NFT-Enabled Identity and Access Management at IIT Dharwad

Sree Naga Manikanta Katta - 210010050
Prajwal Biradar - CS21BT065
Kushal Mohta - 210020023

April 13, 2025

1 Introduction

Blockchain technology is seeing an increase in usage and has evolved over the years. Part of the evolution and a significant breakthrough is the ability to perform digital identity management. One such application is the use of **Non-Fungible Tokens (NFTs)**—unique, indivisible tokens that reside on decentralized blockchains. NFTs are particularly well-suited for representing singular assets such as identity credentials, certificates, and access rights due to their uniqueness and verifiability. Keeping this is mind, this project introduces a blockchain-based Identity and Access Management (IAM) system for the academic and administrative ecosystem of IIT Dharwad.

The project adopts a hybrid architectural approach that combines decentralized identity via NFTs with traditional backend infrastructure. Students and faculty are issued unique NFT-based digital identities through a smart contract deployed on an Ethereum-compatible blockchain. These identities, minted during registration from a web-based interface, contain key user metadata and serve as proof of identity across the platform. Off-chain user records and transaction metadata are stored in a database to ensure query efficiency and state synchronization.

This NFT identity enables interaction with various academic modules through the same platform:

- **Digital Identity and Authentication:** NFTs are minted at the time of registration and serve as the user's persistent digital identity, verifiable through smart contracts.
- **Academic Course Registration:** Students register for courses via a web interface, where their NFT identity is validated through smart contract calls before course data is stored in a PostgreSQL table.
- **Course Completion Certificates:** Upon completing a course, the system issues a verifiable certificate as a new NFT, generated and minted via smart contract interactions initiated by an administrator.
- **Semester Fees Payment:** The fee payment module interacts with a payable smart contract to accept payment and issue an NFT receipt.
- **Access to Amenities and Events:** Using the NFT as a digital passport, users can authenticate themselves to book rooms and participate in cultural events through secure contract calls.

By tightly integrating blockchain-backed NFTs with traditional web and database technologies, the system ensures transparency, decentralization, and secure access control while maintaining usability and scalability. This novel approach demonstrates how NFTs can transform academic workflows—from authentication to certification—in a secure and verifiable manner.

2 Approach

Centric to this system is the concept of issuing Non-Fungible Tokens (NFTs) as unique digital identities to each student or faculty member. These NFTs encapsulate essential identity details and **serve**

as verifiable proof of ownership and authentication on a decentralized network.

The registration process begins by means of a web interface built using Node.js, Express, and EJS templates. When a user (a student or a professor) registers, the backend system interacts with a deployed smart contract on the Ethereum blockchain (via Web3.js) to mint a new NFT linked to their identity. Metadata for the NFT—such as name, date of birth, and student details—is either embedded directly ensuring data integrity and decentralization. Alongside this, the user information and blockchain transaction hash are stored off-chain in a PostgreSQL database for efficient querying and cross-referencing. In essence, the priority data is still being stored in the blockchain.

Once a user possesses a valid NFT identity, they can interact with various courses through the platform. The courses available to students are added to the interface by faculty members. Course registration is performed through the UI, with the backend invoking a smart contract function to log the registration on-chain. This guarantees that the student's participation is permanently recorded and tamper-proof. Similarly, course completion certificates are issued in the form of NFTs only after an authorized administrator has given a grade to a particular student. Each certificate NFT contains metadata indicating the course name, issuing authority, and completion date, making it verifiable and shareable in academic and professional circles, thereby ensuring validity and authenticity of coursework and streamlining the process further.

Semester Fee payments are handled through smart contract functions that not only record the transaction but also issue a receipt NFT to the user. This NFT serves as a transparent and immutable proof of payment. Each of these blockchain interactions is mirrored in the PostgreSQL database to maintain a consistent record across on-chain and off-chain components.

In addition to courses, the NFT identity is also used for participating in events and booking rooms or venues through a dedicated amenities system. Smart contract functions validate token ownership before granting access, ensuring that only authorized users can use these amenities.

Overall, this hybrid approach ensures a decentralized, transparent, and secure ecosystem, while also maintaining performance, ease of use, and compatibility with existing institutional infrastructure.

3 System Design and Architecture

The build combines a decentralized smart contract infrastructure with a traditional web-based frontend and backend stack as an interface for students and administrators. The architecture and design is further explained below:

3.1 Digital Identity and Authentication

- **Frontend**: A student or a professor can register or login to the portal by entering details in the `login.ejs`, `register.ejs`, `faculty-login.ejs` and `faculty-register.ejs` pages.
- **Backend**: On user registration, the backend interacts with the `myNFT` smart contract using Web3.js to mint an NFT that represents the user's digital identity.
- **Blockchain**: Function `mintNFT` in the `myNFT` smart contract is called to achieve the required functionality.
- **Database**: The PostGreSQL database stores user details, including their wallet address and their NFT details in the `users` and `faculty` tables.

