform integrates custom ERC-721and ERC-20 (YODA) smart contracts on the Sepolia testnet using Hardhat.


ChainStat.

Experience the thrill of cricket like never before. Our decentralized marketplace lets you collect, trade, and own dynamic NFTs tied to live IPL player stats. As matches unfold, NFT metadata updates automatically to reflect real-time performance.

Connected: 0x3a2e9035aae783bfb3a0d6685a52d2de5a4b584a

Balance: 9984.29 YODA

Price	Category
5.29 YODA	Most Runs
3.58 YODA	Most Wickets
4.89 YODA	Highest Innings
1.71 YODA	Best Strike Rate

CSE 426LEC Blockchain

Term Project Report

Tech Stack

- **Frontend:** React.js, TypeScript, TailwindCSS, GSAP
- **Smart Contracts:** Solidity (ERC-721, ERC-20, Marketplace contract)
- **Smart Contract Tools:** Hardhat, Ethers.js
- **Storage:** In-memory metadata (CSV source), optional IPFS support
- **Blockchain Network:** Ethereum Sepolia Testnet
- **Wallet Integration:** MetaMask
- **Token:** Custom ERC-20 (YODA)
- **UI Libraries:** Sonner (toasts), Custom modals, GSAP for animations

DApp Workflow

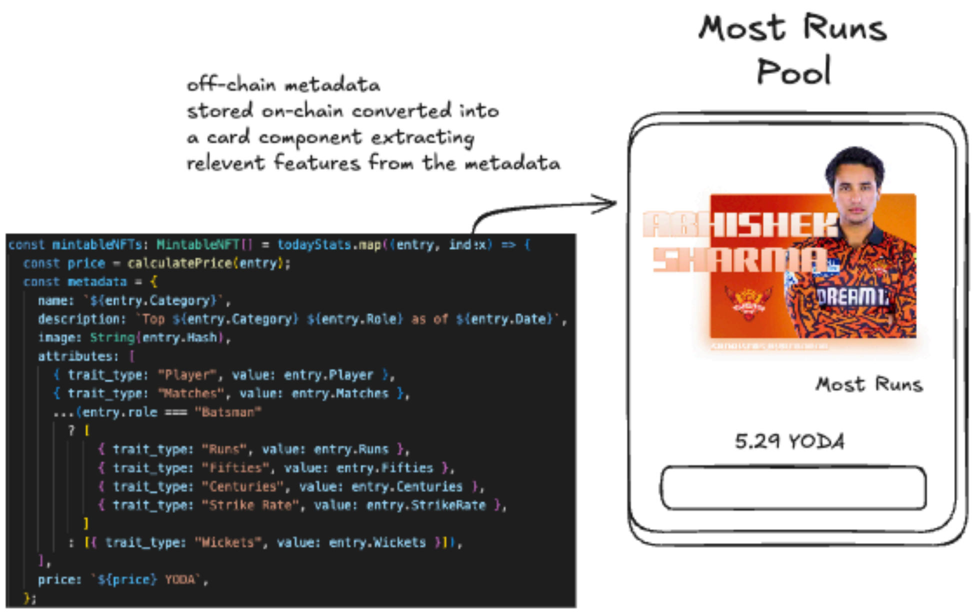
POOL CREATION, DATA GENERATION and ONCHAIN STORAGE

Application starts with parsing of IPL stats and extracts out the current day stat fields and generates metadata for each category based on parsed data which contains fields like- **name, description, role, category, image, runs, wickets** etc, off-chain. Price for each category is also calculated off-chain dynamically based on custom calculations for each category. This generated metadata along with the dynamically calculated price is then converted to JSON string, base-64 encoded and stored on-chain (yes, full on-chain!)

Now admin creates pools for each category where every pool has its own **poolID** and stores metadata like player name, category, price, image IPFS URL and other initial stats. We have stored the metadata on-chain inorder to fully make the application decentralized unlike relying on some off-chain IPFS storage for metadata storage.

