PROJECT REPORT

On

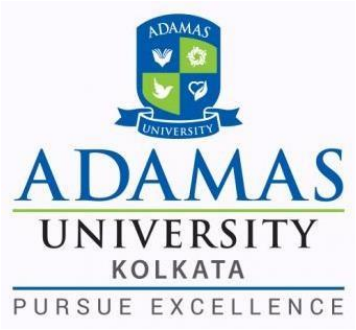
"Student Verification and Management System"

Submitted in partial fulfilment of the requirements for the award of

Bachelor of Computer Applications (BCA)

In the department of

Computer Science and Engineering



Submitted by:
Tamal Sur (UG/02/BCA/2022/041)

Manish Mandal (UG/02/BCA/2022/017)

Nadia Chand Mondal (UG/02/BCA/2022/031)

Under the Guidance of

Dr. Sudipta Das
(Assistant Professor)
Department of CSE

School of Engineering & Technology
ADAMAS University, Kolkata, West Bengal
July 2024 – December 2024

CERTIFICATE

This is to certify that the design report entitled " Student Verification and Management System ", submitted to the School of Engineering & Technology(SOET), ADAMAS UNIVERSITY, KOLKATA in partial fulfilment for the completion of Semester – 5th of the degree of Bachelorette of Computer operations in the department of Computer Science and Engineering, is a record of Bonafide work carried out by Tamal sur, UG/ 02/ BCA/ 2022/041, Manish Mandal, UG/ 02/ BCA/ 2022/017, Nadia Chand Mondal, UG/ 02/ BCA/ 2022/031 under the guidance of Dr. Sudipta Das..

All help entered by us from colorful sources have been properly conceded.
No part of this report has been submitted away for award of any other degree.

Dr. Sudipta Das
(Assistant Professor)

Mr. Sayantan Singha Roy /
Mr. Aninda Kundu

(Project Coordinator)

Dr. Sajal Saha
(HOD CSE)

ACKNOWLEDGEMENT

We'd like to express our sincere gratefulness to Dr. Sudipta Das, our recognized companion, for furnishing us with inestimable guidance, support, and stimulant throughout the development of this design, Student Verification and Management System. Das's perceptivity and moxie have been necessary in helping us understand the complications of blockchain technology and its operation in creating an effective and secure system. Without their nonstop support and formative feedback, this design would not have been possible. We're also thankful to our institution, faculty members, and peers for their constant stimulant and cooperation during this trip. Incipiently, we extend our appreciation to everyone who directly or laterally contributed to the successful completion of this design.

Thank you.

DECLARATION

We, the undersigned, declare that the project entitled "Student Verification and Management System", being submitted in partial fulfillment for the award of Bachelor of Computer Applications Degree in Computer Science and Engineering, affiliated to ADAMAS University, is the work carried out by us..

Tamal Sur

(UG/02/BCA/2022/041)

Manish Mandal

(UG/02/BCA/2022/017)

Nadia Chand Mondal

(UG/02/BCA/2022/031)

ABSTRACT

This system on Blockchain-based Student Verification and Management System is specially designed to overcome the existing issues related to student data management and verification, where tampering of data can happen, delay in services and also nine inefficiencies prevail. Blockchain technology will safeguard all records and certificates on account while availing smart contracts and decentralized storages. This way grade updation and certificate distribution happens without fraud.

and delays. Instant records access by authorized users-schools and employers, promotes transparency, trust, and efficiency. It is an efficient and innovative system for the management of student data.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Background

This blockchain technology of this student verification and credential management system will assure tamper-proofing while showing a transparent way to authenticate credentials stored in the academic framework with safety. Utilizing such a combination of novel technologies such as blockchain, IPFS, and MongoDB, within no time, students and authenticators will receive credentials precisely by the institution, keeping readability and maintainability easier with regard to code in the application of Node.js.

1.2 Purpose of the Project

It deals with the development of a blockchain-based paradigm that would decrease the inefficiencies and risks associated with credential management security. Most common manual or central operations result in qualification lost or tampered with, but blockchain technology can easily ensure dispersal, immutability, and verifiability.

1.3 Problem Statement

These systems have various disadvantages, such as vulnerability to data, inefficiency, and lack of trust from the stakeholders. Usually, academic records are kept in a central database; hence it is at the risk of breaches, unauthorized modifications, and loss. Verification processes tend to be time-consuming; most because they are conducted on paper and by doing manual checks.

This ensures that the institutions have a safe and reliable credential management system that can help build trust and integrity in the credential management process in the digital world..

1.4 Objective

Security: Ensure that the sensitive credential information is not tamper-proof and non-changeable. Transparency: It offers verifiable report for students, verifiers, and institutions. Efficiency: It automates issuances and verifications on credentials without delay Access: It makes the credentials accessible to the authorized stakeholders by using blockchain and IPFS. Scalability: It supports mass deployment for institutions with significant numbers of students..

1.5 Structure of Project

Outline of this project:

Chapter 2: Literature Review of past work done in the fields of blockchain and credential management.

Chapter 3: Technologies Used that will contain the blockchain platform, IPFS, MongoDB, among others used in the building process.

Chapter 4: Methodology, to be the comprehensive work flow and design of the proposed system.

Chapter 5: Hardware and Software Requirements, to be a list of tools and frameworks used in the project

Chapter 6: Implementation and Results, to be used in the presentation of the working prototype and evaluation metrics.

Chapter 7: Conclusion and Recommendations, putting forward grounds for improvement and scalability..

CHAPTER 2
LITERATURE REVIEW

2.1 Literature review of some of the previous reports

The Student Verification and Management System is proposed to provide a secure, efficient, and transparent way of storing and verifying student records. Previous studies on student data management have been conducted differently to address problems in traditional systems. Several studies have proposed using technology, especially blockchain, to enhance data security and verification.

Zian Alfaen 2022 designed a system in which students can access schedules and exam details. Such a system is aimed towards creating an effective tool for maintaining student records. Ave Prasojo and Mira Kania, 2022, used Goal-Directed Design (GDD) and ensured that systems meet the preferences of users by conducting interviews with users.

Jason Beaird (2007) did research on web design principles with regard to user-friendly and aesthetic interface approaches. This concept is applied in our work to ensure that our interface will be clean and easy to navigate. Previous studies have focused on blockchain technology because of its decentralized and tamper-proof nature can be suitable for the safe storage of data such as student records which cannot be altered. That is, blockchain guarantees transparence in student records without being changed.

Many other researchers have researched how smart contracts have automated certificate issuance or grade updating. Therefore, the intervention of human beings is less important, and efficiency is higher in such cases. Guided by these ideas, the Student Verification and Management System were designed and developed to make them secure and efficient and fulfill the needs of the end-users..

CHAPTER 3
TECHNOLOGY USE

3.1 Introduction

This project is designing an efficient, safe, and decentralized system of verification for student credentials with the aid of up-to-date technologies. It creates an interactive and current look and feel using React and Tailwind CSS for the frontend. For the backend, it has API, database, and authentication handling using Node.js and Express.js and MongoDB. It also develops Blockchain technologies with Solidity and Smart contract creation. Decentralizes the storage of files by Pinata in an IPFS too. These technologies come with scalability, security, and user-friendly feature.

3.2 Frontend Technologies

React is the JavaScript library for building a dynamic UI. It allows modular, reusable, and maintainable code through component-based development. In this project, dashboards of users including dynamic forms for students, teachers, and administrators are running on React. Tailwind CSS is a utility-first CSS framework that makes styling easier, using predefined classes in the actual HTML structure, which produces quicker prototyping and keeps responsiveness on all devices so that there is a clean and modern user experience on styled buttons, forms, and layouts. JavaScript: It brings interactivity and functionality on the frontend. It is used for manipulating user input, managing state on React, and for creating components like modals and form validation.

3.2 Backend Technologies:

Node.js: It is a JavaScript runtime built for this project's back end. The architecture being non-blocking and event driven, it's very efficient about handling multiple requests on the API.

Express.js is the name of the lightweight framework in order to develop web applications and APIs on Node.js. Express.js reduces complexity in routing, middleware integration, and error handling. This project applies the use of Express.js for API endpoints that have something to do with user authentication, issuance of credentials, and verification.

