A Student Attendance Application Based on NaiveCoin

Group 9

MA Zhiyuan, WANG Zepeng, QIN Cailing, XU Junda, DU Jiahao

Course Number: COMP5521

November 30, 2024

Table of Contents

Introduction

Project background

The application of blockchain technology has expanded from cryptocurrency to numerous real-world scenarios, with education being one of them. Traditional attendance systems often face challenges such as a high risk of data tampering, lack of transparency, and inefficient record management. Blockchain technology, with its decentralized, immutable, and transparent characteristics, offers a promising solution to these problems.

In this project, we implement a student attendance system using blockchain technology, built on the NaiveCoin. This system aims to record attendance information securely and transparently, ensuring data integrity while simplifying attendance management processes. By incorporating blockchain features such as Proof of Work (PoW) and cryptographic signatures, the system provides a solid technical foundation for decentralized attendance data management, eliminating the reliance on centralized control.

Problem statement

The current student attendance management system has the following major problems:

The system relies on manual data entry, which is error-prone and inefficient, especially when recording and checking attendance.

Centralised systems are susceptible to data tampering, affecting the accuracy and credibility of attendance records

Students and teachers would find it difficult to view and verify the authenticity of attendance records. Especially when the data is controversial, there is a lack of effective resolution mechanisms.

At the same time, the NaiveCoin framework used as the basis of this project also has some limitations and cannot be directly applied to the student attendance system. These problems include:

NaiveCoin is just a blockchain framework, mainly used to demonstrate the basic functions of blockchain, such as blockchain structure, wallet, mining and basic transaction functions, and lacks core

functional modules related to the attendance system, such as user registration, attendance record storage and query.

The current NaiveCoin does not support the dynamic adjustment of the difficulty mechanism, which may lead to reduced mining efficiency or excessive network load during peak attendance periods.

NaiveCoin lacks an effective fork resolution strategy, which may cause chain inconsistency in multi-node competition and affect data reliability.

It only provides a basic on-chain operation interface, lacks a friendly user interface, and cannot meet the needs of efficient interaction between students and teachers.

Objectives

Design and implement a blockchain-driven attendance management system based on the NaiveCoin, supporting core functions such as student sign-in, attendance record storage and query. The specific goals are as follows:

- Provide students and teachers with a simple and intuitive operation interface, support sign-in record signature, attendance record query, data verification and other functions to improve user experience.
- Based on the proof-of-work (PoW) mechanism of the blockchain, the mining difficulty is dynamically adjusted to ensure the stability and efficiency of the system under different load conditions.
- Develop a simple chain fork solution to select the longest chain or the most complex chain when multiple nodes conflict to ensure data consistency and system reliability.
- Provide secure student information management: Use the encryption mechanism of public and private keys to realize the registration and authentication of student information to ensure the security and uniqueness of data.
- Store all attendance data on the blockchain, use its tamper-proof and transparent characteristics to avoid malicious modification of data, and provide a convenient verification method.

Literature Review

NaiveCoin

Naivecoin[1] is an educational blockchain project implemented in JavaScript and developed by Conrado Quilles Gomes. As a simplified version of the blockchain system, it retains the core features of the blockchain: including the basic block structure, the proof-of-work (PoW) algorithm, P2P network communication, and the HTTP API interface. Compared to complex systems such as Bitcoin, Naivecoin's clear code structure and high degree of modularity make it particularly suitable as a basis for educational programs and development prototypes.

Blockchain for Education

Blockchain technology was introduced in 2008 and achieved an important breakthrough in 2014 by integrating smart contract functions. With the continuous maturation of the technology, blockchain has carried out many practical applications in the field of education: MIT took the lead in launching the blockchain-based Digital Diploma project[2] in 2017, which issued digital certificates to more than 100 graduates through the Bitcoin blockchain technology to enable the graduates to securely share and validate their academic certificates; The European Commission has developed a blockchain-based digital certificate system through the European Blockchain Services Infrastructure (EBSI) project[3], which has developed a blockchain-based digital certificate system to help students achieve seamless mutual recognition of credits and qualifications across Europe. These success stories fully demonstrate the great potential of blockchain technology in enhancing the efficiency of education management, safeguarding data security and promoting education innovation.

Blockchain characteristics

The characteristics of the NaiveCoin-based student attendance system leverage blockchain technology to ensure secure, transparent, and decentralized management of attendance data. These characteristics address common issues found in traditional attendance systems, such as data tampering, inefficiency, and lack of accessibility. The characteristics of Blockchain are:

- Immutable: Attendance records stored on the blockchain cannot be deleted, ensuring data permanence and reliability.
- Unchangeable: Data once recorded cannot be modified, guaranteeing the integrity of attendance records over time.
- Secure: Records are encrypted and validated using cryptographic techniques like hashing and Proof-of-Work, protecting against tampering.
- Transparent: Attendance data is accessible to all nodes, enabling students and lecturers to verify records and ensure accountability.
- Peer-to-Peer: Transactions are recorded directly between students and the blockchain without intermediaries, ensuring efficiency and accuracy.
- Distributed: Data is replicated across all nodes, reducing the risk of single-point failures and ensuring record availability.

Previous Research

There have been several innovative practical explorations in the research field of blockchain technology applied to student attendance systems. Early research focused on the transition from traditional attendance systems to blockchain solutions. Arnab et al. (2021) proposed an Ethereum-based smart-contract attendance system, which significantly improves the trustworthiness of attendance data through the combination of biometrics and blockchain[4]. In terms of practical applications, the blockchain-based student attendance tracking system developed by Nanyang Technological University successfully automated attendance recording and real-time data verification. The system uses distributed ledger technology, which not only improves the accuracy of attendance records but also greatly reduces management costs[5]. These studies provide an important theoretical foundation and practical reference for the development of student attendance systems based on NaiveCoin.

Methodology

RAD model

This project was developed using the Rapid Application Development (RAD) model, which emphasizes rapid prototyping and iteration and is particularly suitable for applications with short development cycles and relatively independent functional modules. In the process of developing a student attendance system based on Naivecoin, the implementation of the RAD model is divided into four main phases:

Requirements Planning Phase

In the process of determining the core requirements of the system, we deeply analyzed the key functional requirements of the blockchain attendance system. The first task was to design a blockchain-based attendance record storage mechanism to ensure the data's tamper ability and traceability. Secondly, the specific requirements of the data uploading process and validation mechanism are clarified, including the format specification of attendance data and block generation conditions. At the same time, considering the characteristics of the project developed based on the Naivecoin framework, the limitations and scalability of the framework are evaluated in detail, and a reasonable time plan is set for the subsequent development.

User Design Phase

This phase focuses on the overall architecture design of the system. We put the attendance information in the block data and adjust the design of the block header accordingly. At the same time, the consensus mechanism was optimized to make it more suitable for the application scenario of the attendance system. Through repeated testing and verification, the rationality of the design of the attendance data structure and the reliability of the block generation and linking mechanism are ensured.

Construction phase

In this phase, we carried out a lot of code development work. Based on the Naivecoin framework, we focused on the implementation of the block storage function of attendance data, including the core

modules such as block packing and linking. The original block verification mechanism is improved, and the integrity check of attendance data is added. Meanwhile, a data query interface is developed to support efficient retrieval of historical attendance records. During the development process, continuous code testing and performance optimization are carried out to ensure the stability and responsiveness of the system.

Cutover Phase

The final phase focused on completing the deployment and wrapping up of the system. A complete system test was conducted to verify the correctness of all functional modules. Detailed technical documents were written, including system architecture description, interface documents and deployment guidelines. Collected system operation data through performance monitoring and testing, which provided the basis for subsequent optimization. Throughout the process, development experience and improvement suggestions were continuously recorded, laying the foundation for the subsequent upgrading of the system.

