# CS426 Course Project Report
# One Stop Service - IITDH

B.Dheeraj Reddy - 220010009
P.Tarun Sai - 220010042
B.R.AKash - 220010011

## Abstract

This report presents the design and implementation of a blockchain-based Identity and Access Management system for IIT Dharwad. The system leverages non-fungible tokens (NFTs) to offer secure digital identities, academic course registration, certificate issuance, fee payment receipts, and access to campus services, all over a decentralized and tamper-proof infrastructure.

## 1 Introduction

Non-Fungible Tokens (NFTs) are unique digital assets stored on a blockchain, representing ownership of both digital and physical items. In this project, NFTs are utilized to manage digital identities and control access to resources and services within the campus. The goal is to transition from centralized authentication and record-keeping systems to a decentralized, transparent, and secure NFT-based solution.

## 2 System Design and Architecture

### 2.1 Overview

The system consists of the following components:

- **Smart Contracts:** Deployed using the Truffle framework, written in Solidity. Contracts include `IdcardNFT`, `CertificateNFT`, and `FeepaymentNFT`.
- **Frontend:** Developed using HTML, CSS, and JavaScript with MetaMask integration for wallet-based authentication.
- **Backend:** Node.js with Express, connected to IPFS database for decentralized metadata storage using Pinata.
- **Blockchain:** Local Ethereum environment via Ganache and Etherem Test(sepolia) Networks.

## 2.2 Smart Contracts

Each smart contract corresponds to a different NFT type:

- `Id card NFT`: Minted once per student with identity data by the admin.It is like a virtual Id card for the student which contains a url to the metadata which contains the student's information like name,DOB,Phone number,etc...

- `CertificateNFT`: Minted on course or degree completion by the admin.When a student completes a course or a degree then the admin can send him the certificate of compleion by minting this NFT to the student's wallet address.

- `Fee payment NFT`: Minted upon successful semester fee payment by the student.When the student wants to pay Fee,he/she will enter the respective details and then when they click pay Fee then if they have the respctive etthers in their wallet then the ethers will be transferred to the admin and the transaction will be recorded to the blockchain,The student will then recieve this NFT as the reciept of this transaction.

# 3 Implementation/Architecture

## 1. MetaMask and Ganache Setup/Sepolia test net

- Installed MetaMask and imported an account using private keys from ganache.

- Connected to the local Ethereum network via MetaMask(local).

- Connected to sepolia testnet via truffle dashboard and Metamask (real-world).

## 2. Smart Contract Development

We implemented three primary smart contracts in Solidity (v0.8.19) for identity management, fee payment tracking, and course certification using NFTs. All contracts leverage the OpenZeppelin library for secure and modular development.

### 1. Identity NFT Contract: `Id_card_NFT.sol`

- **Purpose:** Issues a unique NFT to each student or faculty, representing their digital identity at IIT Dharwad.

- **Inherits:**
  - `ERC721URIStorage` – for managing metadata associated with NFTs.
  - `Ownable` – restricts minting functionality to the contract owner.

- **Features:**
  - Uses OpenZeppelin's `Counters` to auto-increment token IDs.
  - Only the admin can mint new identity cards.
  - Emits a custom `Id_card_minted` event for logging mint actions.

- **Security:**
  - Access control is enforced through the `onlyOwner` modifier.
  - Ensures one unique identity NFT per token.

2. **Certificate NFT Contract:** `CertificateNFT.sol`
   - **Purpose:** Mints NFTs as course completion certificates or final degree certificates.
   - **Inherits:**
     - `ERC721URIStorage` – for storing certificate metadata.
     - `Ownable` – restricts minting to administrative users only.
   - **Features:**
     - Token URIs point to IPFS-stored metadata (uploaded via Pinata ).
     - Designed to differentiate between course-level and degree-level certificates.
     - Emits events to track which student received what certificate.
   - **Security:**
     - Ensures certificates are issued only by authorized personnel.
     - Ownership of the token proves certificate validity.

3. **Fee Payment NFT Contract:** `fee_payment_NFT.sol`
   - **Purpose:** Tracks semester fee payments using NFTs and Ether transfers.
   - **Inherits:**
     - `ERC721URIStorage` – stores metadata such as amount paid, semester, and timestamp.
     - **Note:** `Ownable` is **not** used, so that any student can trigger minting.
   - **Features:**
     - The minting function is marked `payable`, allowing ETH payments.
     - Students invoke the mint function to both pay fees and receive a verifiable NFT receipt.
     - Events are emitted to record each fee payment transaction.
   - **Security:**
     - Utilizes `msg.sender` to assign receipt ownership to the payer.
     - Tracks and stores financial proof immutably on-chain and in metadata.