MongoDB: It is the NoSQL database, or in other words, the information of user and credential will store in flexible JSON-like format. For all the above-reasoned points, for the dynamic changing data models MongoDB would be the best. Information regarding the students, teacher, credential, and even IPFS hash is stored inside.

mongoose: mongoose is ODM. Library using MongoDB. It enables defining schemas in the database, so

validation, middleware, and easy queries are possible. Below is an example where Mongoose is used to define models for data about students, teachers, and credentials to be more effective in working with MongoDB.

3.4 Blockchain Technologies

Solidity: It is a high-level programming language mainly used for writing and deploying smart contracts on the Ethereum blockchain. This project uses issuance, validation, and revocation of credentials. The smart contract provides transparency, immutability, and security due to the storage of credential hashes in the blockchain. Pinata (IPFS): Pinata is interface, which makes the process of storing files via IPFS easy. In this project, student credentials and documents are uploaded to the IPFS through the Pinata service. Unique CIDs are fetched from it, which further gets stored in MongoDB for future reference within blockchain for decentralized and tamper-proof file access.

3.5 Supporting Tools and Utilities

Postman: This is an app that helps aid in API test and debug phase during developing it and checks the authentication part of users and issuing credentials, issuing and authentication point of APIs are consistent.

dotenv : dotenv allows the handling environment variables, allowing sensitive information - API key for Node.js like database connection strings and other JWT related key will be kept with security as the same source. JWT (JSON Web Token): It is used to safely authenticate users. The tokens are generated at the time of login, after which they can access certain routes and resources only if proper authorization is done.

bcrypt: bcrypt is a library for password hashing. It ensures that the password is encrypted before it hits the database. In this way, even if the database gets compromised, there will be no unauthorized access.

3.6 Summary

The system hence has a strong and secure verification for the student's credentials amalgamated through React, Tailwind CSS, Node.js, MongoDB, Solidity, and IPFS technologies hence becomes scalable and usable while making sure that confidence is instilled in the users in all the systems applied.

CHAPTER 4
METHODOLOGY

4.1 Database Description:

Database Description: This student verification and management database uses blockchain technology for safe information storage of all student's personal details, grades, and certificates in an inviolable manner. It organizes data into "blocks" arranged chronologically; hence, any alterations will prove to be quite simple to identify because data is ensured in integrity. Each block holds a unique identifier, timestamp, and hash value to guarantee that all records can be traced and verified. Cryptographic methods ensure data protection. Only the authorized individuals who include school administrators, employers, and students are allowed to access or update records. Mechanisms of access control will ensure that only proper permission holders will modify the sensitive data. This is due to the reason that such smart contracts automatically update grades and issue certificates and thus remove human intervention, with consistency in the process, making it safe, transparent, and easy to audit with risks being seen in the traditional centralized systems. It is a scalable and efficient system that can rely on managing large and small student records within the institution. Furthermore, the system updates the records in real-time, which means that delays associated with verification are eliminated and the records are always updated. The system further decentralizes data storage and hence reduces the risk of breaches or unauthorized access.

4.2 Primary Key:

The Student Verification and Management System use the primary key as an identification of the record of a student uniquely. It is ensured that every record will be accessed, updated, or verified without confusion. It will help maintain the integrity and uniqueness of every student's data in the blockchain..

4.3 Foreign Key:

A foreign key in the record of students links related information such as courses of departments. A Foreign key will ensure that records remain consistent and that records are always in a relationship.

4.4 Process Flow Diagram:

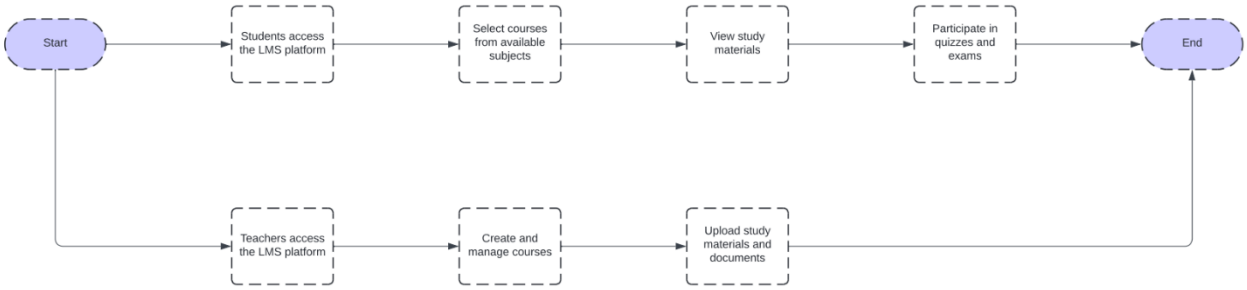


Figure 4.4.1: Process Flow Diagram

Explanation:

The diagram shows the LMS workflow: students access the platform, select courses, view materials, and complete quizzes or exam. Teacher access the platform to create and manage courses, upload materials, and support learning. The system steamlines academic peocesses for both students and teacher, ensuring efficient course management. It provides a centralized platform for learning and teching activities, enhancing overall productivity.

4.5 ER Diagram:

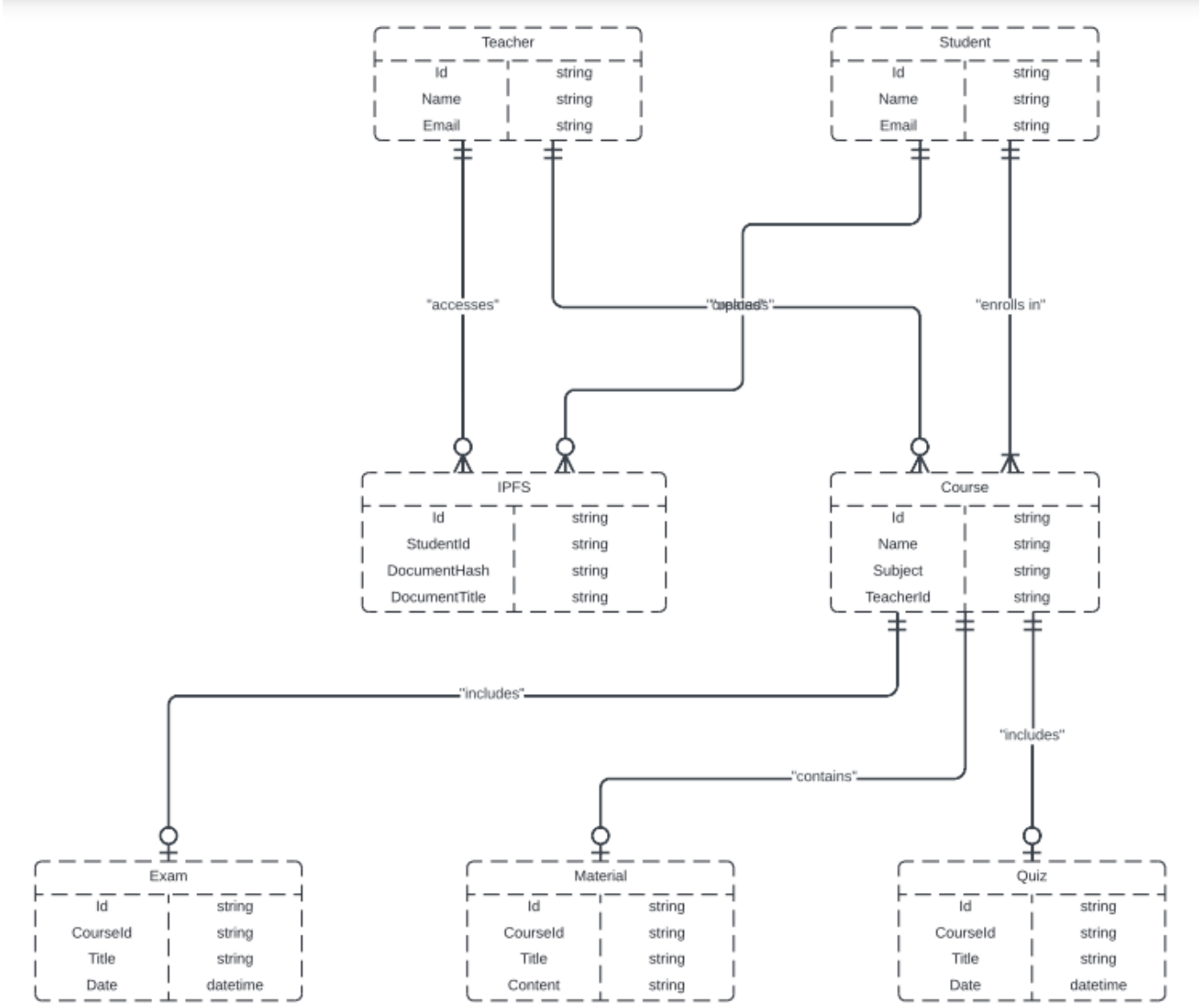


Figure 4.5.1: ER Diagram

Explanation:

The full form of ER diagram is Entity Relationship Diagram. ER diagram represents relation between data tables. It consists of some symbols. ER diagram shows how many relation a table has with another table.