System Framework

The student attendance system based on NaiveCoin adopts a three-layer architecture, including the presentation layer, application layer and data layer, to achieve the complete process from user interaction to blockchain storage. The following Figure 1 is a system architecture diagram:

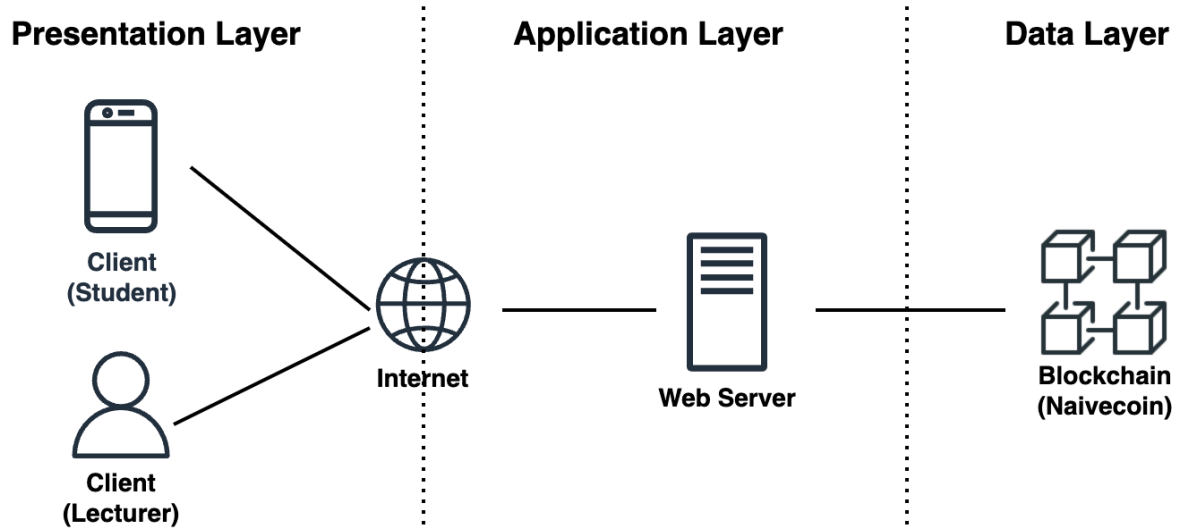


Figure 1 System Architecture Diagram

- Presentation layer: Users access the system through clients (students, teachers).
- Application layer: The web server is responsible for processing user requests, verifying data and interacting with the blockchain.
- Data layer: The blockchain stores all attendance records and provides mining and query support.

System analysis and design

The goal of the system is to develop a decentralised student attendance management system based on naivecoin to achieve data that cannot be tampered with, transparent, and efficient recording and querying. To this end, the system needs to meet the following requirements:

- User registration: Students generate public and private key pairs through the client, complete identity registration, and record the public key on the blockchain.
- Attendance records: Teachers post sign-in requirements. Students generate signature data when signing in and submit it to the blockchain. The system needs to verify the validity of the sign-in data and package it for storage.
- Query function: Students can query their own attendance records. Teachers can query attendance records by student or course.
- Mining Function: Student nodes are responsible for verifying the validity of the attendance data and packaging it to generate new blocks. Upon successful mining, the student node broadcasts the new block to the blockchain network and receives a reward to incentivize participation.

The Use Case Diagram (figure 2) showcases the main interaction functions of students and teachers within the NaiveCoin-based attendance system. The system defines the responsibilities and operational processes of these two user roles through role partitioning and functional module design. The core functions for students include Registration, Login and Logout, ensuring identity uniqueness and security. Students can also participate in the mining process (Mint) by verifying attendance records and generating new blocks

to earn rewards. The key attendance functions for students are signing in (Scanning the QR Code to Attendance) and uploading attendance records (Post Attendance), which ensure accurate and real-time attendance data. Additionally, students can check their own attendance records (Check Attendance) through the system. The main functions for teachers are focused on attendance management, including posting attendance events (Post Attendance) and checking course or student attendance data (Check Attendance). The simplified interactive interface enables instructors to efficiently accomplish their course attendance tasks.

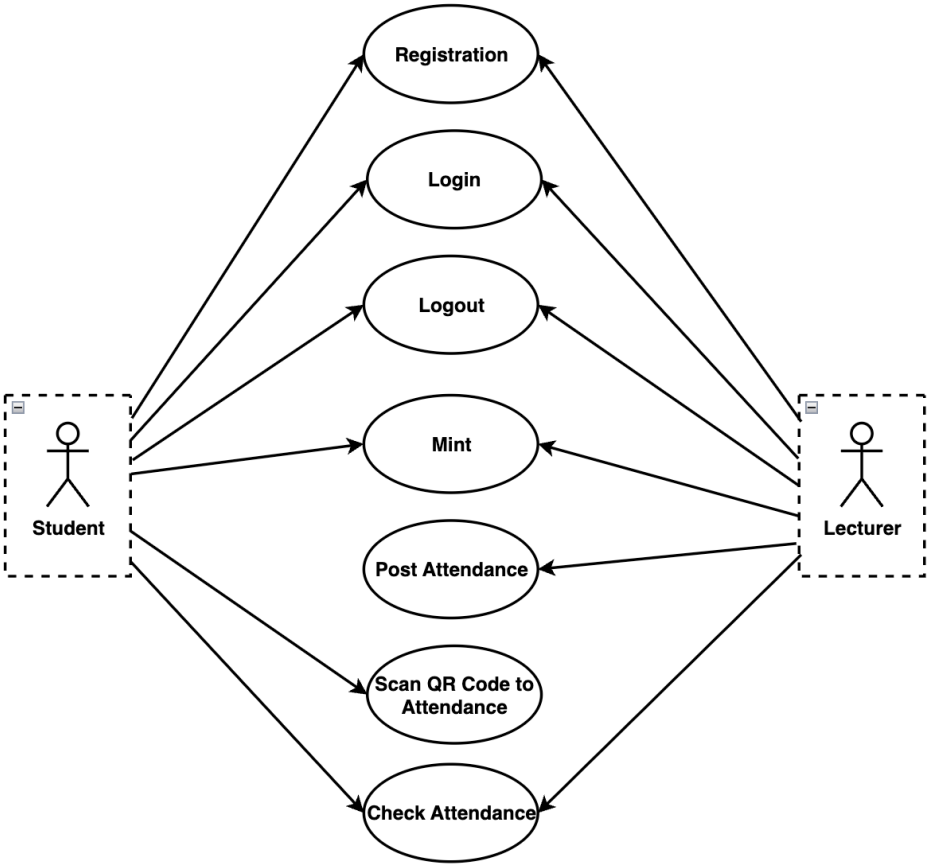


Figure 2 Use Case Diagram

In the system design, we have not only clarified the functional requirements of users but also sorted out the detailed operation flow of the system to ensure the coherence and efficiency of user interaction and function realisation. The activity diagram(figure 3) shows the activity flow chart of the

system, which visualises the operation logic of students and teachers, as well as the key steps of different user roles in accomplishing their respective tasks.