## 3. Truffle Configuration

- Installed Truffle and initialized a project using `truffle init`.
- Configured `truffle-config.js` for local development/Used `truffle dashboard` for real-world test net deployment and Solidity 0.8.19.
- Compiled and deployed contracts using `truffle compile` and `truffle deploy` for local deployment using ganache/using `truffle migrate --network dashboard` for test net deployment.

## 4. Backend Development

The backend of the application was developed using **Node.js** and the **Express.js** framework to handle HTTP requests between the frontend interface, smart contracts deployed on the blockchain, and the IPFS network through Pinata. The backend facilitates critical operations such as minting NFT metadata, managing course registration, handling semester fee receipts, collecting feedback, and accepting work requests.

### 1. Server Initialization and Configuration

- The backend uses `express` for routing, `axios` for making HTTP requests to Pinata's API, `cors` to enable Cross-Origin Resource Sharing between the frontend and backend, and `dotenv` to securely store and load Pinata API credentials from a `.env` file.
- The server listens on port `5000` and uses `express.json()` middleware to parse incoming JSON request bodies.

### 2. IPFS Integration via Pinata

- The backend handles three major NFT-related uploads:
  1. **Identity NFT Metadata:**
     - The route `/upload-to-ipfs` accepts student details (name, date of birth, and phone number).
     - It generates a structured JSON metadata format and uploads it to IPFS using the Pinata API.
     - On successful upload, it returns the resulting IPFS URL which can be set as the token URI during minting.
  2. **Certificate NFT Metadata:**
     - The route `/upload-certificate` accepts a `cert_type` field ("course" or "degree") and an acknowledgment message.
     - Based on the certificate type, distinct metadata structures are created and uploaded to IPFS.
     - The backend returns the corresponding IPFS URL, used to mint the certificate NFT.
  3. **Semester Fee Receipt Metadata:**
     - The route `/upload-semester-fee` accepts student name, semester number, and amount paid.

    - A timestamp is added, and the data is structured as an NFT receipt metadata JSON object.
    - This metadata is uploaded to IPFS and the IPFS URL is returned.

3. **Course Management System**

    - The backend maintains a predefined list of available courses, each identified by a course code and name.

    - **GET /courses** returns the entire course list to the frontend.

    - **POST /register-courses** accepts a student's name, wallet address, and the list of selected courses.

        - Each registration is assigned a unique ID using `uuid.v4()` and stored in an in-memory `registrations` array along with a timestamp.

    - **GET /registered-courses** accepts a wallet address and returns the names of all courses the corresponding student has registered for.

4. **Mess Feedback Collection**

    - The `/submit-feedback` route is used to collect anonymous feedback from students regarding meals (e.g., lunch or dinner).

    - Each feedback entry includes the meal type, the text feedback, and a timestamp, and is stored in an in-memory array for future processing or display.

5. **Work Request Submission**

    - The route `/submit-work-request` accepts general work or maintenance requests from students.

    - These requests are logged to the console and can be extended to trigger notifications or send data to administrators.

6. **Security**

    - API keys for Pinata are never exposed to the frontend. They are securely stored in the `.env` file and loaded using the `dotenv` library.

    - The backend is modular, separating identity management, certificates, course registration, and miscellaneous services into clearly defined endpoints.

    - For simplicity and ease of testing, data like course registrations and feedback are stored in-memory. This can be extended in the future to persistent storage using a database such as MongoDB or PostgreSQL.

## 5. Frontend Integration

### 1. NFT Minting Interface(ADMIN)

The `admin.js` script enables administrators to mint both Identity NFTs and Certificate NFTs through a web interface. It provides two main functions: `mintForUser()` for minting Identity NFTs, and `mintCertificate()` for issuing academic course or degree certificates.

**Identity NFT Minting (mintForUser):**

- Admin inputs the student's name, date of birth, phone number, and recipient wallet address.
- The script uploads this metadata to IPFS via a backend server using Pinata, and receives a `tokenURI`.
- The Identity NFT contract is initialized using `ethers.js` with the admin's signer.
- Only the contract owner (admin) is authorized to mint the NFT, verified via `contract.owner()` and the signer address.
- Upon successful transaction, the `Id_card_minted` event is emitted and the token ID is displayed to confirm success.