4.6 Use Case Diagram:

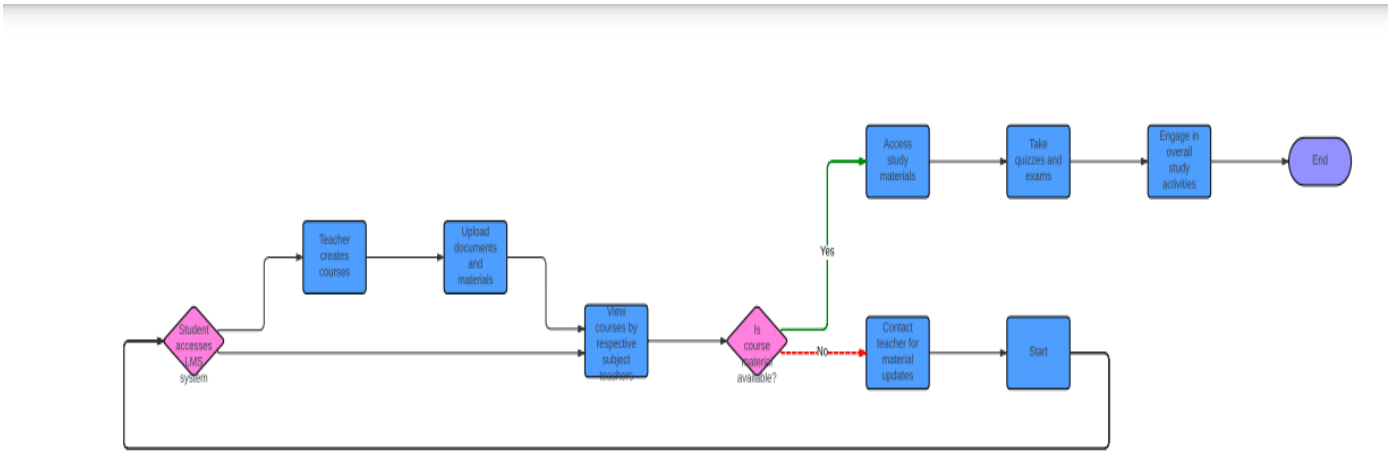


Figure 4.6.1: Use Case Diagram

Explanation:

Use Case Diagram is used to represent interaction between user and system. It describes how a system is designed to operate for user's needs. It helps to determine system behaviour. Use Case Diagram in this project shows interaction between user and the system. According to the diagram student can register, and lodge complaint. Disciplinary committee is able to receive complaint, check report, and take decision.

4.7 DFD Diagram:

Explaination:

The DFD explains the interaction between the students, teachers, and Learning Management System. The LMS is a central platform where all activities of the students and teachers are performed. All students interect with the LMS in order to enroll for courses, and monitor there academic process. The LMS allows teachers to create and manage courses, upload learning resources , and assess the performrnce of there students. It bridges communication and workflow between students and teachers in an efficient and organized way.

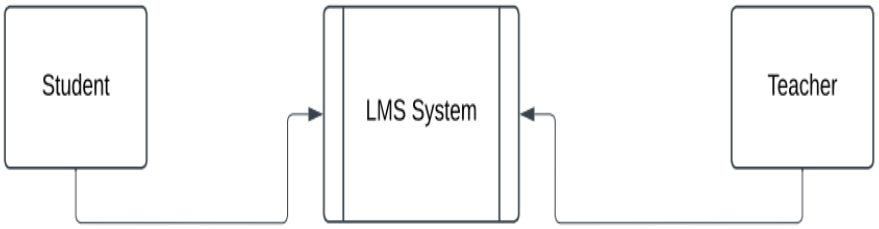


Figure 4.7.1: Level 0 DFD

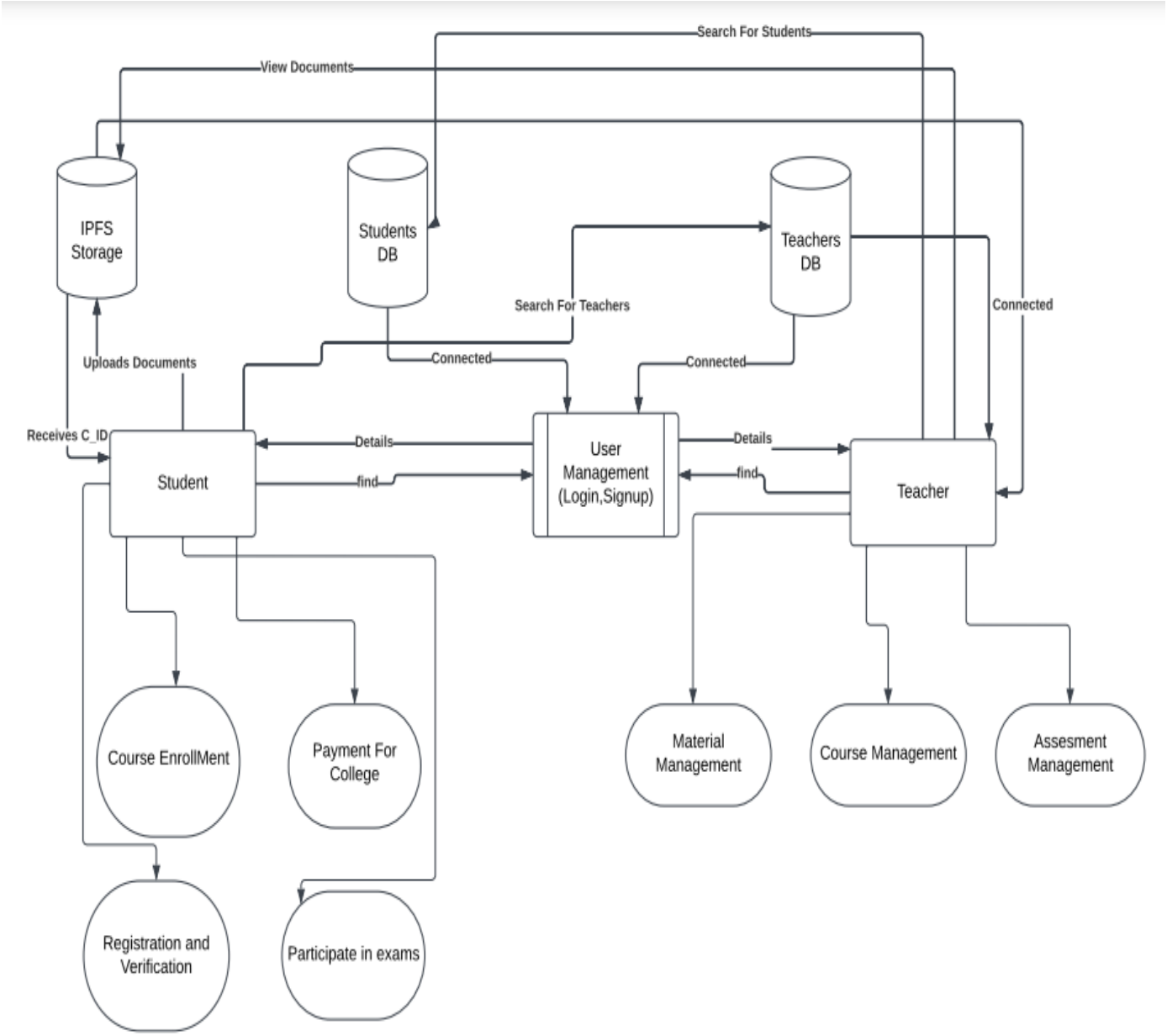


Figure 4.7.1: Level 1 DFD

CHAPTER 5
OUTPUT

Figure 5.1: Home Page

Home Page: This is Student verification and Management System Home Page where student and admin can login and learn about awareness.