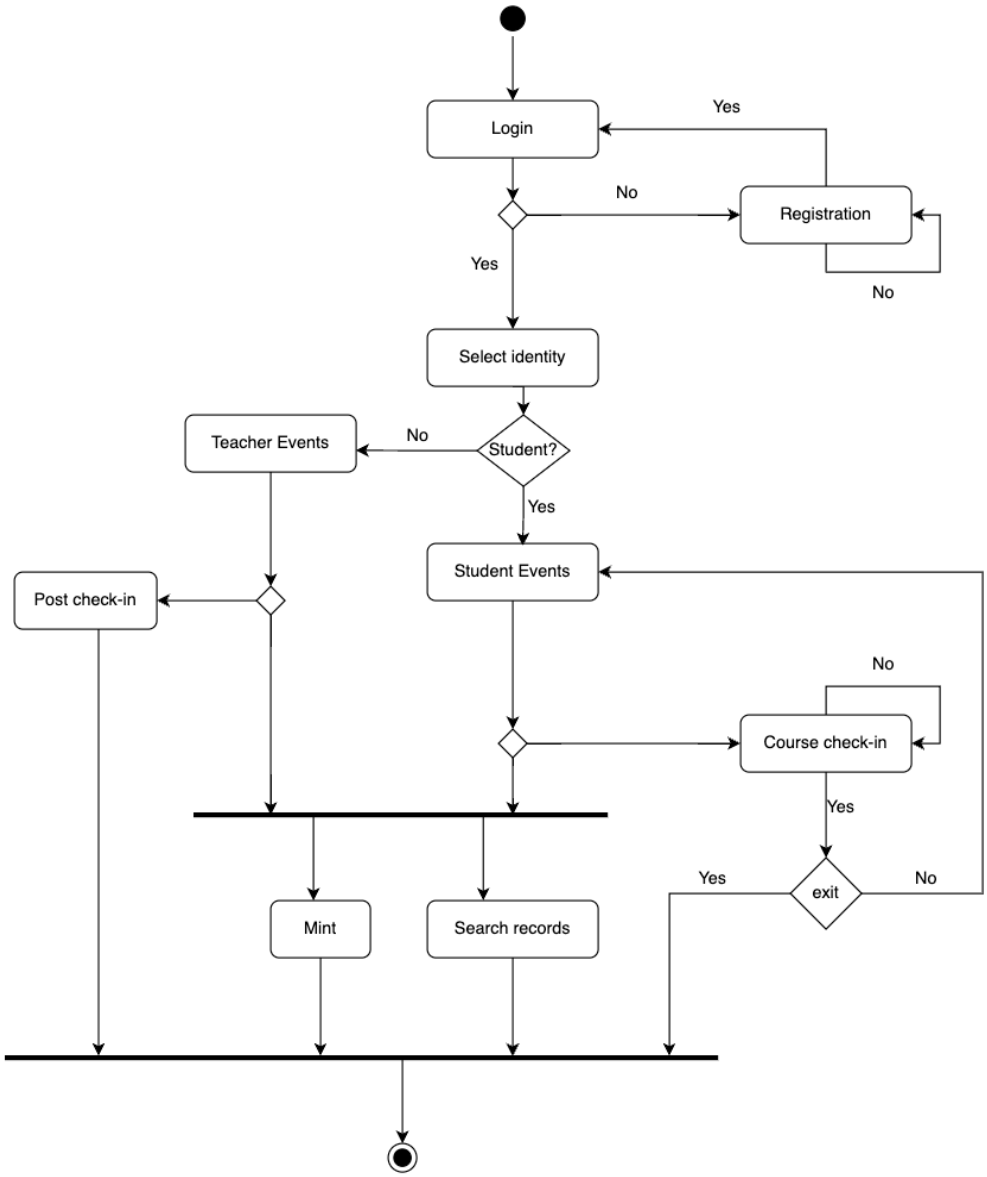


Figure 3 Activity Diagram

The system starts with user login, and users with different identities will enter different branches of operation. If a user has not yet registered, the system will guide them through the registration process; registered users can then choose to log in as a student or a teacher and continue the operation. After entering the system, students can participate in attendance events and complete tasks by signing in or checking their

attendance history. In addition, the system supports students to participate in the blockchain mining process to verify and package attendance data. Teachers, on the other hand, can create new attendance events or view attendance records. The design of the activity flowchart clarifies the core logic of the system, including the functions of login and authentication, event creation and attendance record, data query and mining mechanism. This process planning based on user role branching not only improves the usability of the system but also ensures the efficient operation of the system functions.

Development Environment

React Framework

React, developed and maintained by Facebook, is a JavaScript library for building user interfaces that is efficient, flexible, and highly extensible. React adopts a component-based development approach, breaking down pages into reusable and independent components, making the code easier to maintain and scale. In this project, React is used to build the frontend interface, providing students and lecturers with an intuitive and user-friendly platform.

Next.js

Next.js is a powerful React framework that lets developers build high-performance web apps with server-side rendering, static site generation, and automatic code splitting. It provides an enhanced development experience with features like built-in CSS support, TypeScript integration, and automatic optimization, making it ideal for creating scalable and SEO-friendly websites. Next.js simplifies complex web development tasks by offering a simple setup, efficient routing, and seamless deployment options.

NaiveCoin

NaiveCoin is a lightweight blockchain implementation designed for educational and experimental purposes. It has a simple codebase and clear structure, making it suitable for secondary development. This project uses NaiveCoin as the backend framework, enhanced with additional functionalities to meet the requirements of the student attendance system.

Tools Support

Github

GitHub is a Git-based distributed version control platform that provides powerful support for code management and team collaboration for project development: Each code commit, modification and update is recorded, making it easy to track development history and roll back versions when needed; Through the branching mechanism, multiple developers can develop different functional modules in parallel to avoid code conflicts. Developers can submit code changes via Pull Request, and other team members can review the code to ensure code quality and consistency. Teams can record and assign issues in the development process and track their progress at any time.

Trello

Trello is a visual task management tool for tracking project progress and assigning tasks: Development teams can assign specific tasks to each member and mark their priorities and deadlines. Drag-and-drop cards to move between “To-Do”, “In-Progress” and “Completed” statuses to reflect task status in real-time. Team members can comment, upload attachments, or add notes to the task cards, ensuring that the communication and execution of each task are documented. Tasks in each development cycle can be managed through Kanban, ensuring clear and controllable goals for each iteration.

Visual Studio Code

VSCode is a lightweight yet powerful code editor developed by Microsoft, widely regarded as one of the best tools for modern software development. Its flexibility, speed, and rich extension ecosystem make it an indispensable tool for developers.

Project Work Division and Design

Table 1 Project Task Distribution Table

No.	*Task Name*	*Task Description*	*Owner*	*Collaborator*
1	*Requirement Analysis*	*Analyse project requirements, and written descriptions, define objectives, and allocate tasks.*	*MA Zhiyuan*	*QIN Cailing, DU Jiahao, XU Junda, WANG Zepeng*
2	*Task Planning*	*Plan tasks, assign work, and track progress in real time using Gantt charts and Trello.*	*MA Zhiyuan*	
3	*Front-End Development*	*Implement core functionalities (Student side: login, registration, logout, data query; Teacher side: create events), integrate code, resolve conflicts, optimise and enhance features, and develop the user interface (UI).*	*MA Zhiyuan*	*QIN Cailing, DU Jiahao*
4	*Blockchain Enhancements*	*Address the NaiveCoin fork issue, add essential APIs (such as login authentication API), and support the implementation of front-end functionalities.*	*MA Zhiyuan*	
5	*Student Registration*	*Develop student registration and login functionality*	*QIN Cailing*	*MA Zhiyuan*
6	*Teacher Event Management*	*Develop a page for Lecturers to create events and implement an event search function. Optimise the visualisation of the return window, ensure account segregation, and support three filtering methods for querying attendance records (student ID, event ID, and time).*	*DU Jiahao*	*MA Zhiyuan, XU Junda*
7	*Attendance Functionality*	*Publishing check-ins, displaying all check-ins, student check-ins, preventing duplicate check-ins, and modifying the Transaction class in NaiveCoin to meet the requirements for check-ins.*	*XU Junda*	
8	*Blockchain Optimization*	*Implement dynamic difficulty adjustment functionality, optimise and test it, and modify the basic structure of the block and the hash computation method.*	*WANG Zepeng*	
9	*Project Documentation Writing*	*Draft and revise project documentation content.*	*QIN Cailing*	*WANG Zepeng, XU Junda*
10	*Design and Production of Project Diagrams*	*Design and create a System Architecture Diagram, Use Case Diagram, Activity Diagram, and Wireframe Diagram.*	*WANG Zepeng*	*MA Zhiyuan, QIN Cailing, DU Jiahao*

System Analysis and Design

Requirement Elicitation

To define the system's scope and objectives, the team conducted brainstorming sessions, leveraging prior development experience to identify core functionalities. This process focused on understanding stakeholder needs and outlining the essential requirements.