CSE 426LEC Blockchain

Term Project Report


off-chain metadata
stored on-chain converted into
a card component extracting
relevent features from the metadata
Most Runs
Pool
Most Runs
5.29 YODA

```
const mintableNFTs: MintableNFT[] = todayStats.map((entry, index) => {
  const price = calculatePrice(entry);
  const metadata = {
    name: `${entry.Category}`,
    description: `Top ${entry.Category} ${entry.Role} as of ${entry.Date}`,
    image: String(entry.Hash),
    attributes: [
      { trait_type: "Player", value: entry.Player },
      { trait_type: "Matches", value: entry.Matches },
      ...(entry.role === "Batsman"
        ? [
            { trait_type: "Runs", value: entry.Runs },
            { trait_type: "Fifties", value: entry.Fifties },
            { trait_type: "Centuries", value: entry.Centuries },
            { trait_type: "Strike Rate", value: entry.StrikeRate },
          ]
        : [{ trait_type: "Wickets", value: entry.Wickets }]),
    ],
    price: `${price} YODA`,
  };
```

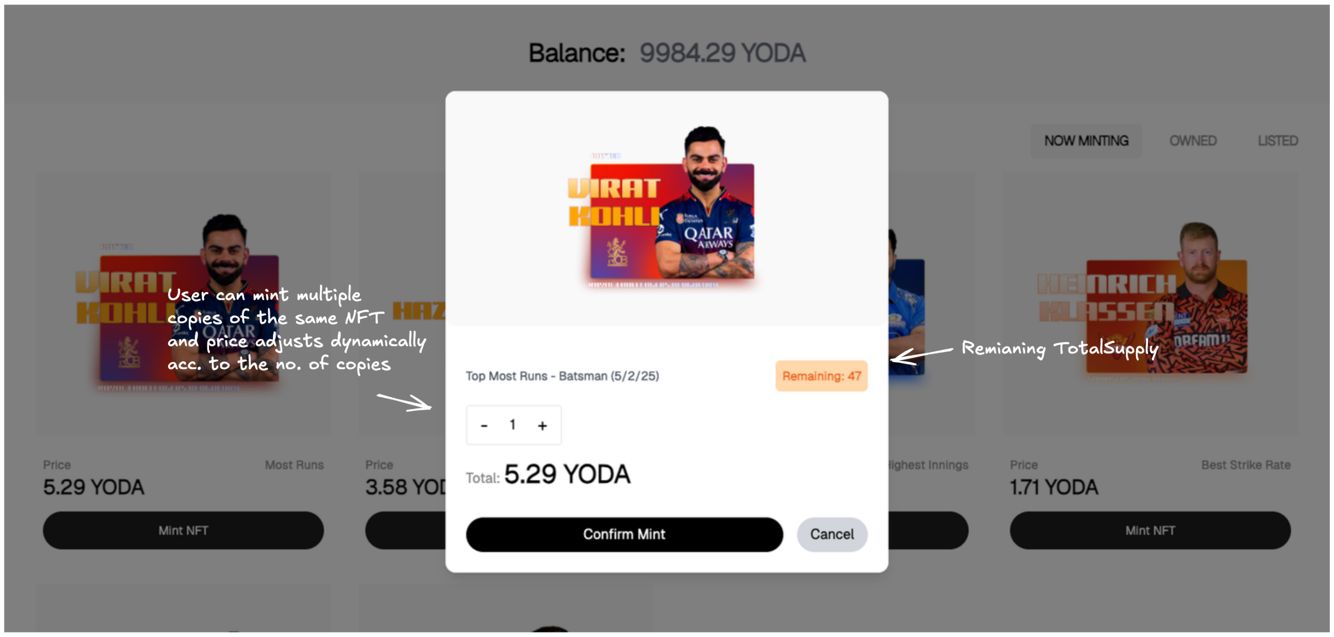
METADATA UPDATES and MINTING

As IPL data changes, the application checks and compares the off-chain generated data with on-chain stored data if there is any change it triggers smart contract to update the metadata. This ensures all the NFTs tied to a pool, sharing same metadata showcase the new updated metadata, enabling true dynamism in NFTs.