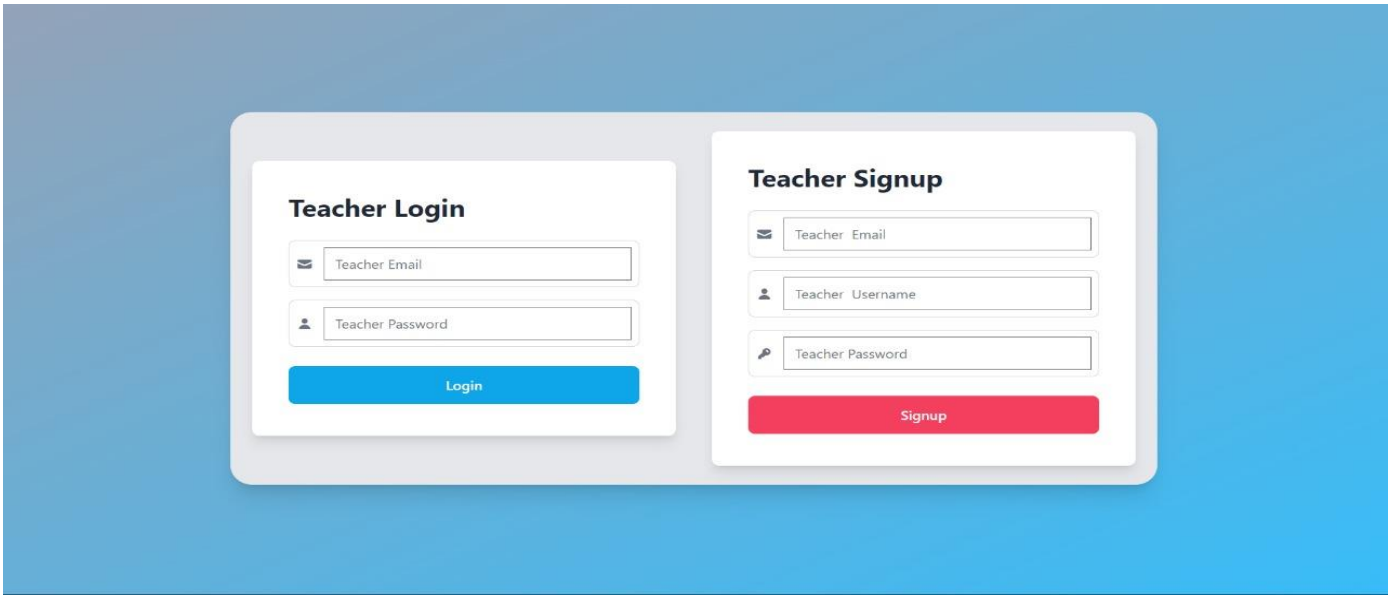


Figure 5.2: Sign up and Login Page

Sign up and Login Page: Teacher can Sign up and Login using this page.

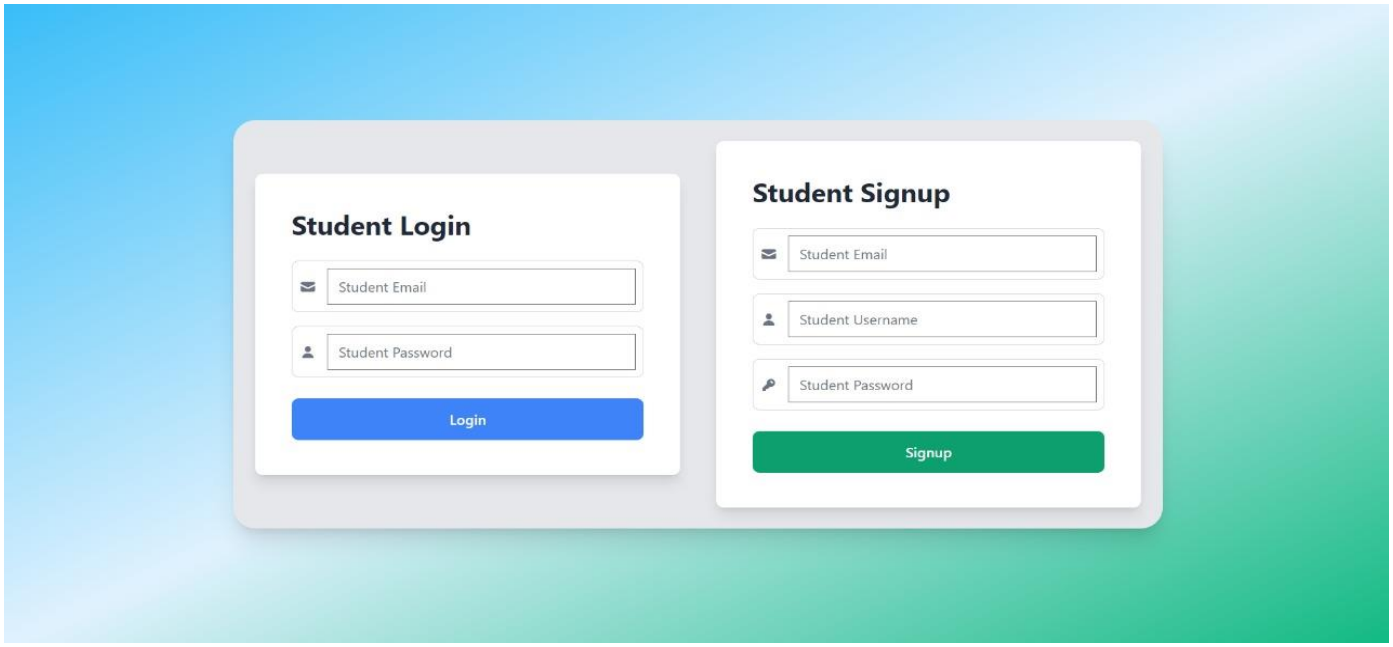


Figure 5.3: Sign up and Login Page

Sign up and Login Page: Student can Sign up and Login this page.

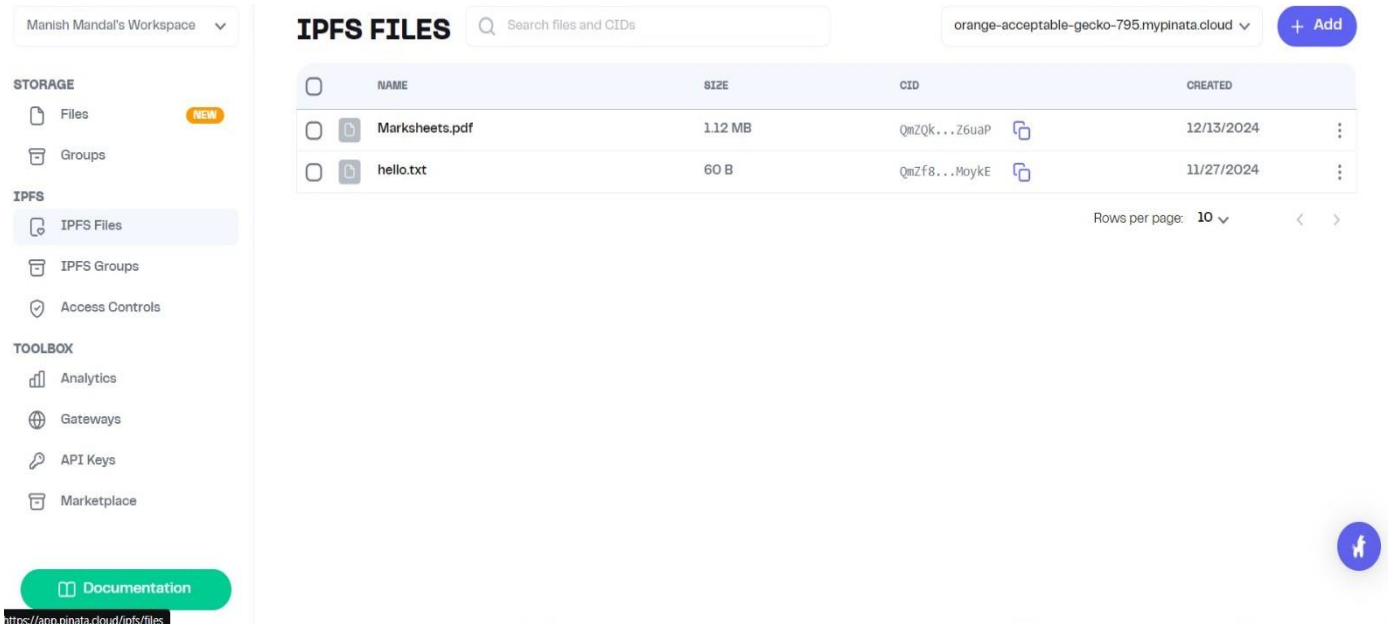


Figure 5.4: IPFS Page

Report Page: Students Credentials are stored in IPFS.