Requirement Specifications

The **Requirement Specifications** phase formalizes the identified requirements into structured definitions. This includes detailing the expected inputs and outputs for each major functionality of the system. The specifications focus on what the system should achieve rather than how it is implemented.

Functional Requirements

1. **Login (User Authentication)**
    - **Purpose**: Allow both teachers and students to securely log in to the system with role-based access.
    - **Input**:
        - userId: A unique identifier for the user (e.g., username or email).
        - password: The user's login password.
    - **Output**:
        - On success: User's role (teacher or student) and blockchain address.
        - On failure: Error message (e.g., "Invalid credentials").
2. **Sign-In Event Creation (Teacher)**
    - **Purpose**: Enable teachers to create and publish sign-in events with relevant parameters.
    - **Input**:
        - walletId: Teacher's blockchain wallet ID for transaction validation.
        - password: Teacher's password for authentication.

        - fromAddress: The blockchain address initiating the transaction (teacher's wallet address).
        - toAddress: The blockchain target transaction address.
        - changeAddress: Address for recording excess funds.
        - teacherId: Teacher's unique identifier (equal to userId from login).
        - deadline: Deadline for the sign-in event.
        - remark: Additional remarks for the event.
    - **Output**:
        - On success: Successful transaction.
        - On failure: Error message indicating invalid input or authentication failure.

3. **Sign-In Participation (Student)**
    - **Purpose**: Allow students to participate in active sign-in events and record their attendance on the blockchain.
    - **Input**:
        - walletId: Student's blockchain wallet ID for recording the sign-in operation.
        - password: User's password for authentication.
        - fromAddress: The blockchain address initiating the transaction.
        - toAddress: The blockchain target transaction address.
        - changeAddress: Address for recording excess funds.
        - studentId: Student's unique identifier.
        - deadline: Deadline for the sign-in participation.
        - remark: Additional remarks for the sign-in.
    - **Output**:
        - On success: Successful transaction.
        - On failure: Error message indicating invalid input or authentication failure.

4. **Mining (Reward Distribution)**

- **Purpose**: Allow both teachers and students to mine blockchain-based rewards for their participation.
- **Input**:
  - rewardAddress: Address where the mining reward will be sent.
  - feeAddress: Address for transaction fees related to mining operations.
- **Output**:
  - On success: Successful transaction.
  - On failure: Error message indicating an error in reward distribution.

5. **Query Functionality**
   - **Purpose**: Allow teachers and students to query attendance records or related data securely and efficiently, ensuring accessibility while maintaining data integrity.
   - **Input**: TBD.
   - **Output**: TBD.
6. **Basic Fork Resolution**
   - **Purpose**: Ensure the blockchain system resolves any forks automatically, maintaining a consistent and unified ledger to avoid discrepancies in transaction history or records.
   - **Input**: TBD.
   - **Output**: TBD.
7. **The Design of Dynamic Difficulty**
   - **Purpose**: Adjust the mining difficulty dynamically based on system activity, ensuring fair and balanced reward distribution and preventing resource exhaustion.
   - **Input**: TBD.
   - **Output**: TBD.

Non-Functional Requirements

1. **Performance**:
   - System response time for key operations:

        - Login: Less than 2 seconds.
        - Sign-in creation: Less than 3 seconds.
        - Blockchain transactions: Less than 5 seconds.

2. **Security**:
    - User data, including credentials, must be encrypted.
    - Blockchain must ensure the immutability of sign-in records and transactions.
    - Role-based access control to prevent unauthorized access to restricted functionalities.
3. **Usability**:
    - The system should feature an intuitive user interface that minimizes the learning curve for both teachers and students.
    - Interfaces must be mobile-responsive and accessible on various devices.
4. **Scalability**:
    - The system should support increasing numbers of users and blockchain transactions without significant performance degradation.
5. **Maintainability**:
    - Modular design to facilitate updates and future feature expansions.

System Constraints

1. **Technological Constraints**:
    - The system must utilize blockchain technology for secure data storage and transactions.
    - Limited development time requires prioritization of core functionalities.
2. **Operational Constraints**:
    - Teachers must have access to enhanced features, such as querying aggregate attendance data.
    - The system must prevent unauthorized access to sensitive information.

This structured specification ensures clarity in what the system is expected to achieve and provides placeholders for additional functionalities to be detailed in future stages.

System Architectural Design

The **System Architectural Design** outlines the system's structure and the interaction between its components. This design leverages a blockchain-based backend (Naivecoin) to handle data immutability and transaction processing, eliminating the need for a traditional database.

Overall System Architecture

The system follows a **client-server architecture** with blockchain integration as the core backend. It is divided into the following layers:

1. **Frontend (Presentation Layer)**:
    - A web-based interface that serves as the primary point of interaction for teachers and students.
    - Handles role-based access and data validation before transmitting requests to the backend.
2. **Backend (Blockchain Layer)**:
    - Powered by Naivecoin, it manages all data storage and transaction processing.
    - Stores immutable records for login, sign-in events, participation, and mining rewards.
    - Handles consensus mechanisms and dynamic difficulty adjustment for mining.

Module Division and Interaction

1. **Authentication Module**:
    - Verifies user credentials (userId and password) and determines user roles (teacher or student).
2. **Sign-In Management Module**:
    - Allows teachers to create sign-in events with parameters like deadlines and remarks.
    - Enables students to participate in active events, storing attendance data securely on the blockchain.
3. **Mining Module**:
    - Facilitates reward distribution based on blockchain activity.

    - Adjusts mining difficulty dynamically to balance system load and ensure fair reward allocation.

4. **Query Module**:
    - Enables users to retrieve sign-in and reward data directly from the blockchain.
    - Supports filtering by event ID, date range, or user role.
5. **Blockchain Operations**:
    - Manages all blockchain interactions, including transaction validation, block creation, and fork resolution.

Communication Flow

1. **Frontend to Blockchain Backend**:
    - All user actions (e.g., login, sign-in creation, participation) generate API calls to the Naivecoin backend.
    - The front end formats requests and handles responses for display.
2. **Blockchain Operations**:
    - Transactions are validated and stored in the blockchain.
    - Mining operations ensure data security and reward distribution.

Blockchain's Role

Naivecoin serves as the system's backend, providing:

- **Immutability**: Secure, tamper-proof storage for sign-in and reward data.
- **Transparency**: All records are stored on the blockchain, ensuring trustworthiness.
- **Consensus**: Maintains the system's integrity and resolves chain forks.

This modular architecture simplifies the system design by leveraging Naivecoin as the sole backend, ensuring immutability and security while maintaining a lightweight implementation.

User Interface Design

The **User Interface Design** aims to create a user-friendly and responsive system for both teachers and students. The interface provides distinct role-based functionalities while ensuring shared access to core features like sign-in management and mining rewards.