3.2 Academic Course Registration

- **Frontend**: For students, the registration page allows them to choose courses on their dashboard from a list. For faculty members, the add courses page allows them to add courses to the courses table.

- **Backend**: On course registration, the backend interacts with the `MyCourseReg` smart contract using Web3.js.
- **Blockchain**: The backend invokes a smart contract method (e.g., `registerCourse`) in the `MyCourseReg` smart contract to both verify a student's identity (using their NFT) and register them for a particular course.
- **Database**: The PostgreSQL database maintains a mapping of course registrations and user_ids in the `registrations` table and the details about the courses are stored in the `courses` table.

3.3 Course Completion Certificates

- **Frontend**: The `certificates.ejs` page allows students to avail certificates corresponding to courses if a faculty member has given a grade to them.
- **Backend**: On certificate selection, the backend interacts with the `CertificateNFT` smart contract using Web3.js.
- **Blockchain**: A new NFT representing the course certificate is minted by calling `issueCertificate` in the `CertificateNFT` smart contract.
- **Database**: Each certificate's metadata and transaction hash are stored off-chain in PostgreSQL for easy verification and retrieval in the `certificates` table.

3.4 Semester Fees Payment

- **Frontend**: The fee payment interface allows users to send fees based on the current semester.
- **Backend**: On fee payment for a semester, the backend interacts with the `FeePaymentNFT` smart contract using Web3.js.
- **Blockchain**: A payable smart contract function `payFees` is invoked to complete the transaction and mint an NFT receipt.
- **Database**: All financial records and corresponding transaction hashes are stored in PostgreSQL for audit purposes in the `fees_paid` table.

3.5 Amenities

- **Frontend**: The amenities interface allows users to participate in events and book rooms.
- **Backend**: On event selection or room selection, the nodejs backend interacts with the `AmenitiesNFT` smart contract using Web3.js.
- **Blockchain**: Functions `participateInEvent` and `bookRoom` in the `AmenitiesNFT` smart contract help facilitate this process by verifying the identity of a particular student using their NFT stored on the blockchain.
- **Database**: Tables events, rooms, event_participation and room_booking store the required information

3.6 Project Structure

- `contracts/`: Contains Solidity smart contracts for:
  - `MyNFT.sol` – Minting an NFT
  - `MyCourseReg.sol` – Course Registration
  - `CertificateNFT.sol` – Certificate issuance
  - `FeePaymentNFT.sol` – Fee payment
  - `AmenitiesNFT.sol` - For accessing amenties like events or booking a room

- `build/contracts/`: Contains JSON artifacts generated after compiling the smart contracts using Truffle.
- `migrations/`: Contains scripts to deploy contracts to a blockchain (e.g., Ganache).
- `views/`: Contains EJS templates for:
  - `register.ejs` and `login.ejs` for authentication (Similar .ejs files for faculty as well)
  - `dashboard.ejs` to display registered courses and options
  - `course.ejs`, `certificate.ejs`, `fee.ejs` etc for respective features
  - `error.ejs` for error handling
- `public/`: Contains static files such as stylesheets and client-side JavaScript.
- `index.js`: The main server file that sets up Express, routes, and API endpoints for all blockchain interactions.
- `db.js`: Handles PostgreSQL connections and creates required tables on startup.
- `.env`: Stores environment variables (e.g., DB credentials, port, contract addresses).
- `package.json` and `README.md`: Contain project dependencies, scripts, and documentation.

This modular and scalable approach ensures secure identity verification and efficient access control using NFTs, while maintaining system performance and flexibility through off-chain storage and traditional backend services.

4 Build Screenshots and Proof of Operation

(These screenshots are present in the images directory of the project repository)

5 Conclusion

Overall, the uses for blockchain technology in administrative domains is limitless. A simple solution such as the one detailed in this report boosts the efficiency of all working parties on correct implementation of the concept. More things to be explored in further prototypes could be the involvement of Metamask, and the use of IPFS to store files off-chain. Further 'quality-of-life' updates could involve the addition of a real time log transmitting the transactions occuring within a timeframe for ease of display and communication to users of the chain. Additional database additions such as medical details can also be used to link the programme to healthcare services in a similar manner to coursework, and fee management system designs.

With this the **PROJECT REPORT ENDS HERE**