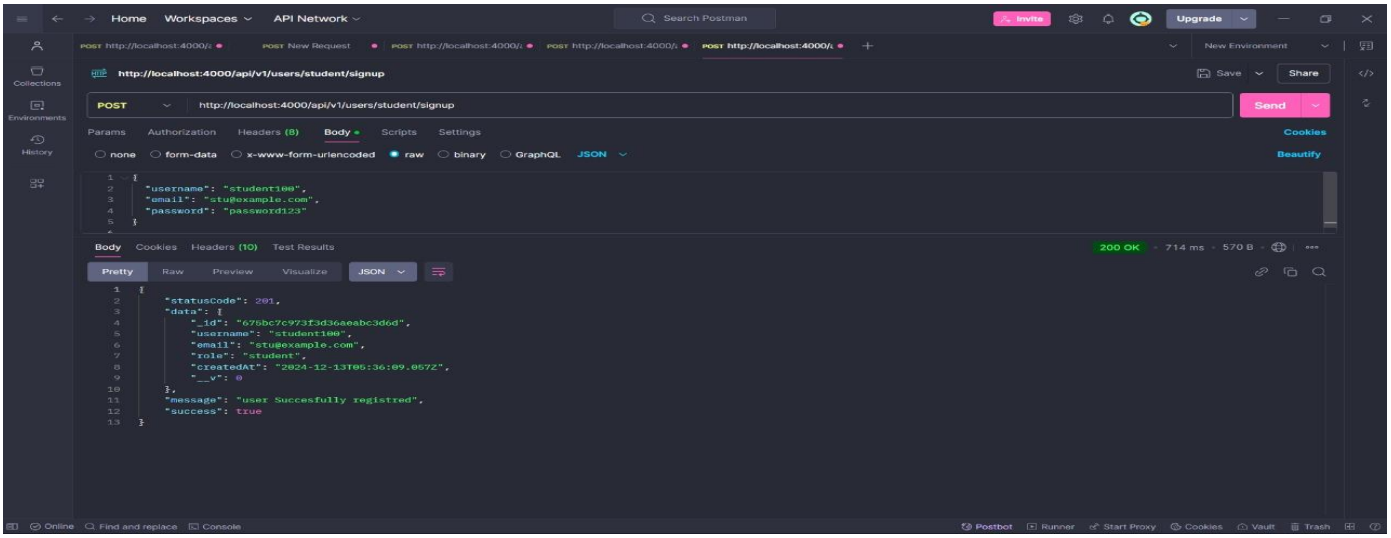


Figure 5.5: Postman Request(Student Sign Up)

Student Details Page: Backend response for student signup using postman API.

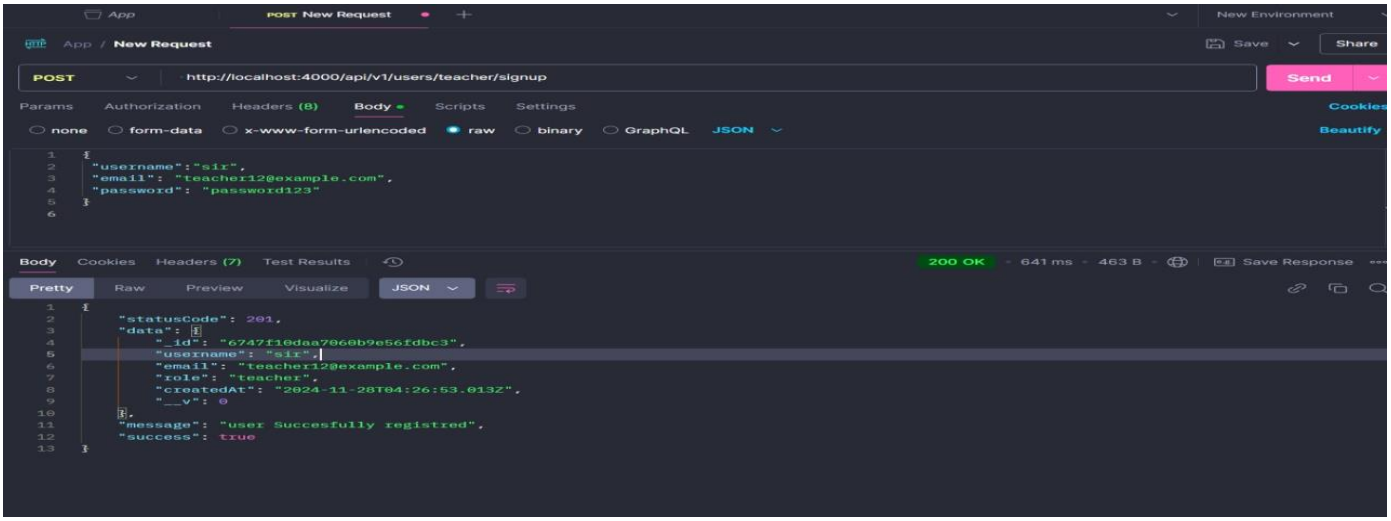


Figure 5.6: Postman Request(Teacher Sign Up)

Report Details Page: Backend response for Teacher signup using postman API.

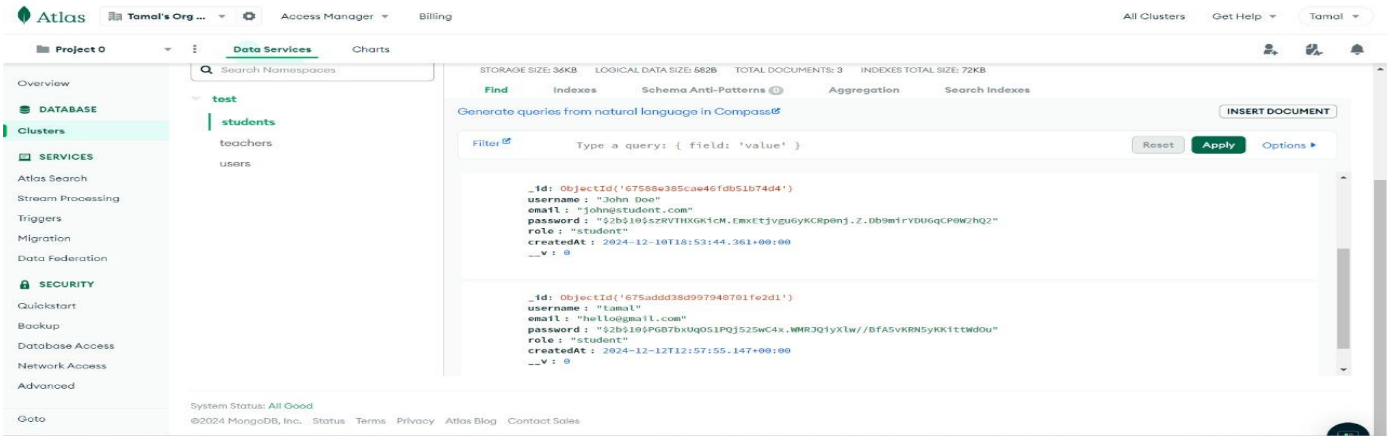


Figure 5.7: Students Database Page

Students Database: Student database will store students' data securely. Every student have unique email id.