Design Principles

1. **Role-Based Accessibility**:
    - Both teachers and students can access shared functionalities like mining rewards and attendance querying, but additional features like sign-in creation are reserved for teachers.
    - The interface dynamically adapts based on the user's role.
2. **Simplicity**:
    - The design emphasizes a clean and organized layout for effortless navigation.
    - Essential actions are highlighted for easy access.
3. **Responsiveness**:
    - Designed to work seamlessly across devices, including desktops, tablets, and smartphones.
4. **Real-Time Feedback**:
    - Users receive immediate feedback on actions such as mining rewards, participation in sign-ins, and transaction success or failure.

Key Components and Screens

1. **Login Page**:
    - **Features**:
        - Input fields for userId and password.
        - Role identification is handled automatically after login.
        - A "Login" button that initiates authentication.
    - **Design**:

        - Centralized login form with a clean, minimalistic layout.

2. **Dashboard**:
    - **Features**:
        - Displays relevant options based on the user's role:
            - **Teachers**: Sign-in creation, attendance querying, and mining rewards.
            - **Students**: Participation in sign-in events, attendance records, and mining rewards.
    - **Design**:
        - Role-based menus with visually distinct sections for core functionalities.
3. **Sign-In Event Creation (Teacher)**:
    - **Features**:
        - Input fields for event details (e.g., deadline, remarks).
        - A "Create Event" button that records the event on the blockchain.
    - **Design**:
        - Step-by-step guided form to ensure accuracy.
4. **Sign-In Participation (Student)**:
    - **Features**:
        - List of active sign-in events.
        - "Join Event" button to confirm participation and record attendance.
    - **Design**:
        - Visual status indicators for active and expired events.
5. **Mining Rewards (Shared for Teachers and Students)**:
    - **Features**:
        - Overview of earned mining rewards.
        - A "Start Mining" button to initiate the mining process.
        - A "Claim Reward" button for receiving blockchain-based rewards.

- **Design**:
    - Display of mining statistics (e.g., rewards earned, transactions processed).
    - Progress bar for mining tasks.

6. **Query Functionality**:
    - **Features**:
        - Teachers: Query attendance records by event ID or date range.
        - Students: View personal attendance history and associated rewards.
    - **Design**:
        - Filter options (e.g., dropdowns, date pickers) for easy querying.

User Friendly Experience Principles

1. **Shared Features**:
    - Both roles share access to mining functionalities, ensuring equal opportunities to earn blockchain-based rewards.
    - The mining interface adapts to display progress and statistics relevant to the user's activity.
2. **Real-Time Feedback**:
    - Notifications for successful transactions, errors, or blockchain confirmations are displayed immediately.
    - Mining operations show real-time status (e.g., "Mining in Progress").
3. **Intuitive Navigation**:
    - A streamlined menu ensures that users can access primary features directly from the dashboard.

Wireframe diagram

In order to better sort out the user flow, a wireframe diagram was designed at the beginning of the project. It can be viewed by clicking on the link below.

https://www.figma.com/proto/s3Szfwdc7tbnbmcCwcbHt6/COMP5521?page-id=2%3A2&node-id=2-4&node-type=canvas&viewport=291%2C302%2C0.04&t=cMDrG5ulFOBst1xG-1&scaling=min-zoom&content-scaling=fixed&starting-point-node-id=2%3A4&show-proto-sidebar=1

Implementation and Discussion

System Hardware

Table 2 The hardware used in System Development

Device Name	Operating System	Usage
Xiaomi 14 Ultra	Xiaomi HyperOS (Android)	Test student attendance function using QR code scanning.
iPhone 15 Pro	iOS	Test student attendance function using QR code scanning.
Macbook air 2020 m1	MacOS	Develop, test and run as a local server.
Lenovo Yoga Pro 14s IRH8	Windows 11 Pro for Workstations	Develop, test and run as a local server.

Students can sign in to class by scanning the QR code published by the teacher using their mobile devices. Taking into account the timeliness of class attendance and the inconvenience of students entering private keys during sign-in, this system only supports class sign-in functionality through smart devices equipped with QR code scanning capabilities. Moreover, Teachers can use their terminal devices to check specific students' attendance records through the system.

System Software

Table 3 Table of software used in this project development

Name	Introduction	Usage
Visual Studio Code	Visual Studio Code, also known as VS Code, is a source-code editor developed by Microsoft.	As the main IDE for the software development of this project.
GitHub	GitHub is an Internet hosting service that uses Git for software development and version control.	GitHub is used to save and manage programs written, record modification logs, etc.
Trello	Trello is a widely popular project management and collaboration tool.	Visualise project management through an intuitive interface featuring Boards, Lists and Cards.

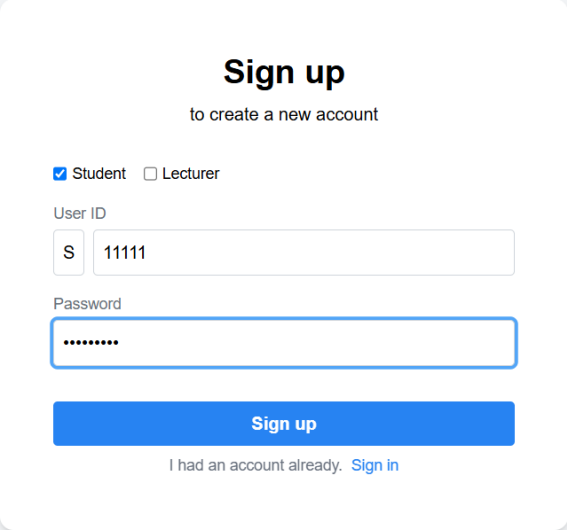
Language

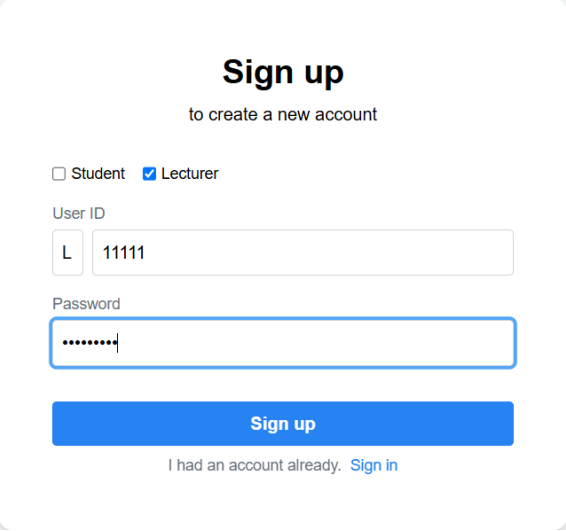
This project's development involves multiple programming languages and platforms. The project is developed based on the naivecoin project. The web application is developed using the React

framework, primarily involving JavaScript, HTML and CSS. The backend logic is implemented using Node.js.

