

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І  
ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ  
Факультет інформаційних технологій**

**ПОГОДЖЕНО**  
**Декан факультету**

\_\_\_\_\_ **Ігор БОЛБОТ**  
(підпис)

**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**  
**Завідувач кафедри інформаційних  
систем і технологій**

\_\_\_\_\_ **Михайло ШВИДЕНКО**  
(підпис)

«\_\_\_\_\_» \_\_\_\_\_ 2026 р.

«\_\_\_\_\_» \_\_\_\_\_ 2026 р.

**БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА**

**на тему: «Блокчейн-орієнтована система контролю успішності студентів»**  
**Спеціальність «126 Інформаційні системи та технології»**  
**Освітньо-професійна програма «Інформаційні системи та технології»**

**Гарант освітньої програми**

к.е.н., доцент

\_\_\_\_\_   
(підпис)

**Максим МОКРІЄВ**

**Керівник бакалаврської  
кваліфікаційної роботи**

\_\_\_\_\_   
(підпис)

**Михайло ШВИДЕНКО**

**Виконав**

\_\_\_\_\_   
(підпис)

**Максим АРТИСЮК**

**Київ-2026**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І  
ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ  
Факультет інформаційних технологій**

**ЗАТВЕРДЖУЮ**

**Завідувач кафедри інформаційних систем і  
технологій**

к.е.н., доцент \_\_\_\_\_ Михайло ШВИДЕНКО  
(підпис)

«\_\_\_\_» \_\_\_\_\_ 2026 р.

**ЗАВДАННЯ  
ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ  
ЗДОБУВАЧУ**

**Артисюку Максиму Артуровичу**

(прізвище, ім'я, по батькові)

Спеціальність «126 Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційні системи та технології»

Тема бакалаврської кваліфікаційної роботи

«Блокчейн-орієнтована система контролю успішності студентів» затверджена наказом від  
«19» грудня 2025 р. № 3205 «С»

Термін подання завершеної роботи на кафедру «23» травня 2026 р.

Вихідні дані до бакалаврської кваліфікаційної роботи: аналіз предметної області, існуючі рішення, структура даних про студентів, результати навчання, ролі користувачів, вимоги до інформаційної безпеки.

Перелік питань, що підлягають дослідженню:

1. Аналіз існуючих систем контролю успішності студентів.
2. Дослідження технології блокчейн, її основних принципів.
3. Проектування архітектури блокчейнорієнтованої системи.
4. Розробка програмної реалізації системи.
5. Оцінка ефективності запропонованого рішення.

Дата видачі завдання «19» грудня 2025 р.

**Керівник бакалаврської  
кваліфікаційної роботи**

(підпис)

**Михайло ШВИДЕНКО**

**Завдання взяв до виконання**

(підпис)