Users can mint their selected NFTs from any active pool(which showcases the total supply of that particular category NFT and it is tracked globally, if a user mints an NFT, the **totalSupply** of it gets decreased by 1 and can be seen by everyone using the platform) This minting generates a new tokenID and mints the NFT showcasing latest category information at that particular time to the user. Also, multiple users can mint same NFTs multiple times (until it runs out of supply) but each NFT will be different as it would be having a different tokenID.

CSE 426LEC Blockchain

Term Project Report



Balance: 9984.29 YODA

NOW MINTING
OWNED
LISTED
User can mint multiple copies of the same NFT and price adjusts dynamically acc. to the no. of copies
Remianing TotalSupply

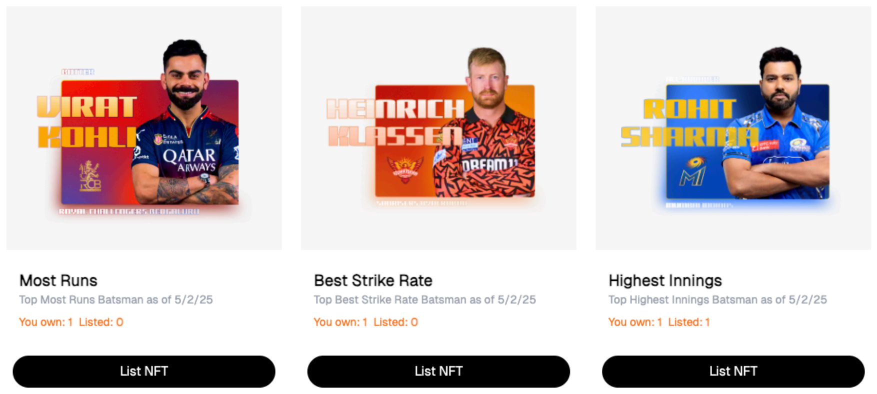
Top Most Runs - Batsman (5/2/25) — Remaining: 47

- 1 +

Total: 5.29 YODA

Price	
5.29 YODA	Most Runs
3.58 YOD	
	Highest Innings
1.71 YODA	Best Strike Rate

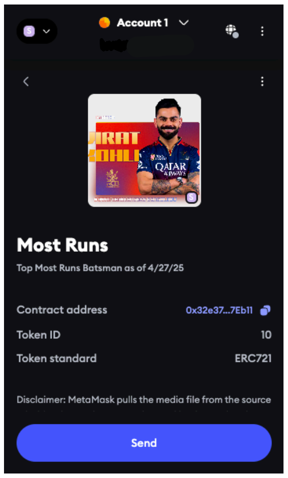
Here we can see how minted NFTs are showcased on Dapp and on metamask- (metadata is fetched from tokenURI and grouped to showcase user how many copies they own.)



Most Runs
Top Most Runs Batsman as of 5/2/25
You own: 1 Listed: 0

Best Strike Rate
Top Best Strike Rate Batsman as of 5/2/25
You own: 1 Listed: 0

Highest Innings
Top Highest Innings Batsman as of 5/2/25
You own: 1 Listed: 1



Account 1 [redacted]