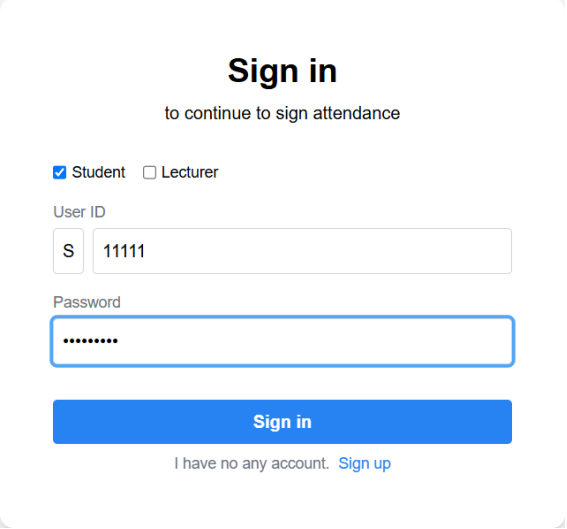
System Implementation

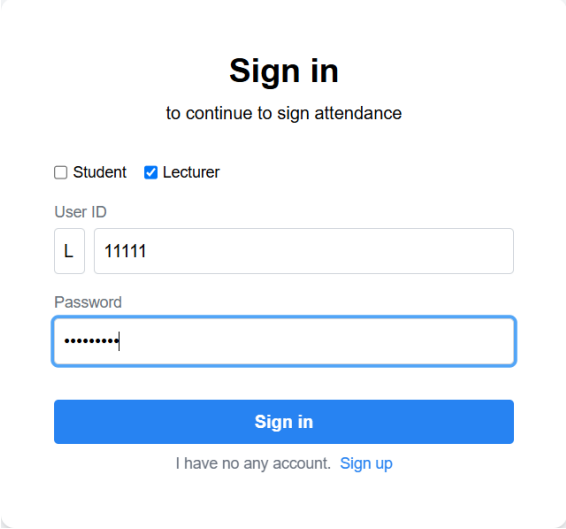
The Process of User Registration (Student and Lecturer)



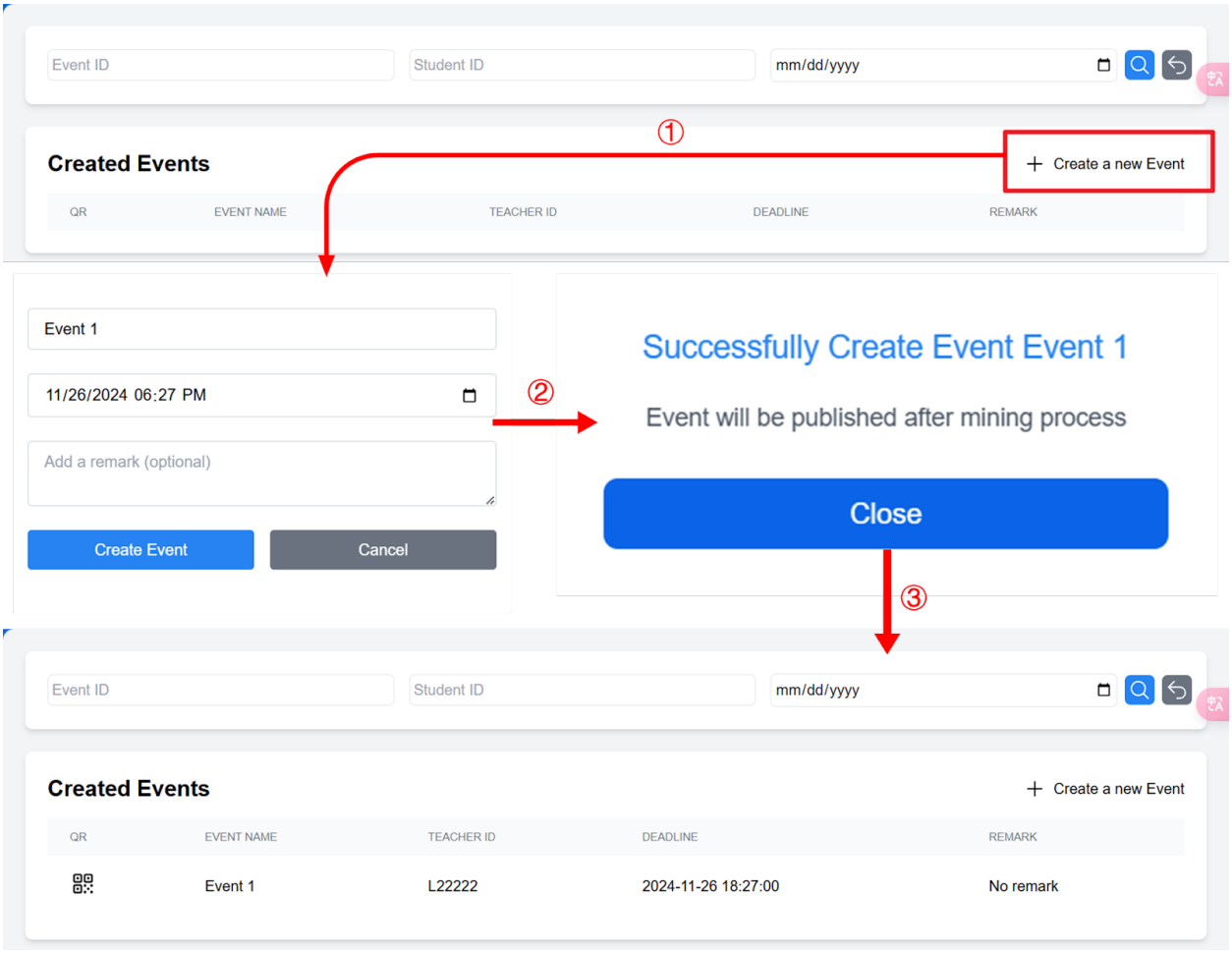


The Process of User Login (Student and Lecturer)