**Максим АРТИСЮК**

## ЗМІСТ

|                                                                            |           |
|----------------------------------------------------------------------------|-----------|
| <b>ВСТУП.....</b>                                                          | <b>5</b>  |
| <b>РОЗДІЛ 1 .....</b>                                                      | <b>7</b>  |
| <b>АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ .....</b>                | <b>7</b>  |
| 1.1 Аналіз існуючих систем контролю успішності студентів .....             | 7         |
| Таблиця 1.1 — Порівняльний аналіз існуючих систем контролю успішності..... | 7         |
| 1.2 Огляд наукових досліджень у сфері блокчейн в освіті.....               | 8         |
| 1.3 Обґрунтування вибору теми та технологічного стека .....                | 9         |
| 1.4 Мета, об'єкт, предмет та завдання дослідження.....                     | 10        |
| 1.5 Висновки до першого розділу .....                                      | 10        |
| <b>РОЗДІЛ 2 .....</b>                                                      | <b>11</b> |
| <b>ПРОЕКТУВАННЯ БЛОКЧЕЙН-ОРІЄНТОВАНОЇ СИСТЕМИ.....</b>                     | <b>11</b> |
| <b>КОНТРОЛЮ УСПІШНОСТІ СТУДЕНТІВ .....</b>                                 | <b>11</b> |
| 2.1 Архітектура системи та вибір проектних рішень.....                     | 11        |
| 2.2 Структурна схема системи та її основні модулі .....                    | 11        |
| 2.3 Проектування бази даних .....                                          | 12        |
| Таблиця 2.1 — Структура блоку блокчейн-ланцюга .....                       | 13        |
| 2.4 Проектування REST API.....                                             | 13        |
| Таблиця 2.2 — REST API маршрути системи .....                              | 13        |
| 2.5 Проектування блокчейн-механізму .....                                  | 14        |
| 2.6 Проектування людино-машинного інтерфейсу.....                          | 15        |
| 2.7 Вибір та обґрунтування технологій реалізації.....                      | 15        |
| Таблиця 2.3 — Порівняння та обґрунтування вибору технологій.....           | 15        |
| 2.8 Проектування системи журналювання та аудиту .....                      | 16        |
| 2.9 Висновки до другого розділу .....                                      | 16        |
| <b>РОЗДІЛ 3 .....</b>                                                      | <b>18</b> |
| <b>РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ БЛОКЧЕЙН-ОРІЄНТОВАНОЇ СИСТЕМИ</b>              |           |
| <b>КОНТРОЛЮ УСПІШНОСТІ СТУДЕНТІВ .....</b>                                 | <b>18</b> |
| 3.1 Структура проекту та файлова організація .....                         | 18        |
| 3.2 Програмна реалізація серверної частини системи .....                   | 19        |
| 3.3 Реалізація блокчейн-модуля .....                                       | 21        |
| 3.4 Реалізація клієнтського інтерфейсу та модуля захисту ролей .....       | 23        |
| 3.5 Демонстрація функціональних можливостей системи .....                  | 23        |
| 3.6 Технічні характеристики розробленої системи .....                      | 33        |
| 3.7 Тестування системи.....                                                | 34        |
| 3.8 Аналіз результатів тестування .....                                    | 35        |

|                                         |           |
|-----------------------------------------|-----------|
| 3.9 Аналіз безпеки системи .....        | 36        |
| 3.10 Висновки до третього розділу ..... | 36        |
| <b>ВИСНОВКИ.....</b>                    | <b>38</b> |
| <b>СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....</b>  | <b>40</b> |
| <b>ДОДАТКИ.....</b>                     | <b>42</b> |

## ВСТУП

Стрімкий розвиток інформаційних технологій та цифрове перетворення освітнього процесу ставлять нові вимоги до систем обліку академічних досягнень студентів. Традиційні підходи, ґрунтовані на централізованих базах даних з рівневим адміністративним доступом, не забезпечують належного рівня прозорості та захищеності даних: оцінки можуть бути змінені адміністраторами системи без будь-якого сліду, а студенти позбавлені можливості незалежно верифікувати власні академічні записи.

Технологія блокчейн, що набула широкого поширення за межами грошового сектора, пропонує принципово новий підхід до забезпечення цілісності даних. Розподілена структура блокчейну, криптографічна прив'язка блоків та незмінність ланцюга роблять її перспективним інструментом для застосування в освітній галузі.

Метою бакалаврської кваліфікаційної роботи є проектування та розробка блокчейн-орієнтованої інформаційної системи контролю успішності студентів, яка забезпечує незмінність, прозорість і криптографічний захист академічних даних.

Для досягнення поставленої мети визначено такі завдання дослідження:

- проаналізувати існуючі системи контролю успішності студентів та виявити їх недоліки;
- дослідити основні принципи роботи технологій блокчейн та можливості її застосування в освітній сфері;
- архітектуру блокчейн-орієнтованої спроектувати системи контролю успішності студентів;
- розробити серверну і клієнтську частини системи з використанням сучасного технологічного сукупності технологій (Node.js, Express, MongoDB);

- реалізувати механізм ланцюга елементів для зберігання оцінок із забезпеченням цілісності даних;
- розробити зручний веб інтерфейс для двох ролей користувачів — викладача (адміністратора) і студента;
- протестувати систему та оцінити ефективність застосованого підходу.

Об'єктом дослідження є процес обліку та контролю навчальних досягнень студентів у закладах вищої освіти.

Предметом дослідження є методи і засоби застосування технології блокчейн для забезпечення цілісності та захищеності даних в інформаційних системах освіти.

Практична стаття роботи полягає у розробці повноцінної інноваційної веб систем блокчейн-архітектурою, яка може бути впроваджена у навчальних закладах для підвищення захищеності та прозорості академічних даних.

Структура роботи. Бакалаврська кваліфікаційна робота складається зі вступу, трьох розділів основної частини, висновків, списку використаних джерел та додатків.