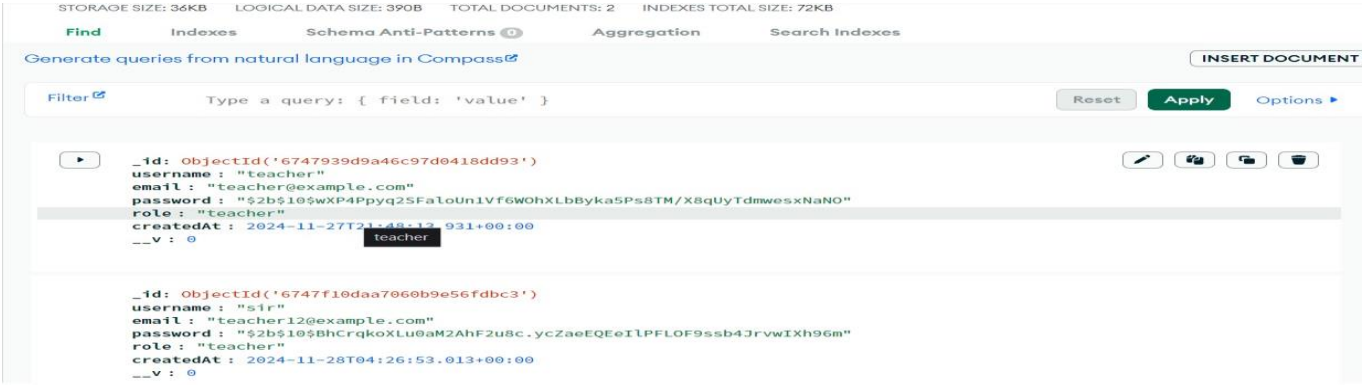


Figure 5.8: Teacher Database Page

Admin Database: Teacher database will store Teacher data securely. Every teacher will have unique Email ID.

CONCLUSION

Conclusion Therefore, the Student Verification and Management System is an even more robust solution meant for streamlining the management and verification of student records. Making use of blockchain technology for security, transparency, and immutability in accessing records that are academic, this offers smooth as well as reliable ways that academic institutions, employers, or any student can access to obtain verification of academic records or records without manual interventions in the process.

It also integrates smart contracts, thus allowing for automation of the processes involved and reducing the chances of error. Also, the interface is user-friendly, thus accessible to all the stakeholders. This makes the system practical and effective in real-world applications. This system also has scalability, which enables institutions to grow without having to compromise on performance or security.

In a nutshell, the project solves the core problems of data integrity, security, and verification, thereby providing a modern solution to outdated systems. This technology will help educational institutions streamline their processes, build trust, and enhance the experience for students, faculty, and employers.

.

FUTURE WORK

The Project already has many features but can add more in the future. For example, Course Management: Create, edit, and manage courses, including syllabus, assignments, and schedules.

Student Dashboard-Included within this dashboard, are courses enrolled, grades received, announcements, and deadlines.

Role-Based Access: Let students, instructors, admins, and all other roles have different permission capabilities.

Gradebook: Manages and tracks grades, attendance, and even performance analytics.

Assignment and Quiz Management Upload assignments, schedule quizzes, automatic grading for objective questions.

Payment Gateway: Enable students to make payments for course, certificate, or other services fees easily..

REFERENCE

[1] Nakamoto, S. (2008). "Bitcoin: A Peer-to-Peer Electronic Cash System." This was a whitepaper that invented blockchain and bought in the concept of decentralization.

[2] Eth ethereum Foundation. 2014. "Ethereum Whitepaper: A Secure Decentralized Generalized Transaction Ledger." Full reference guide to Ethereum smart contracts and decentralized applications.

[3] OpenZeppelin. 2023. "Secure Smart Contract Libraries." Official reference document on reusable components of the blockchain that are used in building safe, efficient applications..

[4] Protocol Labs. 2023. "IPFS Documentation." Detailed implementation information to get started with implementing IPFS for decentralized storage.

[5] MongoDB Inc. 2023. "MongoDB Documentation." Official database management resource that is flexible in schema with scalability.

[6] ReactJS. 2023. "React Documentation: A JavaScript Library for Building User Interfaces." Complete guide on building reusable, dynamic user interfaces with React.

[7] Truffle Suite. 2021. "Truffle Documentation: A Development Framework for Ethereum." Pragmatic documentation about developing and testing your smart contracts on Ethereum.

[8] Pinata. 2023. "Pinata Documentation: Simplified IPFS File Management." Easy how-to use the API of Pinata on IPFS to manage files.

ANNEXTURE

IPFS Testing :

```
import axios from 'axios';
import dotenv from 'dotenv';
import FormData from 'form-data';
import fs from 'fs';


dotenv.config();

const PINATA_API_KEY = process.env.PINATA_API_KEY;
const PINATA_API_SECRET = process.env.PINATA_API_SECRET;


const uploadToPinata = async (filePath) => {
  const url = 'https://api.pinata.cloud/pinning/pinFileToIPFS';

  try {
    const data = new FormData();
    data.append('file', fs.createReadStream(filePath));

    const response = await axios.post(url, data, {
      headers: {
        ...data.getHeaders(),
        pinata_api_key: PINATA_API_KEY,
        pinata_secret_api_key: PINATA_API_SECRET,
      },
    });

    console.log('File uploaded successfully! CID:', response.data.IpfsHash);
    return response.data.IpfsHash;
  } catch (error) {
    console.error('Error uploading to Pinata:', error.response?.data || error.message);
  }
};


const getFromIPFS = async (cid) => {
  try {
    const url = https://gateway.pinata.cloud/ipfs/${cid};
    console.log('Retrieving from URL:', url);
```

```
    const response = await axios.get(url);
    console.log('Retrieved Content:', response.data);
  } catch (error) {
    console.error('Error retrieving from IPFS:', error.message);
  }
};

const testPinata = async () => {
  const filePath = './Marksheets.pdf';
  const cid = await uploadToPinata(filePath);

  if (cid) {
    await getFromIPFS(cid);
  }
};

testPinata();
```

```
  }
?>
```

Explanation:

This is an upload and retrieval of files into and out of IPFS using Pinata. Auth with Pinata happens by using API keys within an.env file to authenticate, which uploads the selected file Marksheets.pdf through Pinata's IPFS service to return the file's CID. A subsequent call utilizes this CID to retrieve the content of the file, this time using the public gateway at Pinata. End-to-end tests of the workflow upload the file if successful and retrieve with CID logging to the console for safe, decentralized storage and retrieval of files.