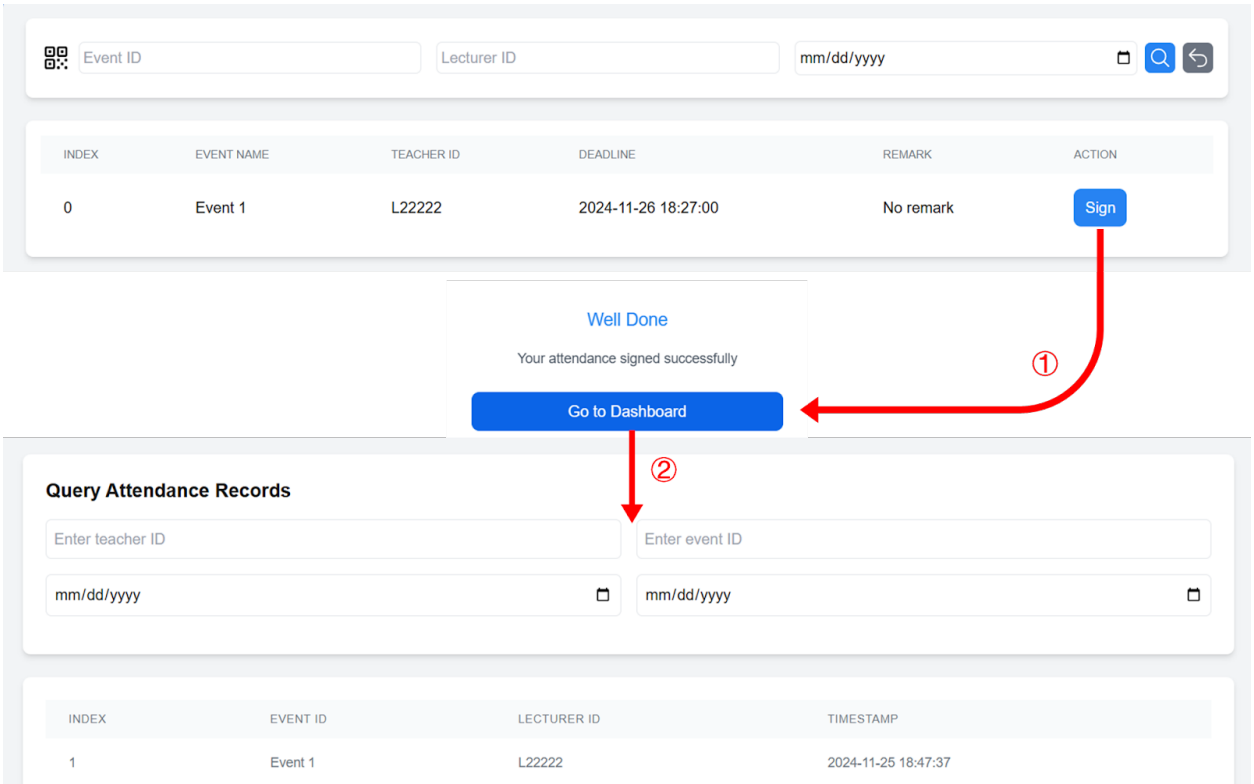




Teacher Posting Attendance Function Implementation

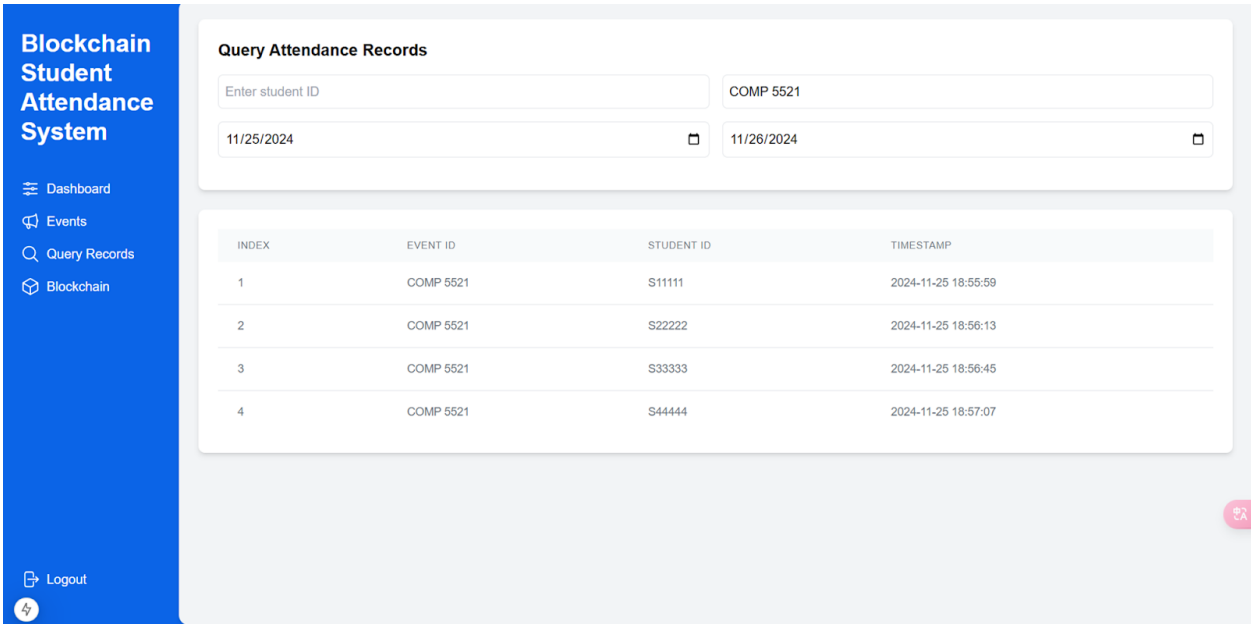


Student Attendance Function Implementation

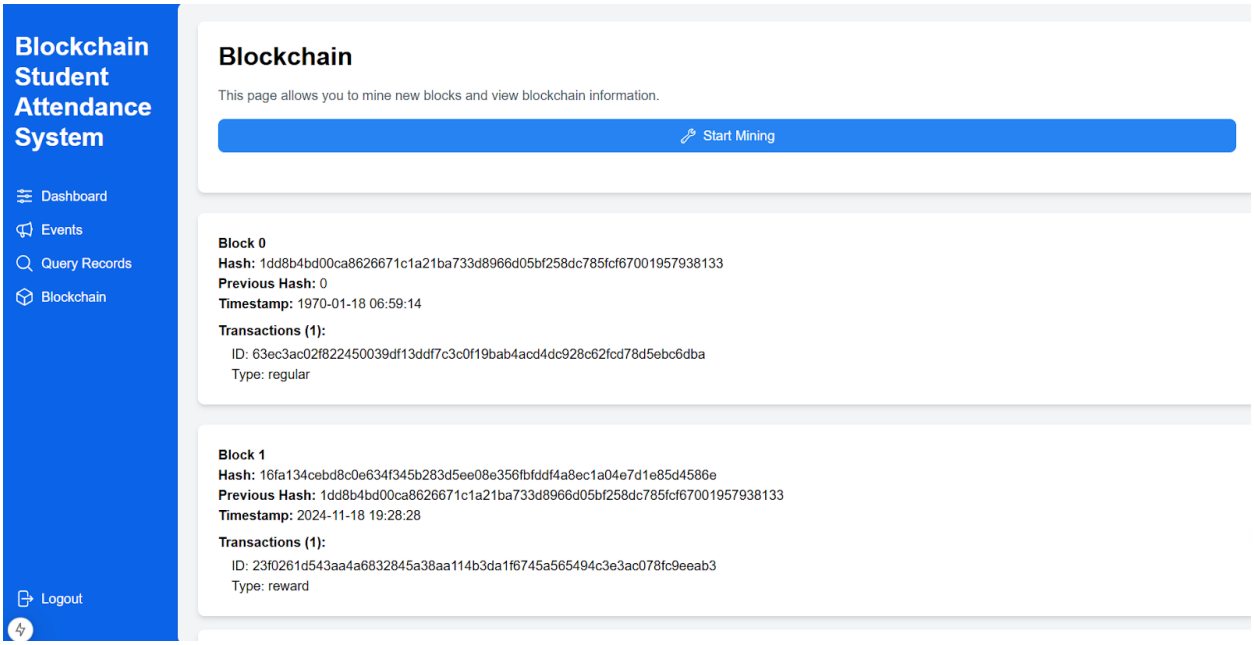


Attendance Enquiry Function Implementation (Lecturer)

Example: Find all student in course COMP5521



User participation in blockchain mining process implementation



Discussion

During the development of this project, our team held multiple discussions regarding the design of the blockchain-based attendance system, focusing on key issues such as system feasibility, efficiency and user experience. Throughout these discussions, we proposed two main design hypotheses and, through comparison and analysis of these two approaches, ultimately determined the system's implementation plan.

Comparison and Analysis of Design Hypotheses

Assumption 1: Coin-Based Check-in Mechanism

In the initial design, we assumed that Coins within the system would have no commercial value, with their primary function being to incentivise students to support blockchain operations through mining activities. The specific design is as follows:

- Students generate blocks through mining to receive Coins upon registration.
- Check-ins are completed through Coin transfer records, which contain the student's ID, event ID (such as course ID), and timestamp.
- After check-in, teachers return the Coins to students for use in subsequent check-ins.

However, this design has significant flaws:

- High complexity: The check-in process involves Coin transfers and returns, increasing system complexity.
- Poor efficiency: Linking mining activities with check-ins may make the process cumbersome due to mining delays.
- Insufficient incentives: Since Coins have no commercial value, students lack motivation to actively participate in mining.

Based on the deficiencies in the aforementioned scheme, after repeated discussions and analysis, we ultimately proposed a second design hypothesis as described below.

Assumption 2: Digital Signature-Based Check-in Mechanism

Through further discussions, we proposed an improved solution that separates the check-in mechanism from coin transfers, replacing it with a digital signature-based approach. The specific design is as follows:

- Students generate public-private key pairs during registration, with the public key serving as identity verification and the private key used for signatures.
- During check-in, students use their private key to sign an attendance certificate (including student ID, event ID, timestamp, etc.), and the signed record is stored in the blockchain.
- Mining activities are independent of check-in, allowing students to decide whether to participate in mining at their discretion.