**Certificate NFT Minting (mintCertificate):**

- Admin selects the certificate type (course or degree) and provides course name and recipient address.
- A contextual message is generated (e.g., "You have successfully completed [Course Name]").
- Metadata is uploaded to IPFS and the resulting `tokenURI` is used to mint the Certificate NFT.
- As with the Identity NFT, minting is restricted to the contract owner.
- If successful, the `Certificate_minted` event emits the new certificate's token ID, confirming issuance.

Both minting processes ensure that only the designated administrator (contract owner) while the metamask extension is connected to their account can issue NFTs, preserving authenticity and preventing unauthorized issuance. All NFT metadata is securely stored on IPFS, ensuring immutability and transparency.

### 2. Login Authentication via Id card NFT(USER)

The `login.js` script enables secure user login by verifying the presence and ownership of an Identity NFT minted to a student's wallet. Upon visiting the login page, the user connects their MetaMask wallet. The script then interacts with the `Id_card_NFT` smart contract deployed on the blockchain and fetches the list of token IDs owned by the connected address using the `balanceOf()` and `tokenOfOwnerByIndex()` methods.

The system identifies the Identity NFT by checking if the connected address owns a token from the Identity NFT contract. If a valid token is found, the user is authenticated and redirected to the dashboard. If no token is found, an error message is displayed indicating that the user is not registered.

This NFT-based login mechanism provides a decentralized and tamper-proof alternative to traditional username-password authentication, ensuring that only students with a valid, blockchain-minted identity and wallet connected to metamask can access protected services on the platform.

### After Logging in:

### 3. Semester Fee Payment Interface

The `fee_payment.js` script allows students to securely pay their semester fees and receive a blockchain-based NFT receipt for their transaction. This process integrates payment, verification, and digital proof via Ethereum and IPFS.

- The student fills out a form containing their name, semester, and fee amount.
- On submission, the script connects to MetaMask using `ethers.js` and retrieves the student's wallet address.
- The form data is uploaded to IPFS through a backend API endpoint (`/upload-semester-fee`), which returns a `tokenURI`.
- The script initializes the `fee_payment_NFT` smart contract using the student's signer.
- The function `mint_fee_NFT` is called with the user address, `tokenURI`, and the specified Ether value (fee amount) parsed with `ethers.utils.parseEther()`.
- Once the transaction is confirmed on the blockchain, the event emitted by the contract returns the NFT token ID, which is displayed to the user.

**Security and Transparency:**

- Only the user can initiate their own payment and mint the associated NFT.
- The NFT serves as an immutable receipt for the semester fee payment, with metadata (name, semester, amount) stored on IPFS.
- This ensures a transparent and verifiable ledger of all fee transactions, while eliminating manual errors or disputes.

### 4. Certificate Ownership Verification

The `verify.js` script allows users to verify whether a certificate NFT is legitimately owned by their wallet. This is a core component of the system's access control and authenticity check.

- The user enters the Token ID of a certificate NFT and clicks the "Verify" button.
- The script first checks for the presence of MetaMask and prompts the user to connect their wallet.

- Using `ethers.js`, the script retrieves the current connected wallet address via `signer.getAddress()`.
- It then initializes the Certificate NFT contract and calls `ownerOf(tokenId)` to get the current owner of that NFT.
- If the connected user address matches the NFT's owner address (case-insensitive), the certificate is declared as "valid and owned by you."
- Otherwise, it displays that the certificate is not owned by the user.

**Security and Trust:**

- This verification ensures that only the rightful NFT holder can claim ownership.
- It prevents forgery or misuse of certificates by comparing blockchain-verified ownership with the connected Ethereum wallet.
- Since the user must be logged in via MetaMask, the system leverages decentralized identity for authentication.

### 5. Course Registration

This module allows authenticated users to register for academic courses and view their registered courses. It is accessible only after successful login, and the user's identity details (name, wallet address, etc.) are passed via URL parameters.

**Key Functionalities:**

- **Initialization:**
  - When the DOM content is fully loaded, the script extracts URL parameters such as student name, wallet address, date of birth, and phone number.
  - These details are populated into the profile section of the frontend interface.
  - The script sends a request to `/courses` to fetch a list of available courses from the backend.
  - The courses are then dynamically rendered as checkboxes on the page.
- **Course Registration (`registerCourses` function):**
  - Gathers the values of all selected checkboxes corresponding to available courses.
  - If no course is selected, the user is alerted.
  - Otherwise, it sends a POST request to `/register-courses` with the student's name, wallet address, and selected courses.
  - The backend stores the registration, and the frontend alerts the user with the result.