Navigation Bar :

```
import React, { useState } from "react";
import { useNavigate } from "react-router-dom";
import Features from "../Features";
import HowItWorks from "../HowItWorks";
import Testimonials from "../Testimonials";
import Footer from "../Footer";
import { motion } from "framer-motion";

const Navbar = () => {
  const [isModalOpen, setIsModalOpen] = useState(false);
  const [actionType, setActionType] = useState(""); // To distinguish between 'Login' or 'Sign Up'
  const navigate = useNavigate();

  // Open modal with specified action type (Login or Sign Up)
  const openModal = (action) => {
    setActionType(action);
    setIsModalOpen(true);
  };

  // Close the modal
  const closeModal = () => {
    setIsModalOpen(false);
  };

  // Handle the navigation based on role (Student or Teacher)
  const handleNavigation = (role) => {
    // Navigate to the appropriate page based on role and action type
    if (role === "Student") {
      if (actionType === "Login") {
```

```
      navigate("/api/studentLoginSignup"); // Navigate to Student Login/Signup
    } else if (actionType === "Sign Up") {
      navigate("/api/studentLoginSignup"); // Navigate to Student Login/Signup
    }
  } else if (role === "Teacher") {
    if (actionType === "Login") {
      navigate("/api/teacherLoginSignup"); // Navigate to Teacher Login/Signup
    } else if (actionType === "Sign Up") {
      navigate("/api/teacherLoginSignup"); // Navigate to Teacher Login/Signup
    }
  }
  closeModal(); // Close the modal after navigation
 };

 return (
  <>
   <nav className="w-full flex justify-between items-center px-8 py-4 bg-white shadow-md fixed top-0
z-50">
    <div className="text-2xl font-bold text-gray-800">Name</div>
    <ul className="flex gap-6 text-gray-600 font-medium">
     <li className="cursor-pointer hover:text-gray-900">Home</li>
     <li className="cursor-pointer hover:text-gray-900">Features</li>
     <li className="cursor-pointer hover:text-gray-900">How It Works</li>
     <li className="cursor-pointer hover:text-gray-900">Contact</li>
    </ul>
    <div className="auth_buttons">
     <button
      className="bg-blue-200 m-2 text-blue-600 px-6 py-2 rounded-lg hover:bg-blue-500 hover:text-
white"
      onClick={() => openModal("Login")}
     >
      Login
     </button>
     <button
      className="bg-blue-600 text-white px-6 py-2 rounded-lg hover:bg-blue-700"
      onClick={() => openModal("Sign Up")}
     >
      Sign Up
     </button>
    </div>
   </nav>

   {/* Modal */}
   {isModalOpen && (
    <div
     className="fixed inset-0 bg-gray-600 bg-opacity-50 flex justify-center items-center z-50"
```

```
        onClick={closeModal}
      >
        <div
          className="bg-white p-8 rounded-lg shadow-xl w-96"
          onClick={(e) => e.stopPropagation()} // Prevent modal from closing on clicking inside
        >
          <h2 className="text-3xl font-semibold text-center mb-6">
            {actionType === "Login" ? "Login" : "Sign Up"} as
          </h2>
          <div className="flex justify-around mb-6">
            <button
              className="w-32 bg-blue-500 text-white py-2 rounded-md hover:bg-blue-600"
              onClick={() => handleNavigation("Student")}
            >
              Student
            </button>
            <button
              className="w-32 bg-green-500 text-white py-2 rounded-md hover:bg-green-600"
              onClick={() => handleNavigation("Teacher")}
            >
              Teacher
            </button>
          </div>
          <p className="text-center text-gray-600">
            Choose whether you are a student or a teacher to proceed with {actionType}.
          </p>
          <div className="flex justify-center gap-4 mt-6">
            <button
              className="w-28 hover:bg-red-600 py-2 text-white rounded-md bg-rose-400"
              onClick={closeModal}
            >
              Cancel
            </button>
          </div>
        </div>
      </div>
    )}

    <section className="w-full flex flex-col items-center justify-center h-screen bg-gradient-to-b from-blue-50 to-white px-8 relative overflow-hidden">
      {/* Animated Circles for Background */}
      <div className="absolute inset-0 flex justify-center items-center overflow-hidden">
        <motion.div
          className="w-72 h-72 bg-blue-100 rounded-full absolute -top-16 -left-32"
          animate={{ scale: [1, 1.2, 1] }}
          transition={{
```

```
      duration: 8,
      repeat: Infinity,
      repeatType: "mirror",
    }}
  />
  <motion.div
    className="w-96 h-96 bg-blue-200 rounded-full absolute top-40 -right-32"
    animate={{ scale: [1, 1.3, 1] }}
    transition={{
      duration: 10,
      repeat: Infinity,
      repeatType: "mirror",
    }}
  />
</div>

{/* Animated Headline */}
<motion.h1
  className="text-5xl font-bold text-gray-800 text-center leading-tight z-10"
  initial={{ y: -50, opacity: 0 }}
  animate={{ y: 0, opacity: 1 }}
  transition={{ duration: 1 }}
>
  Empower Your Learning Journey <br />
  with <motion.span
    className="text-blue-600"
    initial={{ scale: 0.8 }}
    animate={{ scale: 1 }}
    transition={{
      duration: 0.6,
      repeat: Infinity,
      repeatType: "mirror",
    }}
  >
    Name
  </motion.span>
</motion.h1>

{/* Animated Description */}
<motion.p
  className="mt-4 text-lg text-gray-600 text-center max-w-2xl z-10"
  initial={{ y: 50, opacity: 0 }}
  animate={{ y: 0, opacity: 1 }}
  transition={{ duration: 1, delay: 0.5 }}
>
  The ultimate platform for modern education. Simplify learning, enhance collaboration, and unlock
```

```
your true potential.
        </motion.p>

        {/* Animated Buttons */}
        <motion.div
          className="mt-8 flex gap-4 z-10"
          initial={{ scale: 0.9, opacity: 0 }}
          animate={{ scale: 1, opacity: 1 }}
          transition={{ duration: 1, delay: 1 }}
        >
          <motion.button
            whileHover={{ scale: 1.1, boxShadow: "0px 8px 15px rgba(0, 0, 0, 0.2)" }}
            className="bg-blue-600 text-white px-6 py-3 rounded-lg text-lg hover:bg-blue-700"
          >
            Get Started
          </motion.button>
          <motion.button
            whileHover={{ scale: 1.1, boxShadow: "0px 8px 15px rgba(0, 0, 0, 0.2)" }}
            className="border border-gray-300 text-gray-600 px-6 py-3 rounded-lg text-lg hover:border-
gray-500"
          >
            Learn More
          </motion.button>
        </motion.div>
      </section>

      <Features />
      <HowItWorks />
      <Footer />
    </>
  );
};

export default Navbar
```

Explanation:

It's an interactive and responsive navigation bar that uses animations and the ability to open up modals. It is using the use State hook for managing visibility and user actions, "Login" or "Sign Up," and the use Navigate from React Router for navigating across pages. There are sign-in links and buttons along the navigation bar. When "Login" or "Sign Up" is clicked on, a modal pops up from where the user needs to click on either "Student" or "Teacher" and then gets navigated on that respective page. The addition of motion library to the hero section and the background elements infuses it with dynamic animations. Other imports include Features, How It Works, and Footer, through which all the page is structured. The part is user engagement due to trendy design and smooth animation.

Teacher Model :

```
const mongoose=require("mongoose")
const bcrypt=require("bcrypt")

const teacherSchema = new mongoose.Schema({
  username: { type: String, required: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  role: { type: String, default: 'teacher' },
  createdAt: { type: Date, default: Date.now }
});

// Hash password before saving
teacherSchema.pre('save', async function(next) {
  if (this.isModified('password')) {
    this.password = await bcrypt.hash(this.password, 10);
  }
  next();
});

// Generate JWT token
teacherSchema.methods.generateAuthToken = function() {
  const token = jwt.sign({ _id: this._id, role: this.role },
'your_jwt_secret_key', { expiresIn: '1h' });
  return token;
};

const Teacher = mongoose.model('Teacher', teacherSchema);
module.exports = Teacher;
```

Explanation:

This is a schema and model for teacher accounts stored in a MongoDB database using Mongoose. In this example, schema determines the Teacher collection containing fields like username, email, password, and role with added middleware 'pre(' save')' that hashes through Bcrypt the password when teacher information is saved. It defines a method to produce a JWT token for authenticating teachers, which embeds an ID and role of the teacher together with a 1-hour expiration time. Then it compiles the schema into a Mongoose

model so that proper and safe well-structured CRUD operations can be maintained with respect to data related to teachers.

Route Model:

```
const express = require("express");
const { studentSignUp, teacherSignUp, studentLogin, teacherLogin,logout} = require('../controllers/authController');
const router = express.Router();

// Route definitions (don't repeat middleware here)
router.post("/student/signup", studentSignUp);
router.post("/teacher/signup", teacherSignUp);
router.post("/student/login", studentLogin);
router.post("/teacher/login", teacherLogin);
router.post("/logout",logout);

module.exports = router;
```

Explanation:

This code defines authentication routes for a Node.js application using Express. It creates a router to handle HTTP POST requests for student and teacher signup at /student/signup, /teacher/signup, login at /student/login, /teacher/login, and logout at /logout. Each of these routes maps to their corresponding functions (student Sign Up, teacher Sign Up, student Login, teacher Login, logout) from auth Controller, which contain the code for processing user credentials and registration as well as managing sessions. It exports the router into the main server file; otherwise, all the routing functionality would have been cramped up in a small server code with no separation of logic concerning routing and authentication.