This improved solution offers the following advantages:

- Simplified Process: Attendance no longer relies on coin transfers and returns, making operations more efficient.
- Time Independence: Even if there are delays in block generation, timestamps still accurately record attendance times without affecting attendance statistics.
- Clear Incentives: Mining activities are separate from attendance, enabling students to decide whether to participate in mining based on their needs.

Key Design Decisions and Considerations

During our discussions, we conducted an in-depth analysis and reached a consensus on several key design issues:

Genesis Block Generation

To resolve the issue of storing the first user's registration information, we decided that the development team would pre-create the genesis block. This approach is a common solution in blockchain systems, ensuring proper system initialisation.

Identity Authentication and Data Storage

Public-private key pairs are used to implement student identity authentication, ensuring the authenticity and immutability of attendance records. Additionally, all attendance data is stored in the blockchain, eliminating dependence on traditional centralised databases.

Mining Incentive Mechanism

Assuming students have an inherent motivation to mine, there is no need to design complex incentive mechanisms. Meanwhile, mining difficulty is dynamically adjusted to maintain system stability and prevent excessive mining.

The Design of Dynamic Difficulty

Dynamic Difficulty Adjustment Mechanism can be divided into the following key logic components:

(1) Trigger Conditions

Difficulty adjustments do not occur with every block, but rather at specific block intervals (e.g., every 10 blocks). This mechanism reduces the overhead of frequent adjustments whilst ensuring smooth blockchain operation.

(2) Comparing Expected and Actual Time

At difficulty adjustment points, the actual time is calculated based on a previous time window—the block interval (for example, the time between the 10th block and the 0th block). By comparing this with the expected time interval, it becomes clear whether the current mining speed meets expectations.

- If the time is too short: This indicates increased system hash power, requiring increased difficulty.
- If the time is too long: This indicates decreased hash power, requiring reduced difficulty.

(3) Adjustment Magnitude and Tolerance Range

To avoid frequent difficulty adjustments or excessive adjustment magnitudes, a tolerance range and maximum adjustment magnitude are designed:

- Tolerance Range: If block generation time is close to the expected range, difficulty remains unchanged at the current level.
- Maximum Adjustment Magnitude: Limits the multiplier for single adjustments to prevent dramatic difficulty changes due to sudden hash power fluctuations.

(4) Implementation of Adjusted Difficulty

The newly calculated difficulty, based on hash power, is applied to subsequent block generation, ensuring mining difficulty dynamically adjusts with network hash power changes.

Basic Fork Resolution

In the original NaiveCoin implementation, fork resolution was primarily based on blockchain length. To enhance system security and decentralization, we'll improve the algorithm to use cumulative block difficulty as the main determinant.

Difficulty Definition: Each block's difficulty = 2^(block difficulty value)

Cumulative Difficulty = Σ(2^(each block's difficulty value))

Steps：

- Receive New Chain: Initiate fork resolution process when receiving a new blockchain or block
- Basic Validation: Check the validity of the new chain's latest block index and verify hash connections between blocks
- Cumulative Difficulty Calculation: Calculate cumulative difficulty of the current main chain, calculate cumulative difficulty of the received new chain and use Σ(2^difficulty) method for calculation

- Chain Replacement Determination:
    - If new chain's cumulative difficulty > current main chain's difficulty:
        - Replace current main chain
        - Ensure the system always maintains the chain with the most proof-of-work
    - If new chain's cumulative difficulty <= current main chain's difficulty/: maintain the current main chain
- Synchronization and Broadcast: After successfully resolving the fork, synchronize the final determined chain to other network nodes and broadcast the latest blockchain state.

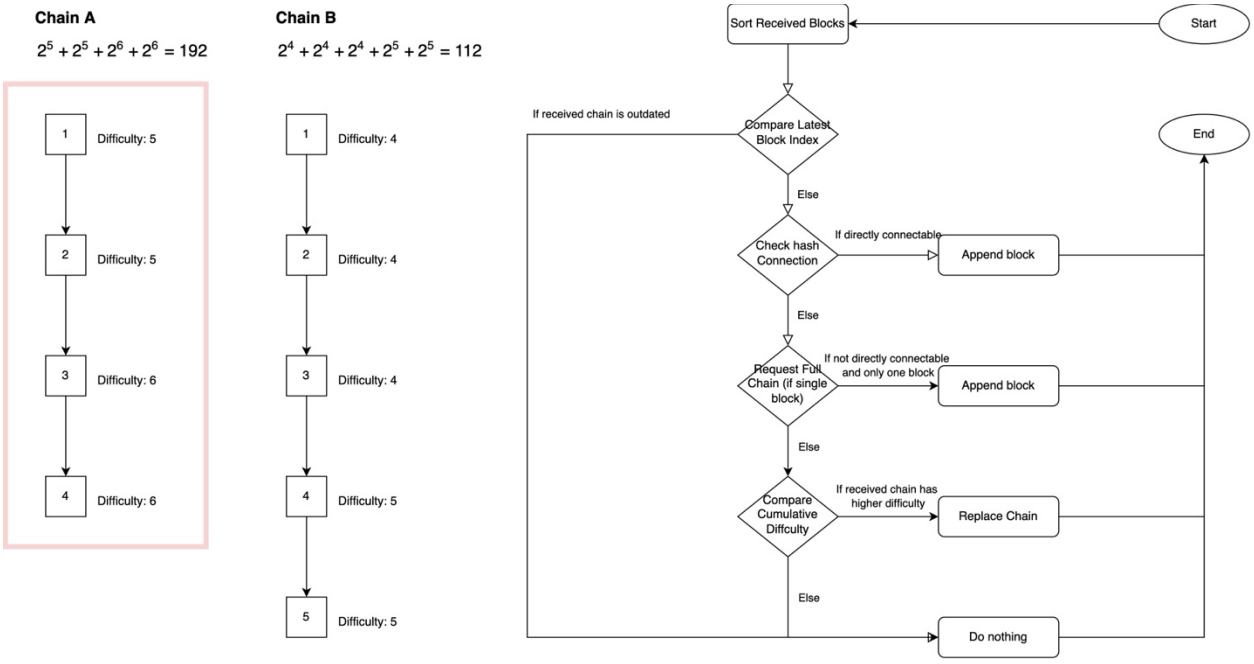


Figure 4 Basic Fork Resolution Process

Query Functionality

The system enables teachers to query student attendance records, displaying student attendance throughout the term or for specific courses, fulfilling the project's basic requirements.

Reference

[1] C. Q. Gomes, "Naivecoin: a tutorial for building a cryptocurrency," GitHub repository, 2017. [Online]. Available: https://github.com/conradoqg/naivecoin

[2] Dunn, J. (2017, October 17). MIT debuts secure digital diploma using Bitcoin blockchain technology. MIT News. https://news.mit.edu/2017/mit-debuts-secure-digital-diploma-using-bitcoin-blockchain-technology-1017

[3] European Commission. (2023). European Blockchain Services Infrastructure (EBSI) - Diplomas and Credentials. Digital Building Blocks. https://ec.europa.eu/digital-building-blocks/sites/display/EBSI/Micro-credentials

[4] M. Arnab, S. Kumar, and R. Singh, "Machine Learning and Blockchain Based Real-Time Attendance Management System," International Organization of Scientific Research Journal of Engineering, vol. 14, no. 4, pp. 61-68, 2021.

[5] Webisoft, "Blockchain for Schools: Enhancing Education Through Decentralized Technology," Webisoft Technical Articles, Aug. 22, 2024. [Online]. Available: https://webisoft.com/articles/blockchain-for-schools/