Most Runs
Top Most Runs Batsman as of 4/27/25

Contract address	0x32e37...7Eb11
Token ID	10
Token standard	ERC721

Disclaimer: MetaMask pulls the media file from the source

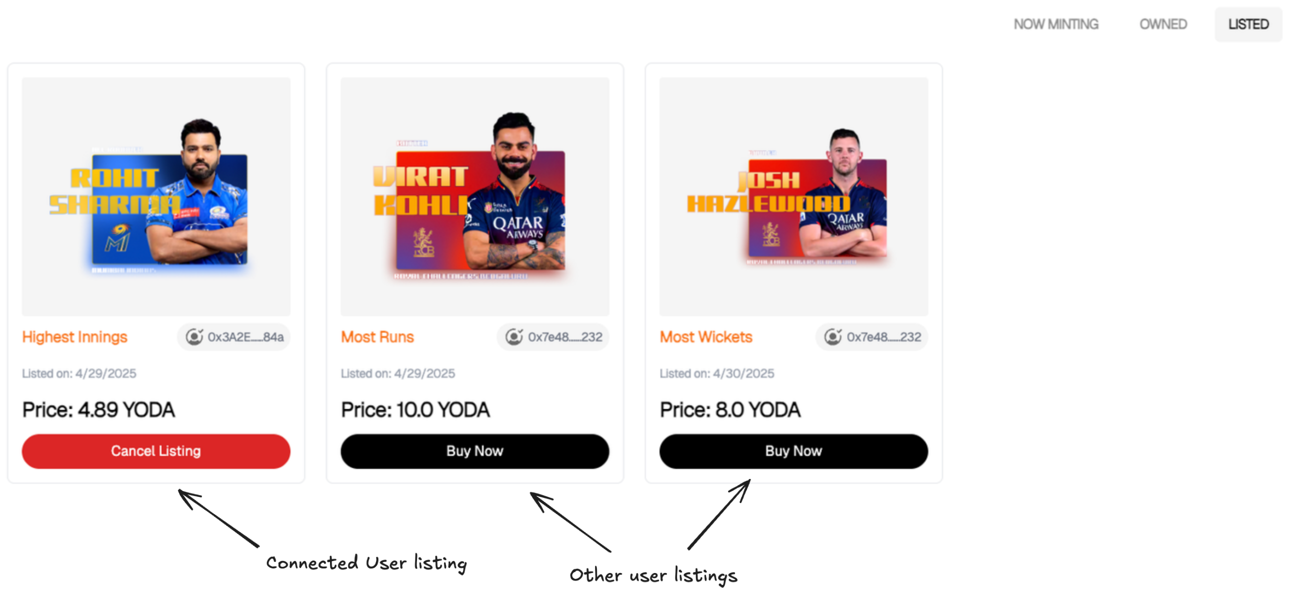
CSE 426LEC Blockchain

Term Project Report

MARKETPLACE

After minting, user has the option to list their NFTs for sale with custom/desired pricing. This uses separate marketplace contract integrating YODA token(ERC-20) transactions. Users have the option to select which copy (i.e. tokenID) they wish to sell. If the user doesn't input a price, it automatically fetches the pool's default price via smart contract interaction, formats it and lists it. This listing/ listed NFTs are stored on-chain via mapping to listedNFTs

We also allow user, the option to cancel their listing which is also an on-chain operation which just invokes **cancellisting(tokenID**) and un-lists it.



Highest Innings 0x3A2E...84a
Listed on: 4/29/2025
Price: 4.89 YODA

Most Runs 0x7e48...232
Listed on: 4/29/2025
Price: 10.0 YODA

Most Wickets 0x7e48...232
Listed on: 4/30/2025
Price: 8.0 YODA

Buyers could see all the active listings which are fetched from the Marketplace contract and lets them buy available NFts for the set price which triggers the buyNFT(tokenID) in the smart contract, transferring the NFT from seller to buyer after buyers pays the seller through YODA(custom ERC-20.

After a successful purchase, the UI state is refreshed instantly removing the purchased NFT from the listed section

So, Metadata for all NFTs is kept fully on-chain, with pool-level updates ensuring efficiency. Off-chain data (CSV stat updates) is only used to trigger and validate metadata updates, not stored externally, thus maintaining decentralization while supporting dynamic performance-based NFTs. Throughout the Dapp, YODA token (a custom ERC-20 token) is used as the currency for minting, listing, and buying NFTs.

Future Work

- We would like to implement Layer-2 chains like polygon for lower gas costs as we are dealing with storing whole metadata on-chain. So gas might be an issue if we have more NFT categories.
- We thought of adding analytics part to the marketplace, but due to time constraints couldn’t go for it but we would like add that in the future.
- If possible, we would implement gasless transactions as well.
- **Immediate action** - Make the application responsive for mobile users!