Controlors Page:

```
const Student = require('../models/Student.model.js');
const Teacher = require('../models/Teacher.model.js');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const {ApiError}=require("../utils/ApiError.js");
const {ApiResponse} =require("../utils/ApiResponses.js");
const {asyncHandler}= require("../utils/asyncHandler.js")

// Student Login
const studentLogin = asyncHandler(async (req, res) => {
  const { email, password } = req.body;

  // Validate input
  if (!email || !password) {
     throw new ApiError(400, "Email and Password are required");
  }

  // Find user by email
  const user = await Student.findOne({ email });

  if (!user) {
     throw new ApiError(404, "User not found");
  }

  // Compare passwords
  const isPasswordValid = await bcrypt.compare(password, user.password);

  if (!isPasswordValid) {
     throw new ApiError(401, "Invalid credentials");
  }

  // Generate JWT token
  const token = jwt.sign({ id: user._id }, process.env.JWT_SECRET, {
     expiresIn: "1d", // Token expiration time
  });

  // Send response with token
  res.status(200).json(
     new ApiResponse(200, { token, username: user.username, email: user.email }, "Login successful")
  );
});

// Student Sign-Up
const studentSignUp =asyncHandler(async (req, res) => {

  const {username,password,email}=req.body;
```

```
  if ([username, email, password].some((field) => !field?.trim())) {
    throw new ApiError(409, "Fill all the fields");
}

  let existedStudent=await Student.findOne({
     $or:[{ username },{ email }]
  })

  if(existedStudent){
   throw new ApiError(400, "User already exist");
  }


  const user=await Student.create({
     username,
     password,
     email,
  })

  const createdUser=await Student.findById(user._id).select(
     "-password -refreshToken"
  )

  if(!createdUser){
     throw new ApiError(500, "internal server error");
  }

  res.status(200).json(
     new ApiResponse(201,createdUser,"user Succesfully registred")
  )
}

);

// Teacher Login
const teacherLogin =  asyncHandler(async (req, res) => {
  const { email, password } = req.body;

  // Validate input
  if (!email || !password) {
     throw new ApiError(400, "Email and Password are required");
  }

  // Find user by email
  const user = await Teacher.findOne({ email });

  if (!user) {
     throw new ApiError(404, "User not found");
  }
```

```
  // Compare passwords
  const isPasswordValid = await bcrypt.compare(password, user.password);

  if (!isPasswordValid) {
     throw new ApiError(401, "Invalid credentials");
  }

  // Generate JWT token
  const token = jwt.sign({ id: user._id }, process.env.JWT_SECRET, {
     expiresIn: "1d", // Token expiration time
  });

  // Send response with token
  res.status(200).json(
     new ApiResponse(200, { token, username: user.username, email: user.email }, "Login successful")
  );
});

// Teacher Sign-Up
const teacherSignUp = async (req, res) => {
  const {username,password,email}=req.body;


  if ([username, email, password].some((field) => !field?.trim())) {
    throw new ApiError(409, "Fill all the fields");
}

  let existedTeacher=await Teacher.findOne({
     $or:[{ username },{ email }]
  })

  if(existedTeacher){
   throw new ApiError(400, "User already exist");
  }


  const user=await Teacher.create({
    username,
     password,
     email,
  })

  const createdUser=await Teacher.findById(user._id).select(
     "-password -refreshToken"
  )

  if(!createdUser){
     throw new ApiError(500, "internal server error");
  }

  res.status(200).json(
```

```
    new ApiResponse(201,createdUser,"user Succesfully registred")
  )
};

const logout = asyncHandler(async (req, res) => {
  // For cookie-based tokens
  res.cookie("token", "", {
    httpOnly: true,
    expires: new Date(0), // Expire the cookie immediately
  });

  // Additional handling for token invalidation (if needed)
  const token = req.headers.authorization?.split(" ")[1];
  if (token) {
    // Optionally blacklist the token here
    console.log("Token invalidated:", token);
  }

  res.status(200).json(
    new ApiResponse(200, {},"Logged out")
  );
});

module.exports = { studentLogin, studentSignUp, teacherLogin , teacherSignUp, logout};
```

Explanation:

It includes student and teacher authentication code. It has support for the development of functionality for login, sign-up, and logout. The application verifies user credentials like email and password while logging in to generate a secure JWT token for session management if the authentication is successful. It does not allow the generation of duplicate accounts at signup time and hashes the password before saving user data into the database. There is an option to delete tokens and invalidate them on logout. It also supports very short responses with self-sufficient error-handling capabilities through utility classes like API Error, API Response, and async Handler. This kind of security, speed, and reliability while dealing with the workflow of an application may be useful while applying user authentication.

Features Page :

```
import React from 'react';

const FeaturesSection = () => {
  return (
    <section className="bg-gradient-to-br from-blue-50 to-white py-
20">
      <div className="max-w-7xl mx-auto px-6 lg:px-8">
        {/* Section Title */}
        <h2 className="text-5xl font-extrabold text-center text-blue-700
mb-10 leading-tight font-[Poppins]">
          Unlock the Future of Learning
        </h2>
        <p className="text-center text-gray-600 text-lg mb-16 max-w-
3xl mx-auto font-[Roboto]">
          Explore cutting-edge tools designed for collaboration, time
management, blockchain security, and personalized learning
environments.
        </p>

        {/* Feature Cards */}
        <div className="grid gap-12 md:grid-cols-2 lg:grid-cols-4">
          {/* Card 1: Blockchain-Based Verification */}
          <div className="relative bg-white rounded-3xl shadow-xl
overflow-hidden hover:shadow-2xl transition-shadow duration-300
group">
            <div className="absolute inset-x-0 -top-16 h-32 bg-gradient-
to-r from-blue-500 to-blue-300 transform skew-y-6 shadow-lg group-
hover:scale-110 transition-transform duration-300"></div>
            <div className="relative z-10 p-8">
              <h3 className="text-2xl font-bold text-blue-700 mb-4 group-
hover:text-blue-900 font-[Merriweather]">
                Blockchain-Based Verification
              </h3>
              <p className="text-gray-600 group-hover:text-gray-800 font-
[Roboto]">
                Ensure the authenticity of certifications and records with
blockchain-powered security for trust and transparency.
              </p>
            </div>
          </div>

          {/* Card 2: QR Code-Enabled Sharing */}
          <div className="relative bg-white rounded-3xl shadow-xl
overflow-hidden hover:shadow-2xl transition-shadow duration-300
group">
            <div className="absolute inset-x-0 -top-16 h-32 bg-gradient-
to-r from-blue-500 to-blue-300 transform skew-y-6 shadow-lg group-
```

```
hover:scale-110 transition-transform duration-300"></div>
          <div className="relative z-10 p-8">
            <h3 className="text-2xl font-bold text-blue-700 mb-4 group-
hover:text-blue-900 font-[Merriweather]">
              QR Code-Enabled Sharing
            </h3>
            <p className="text-gray-600 group-hover:text-gray-800 font-
[Roboto]">
              Share assignments, resources, and status updates instantly
with secure QR code integration.
            </p>
          </div>
        </div>

        {/* Card 3: Live Video Mentorship */}
        <div className="relative bg-white rounded-3xl shadow-xl
overflow-hidden hover:shadow-2xl transition-shadow duration-300
group">
          <div className="absolute inset-x-0 -top-16 h-32 bg-gradient-
to-r from-blue-500 to-blue-300 transform skew-y-6 shadow-lg group-
hover:scale-110 transition-transform duration-300"></div>
          <div className="relative z-10 p-8">
            <h3 className="text-2xl font-bold text-blue-700 mb-4 group-
hover:text-blue-900 font-[Merriweather]">
              Live Video Mentorship
            </h3>
            <p className="text-gray-600 group-hover:text-gray-800 font-
[Roboto]">
              Connect with mentors in real-time through high-quality video
sessions for personalized guidance.
            </p>
          </div>
        </div>

        {/* Card 4: Smart Scheduling and Reminders */}
        <div className="relative bg-white rounded-3xl shadow-xl
overflow-hidden hover:shadow-2xl transition-shadow duration-300
group">
          <div className="absolute inset-x-0 -top-16 h-32 bg-gradient-
to-r from-blue-500 to-blue-300 transform skew-y-6 shadow-lg group-
hover:scale-110 transition-transform duration-300"></div>
          <div className="relative z-10 p-8">
            <h3 className="text-2xl font-bold text-blue-700 mb-4 group-
hover:text-blue-900 font-[Merriweather]">
              Smart Scheduling and Reminders
            </h3>
            <p className="text-gray-600 group-hover:text-gray-800 font-
[Roboto]">
              Stay on top of deadlines with AI-powered scheduling tools
and customizable reminders tailored to your needs.
            </p>
```

```
          </div>
        </div>
      </div>
    </div>
  </section>
 );
};

export default FeaturesSection;
```

Explanation:

This is the code for a Features Section component in React, which will render a section containing four feature cards along with their descriptions. The section has a gradient background and has a title and an introductory text in its center. The four feature cards narrate a particular tool or service-the Blockchain-Based Verification, QR Code-Enabled Sharing, Live Video Mentorship, and Smart Scheduling and Reminders. Stylistic cards that utilize the usage of Tailwind CSS elements on hover effects, gradient features and use of shadows in overall layout. It's rather productive while referring the highlighted features of the concerned service and commodity visually for further discussion.

Plagiarism Report Attach