- **Display Registered Courses (`showRegisteredCourses` function):**
  - Sends a GET request to `/registered-courses` with the user's wallet address.
  - Displays the list of previously registered courses below the course selection area.
  - If no courses are found, a message indicates the absence of course registrations.

**Integration with Identity and Access Control:**

- Course registration is only available to users who have successfully logged in and whose wallet matches an identity NFT.
- The wallet address serves as a unique identifier for linking course registrations to the correct student identity.

### 6. Accessing Amenties

After successful login, users can access additional campus amenities through the unified portal, including viewing the mess menu, submitting feedback, and raising maintenance or support requests.

**Mess Menu and Feedback:**

- The weekly mess menu is hardcoded as a JavaScript array of objects and displayed dynamically in a table format upon page load.
- Students can submit feedback for specific meals (breakfast, lunch, or dinner) via a simple form.
- Feedback data is sent to the backend using a `POST` request to the `/submit-feedback` endpoint.

**Work Request Submission:**

- Users can submit maintenance or infrastructure-related requests by filling out a form with details like roll number, issue category, location, and preferred visit schedule.
- On submission, the data is sent to the backend through a `POST` request to the `/submit-work-request` endpoint.
- Once submitted, a confirmation message is shown, and the form is reset for further use.

These features and we can add many more and these extend the use of the NFT-based identity system beyond academics, allowing users to engage with everyday campus utilities in a streamlined, authenticated manner.

## 4. Key Features Demonstrated

This system showcases a seamless integration of blockchain-based digital identity with academic and campus operations. The core functionalities include:

1. **Decentralized Digital Identity:** Every student is issued a unique `IdentityNFT`, serving as their verifiable digital ID. This NFT anchors their presence on the platform and is used for authentication across all services.
2. **Smart Course Registration:** Students can log in using their IdentityNFT and register for available courses through an intuitive interface. Selections are stored in a backend database, creating a tamper-proof academic trail.
3. **On-Chain Certificate Issuance:** Upon successful course or degree completion, students receive a `CertificateNFT` — an immutable, verifiable proof of achievement stored on the blockchain and linked to their wallet.
4. **Transparent Fee Payments:** The system enables students to pay semester fees through the portal. A `FeeReceiptNFT` is minted after payment, ensuring financial records are securely logged and auditable.
5. **Campus Amenities Access and Feedback:** IdentityNFT holders gain authenticated access to various student services like mess menu feedback, event sign-ups, and infrastructure work requests. All interactions are personalized and securely handled post-login.

This holistic approach ensures security, transparency, and user ownership — redefining how students interact with institutional systems in a Web3-enabled campus.

## 4 Challenges Faced

Building a blockchain-based academic and amenties management system from the ground up came with several technical and conceptual challenges:

- **Wallet Integration:** Ensuring seamless interaction between the frontend and MetaMask was challenging, especially when managing user authentication and transaction signing across different actions like course registration and fee payment.
- **IPFS and Metadata Handling:** Managing off-chain data (like student names, course details, and payment metadata) using IPFS introduced complexity in terms of structuring JSON data, handling pinning services like Pinata, and ensuring reliability of links.
- **Frontend-Backend Coordination:** Synchronizing data flow between the decentralized blockchain and centralized backend (e.g., for course lists and user-specific data) demanded a hybrid architecture that balanced decentralization with practical usability.
- **User Experience:** Designing an intuitive and error-resistant interface for non-technical users while handling asynchronous blockchain transactions, confirmations, and failures was a constant challenge.

## 5 Conclusion

This project demonstrates the potential of NFTs beyond digital art — showcasing their power in transforming academic identity, certification, and resource access. By leveraging blockchain technology, the system ensures that student records and transactions are secure, verifiable, and tamper-resistant.

What sets this solution apart is its decentralization and automation: identity verification, course registration, certificate issuance, and fee payment are all managed seamlessly through smart contracts — eliminating the need for manual validation or centralized oversight.

Importantly, this is just the **first version** of the platform. The architecture has been designed with extensibility in mind, paving the way for future enhancements such as campus-wide voting, attendance tracking, and even NFT-based access to physical spaces.

With further development and institutional collaboration, this system holds the promise to be deployed in real-world academic environments — laying the foundation for a transparent, secure, and student-empowered digital campus.

## References

- https://trufflesuite.com
- https://pinata.cloud
- https://www.web3.university/tracks/build-your-first-nft/how-to-create-an-nft
- https://www.web3.university/tracks/build-your-first-nft/building-a-full-stack-nf