## РОЗДІЛ 1

### АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

#### 1.1 Аналіз існуючих систем контролю успішності студентів

Система контролю успішності студентів є невід'ємною складовою освітнього процесу будь-якого вищого навчального закладу. Традиційні підходи до ведення академічних журналів і виставлення оцінок, що базуються на паперових носіях або централізованих базах даних реляційного типу, мають ряд суттєвих недоліків, які знижують ефективність та надійність обліку навчальних досягнень студентів.

Серед найбільш поширених проблем існуючих систем слід виокремити такі:

- не захищеність від несанкціонованої модифікації — оцінки можуть бути змінені або вилучені адміністраторами системи без жодного сліду;
- відсутність прозорості — студенти не мають можливості незалежно переглянути достовірність виставлених оцінок;
- централізоване зберігання даних, що створює єдину точку відмови;
- недостатній рівень захисту від підробки академічних документів;
- складність перевірки — зворотна перевірка змін в оцінках потребує значних зусиль і не завжди можлива.

Аналіз існуючих програмних рішень — таких як Moodle, Google Classroom, університетські інформаційні системи (АСУ ВНЗ) — показує, що всі вони засновані на класичній архітектурі клієнт–сервер із централізованим зберіганням даних. Це означає, що довіра до системи повністю залежить від довіри до її керівників, а будь-яке порушення цілісності даних може залишитися непоміченим.

**Таблиця 1.1 — Порівняльний аналіз існуючих систем контролю успішності**

| Система            | Тип архітектури          | Захист від модиф. | Прозорість | Аудит       | Блокчейн |
|--------------------|--------------------------|-------------------|------------|-------------|----------|
| Moodle             | Централізована           | Низький           | Відсутня   | Обмежений   | Ні       |
| Google Classroom   | Хмарна<br>централізована | Середній          | Відсутня   | Обмежений   | Ні       |
| АСУ ВНЗ            | Централізована           | Низький           | Відсутня   | Складний    | Ні       |
| EduCTX             | Блокчейн (Ethereum)      | Високий           | Повна      | Криптограф. | Так      |
| Розроблена система | Гібридна +<br>Blockchain | Високий           | Повна      | Дворівневий | Так      |

Як видно з таблиці 1.1, жодна з поширених комерційних систем не забезпечує криптографічного захисту академічних даних, тоді як розроблена система поєднує переваги змінного зберігання та власного блокчейн-механізму.

## 1.2 Огляд наукових досліджень у сфері блокчейн в освіті

Питання застосування технології блокчейн в освітній галузі є відносно новим, але активно розвивається у науковій спільноті. Перші систематичні дослідження у цій сфері з'явилися після 2016 року, коли стало очевидним, що блокчейн має потенціал поза фінансовим сектором.

Grech та Camilleri (2017) у фундаментальному звіті «Blockchain in Education» для Євросоюзу виокремили вісім сценаріїв застосування блокчейну в освіті: підтвердження досягнень, захист авторських прав на навчальний контент, студентські платежі, прийом до закладу, вільний вибір навчальних ресурсів, захищена особистісна ідентичність, портативність документів про освіту та верифікація педагогічних кваліфікацій. Серед цих сценаріїв захист і верифікація академічних досягнень посідають центральне місце.

Chen та ін. (2018) провели огляд перспектив застосування блокчейну в розумному навчальному середовищі. Автори підкреслюють, що децентралізація даних про успішність підвищує довіру між учасниками освітнього процесу та знижує витрати на адміністрування. Дослідники також звертають увагу на

необхідність балансу між прозорістю (яку забезпечує блокчейн) і конфіденційністю персональних даних студентів (яка регулюється GDPR та аналогічними нормами).

Lizcano та ін. (2020) розробили блокчейн-орієнтовану модель довіри для відкритої та повсюдної вищої освіти. Ключовим дослідження є те, що власна реалізація блокчейну, на відміну від публічних мереж, є більш придатною для освітніх закладів з точки зору масштабованості, контролю доступу та мінімізації витрат.

Ці дослідження підтверджують доцільність обраного підходу: власна реалізація блокчейну без сторонніх залежностей, вбудована у навчальну інформаційну систему, є оптимальним рішенням для захисту академічних даних на рівні окремого закладу.

### **1.3 Обґрунтування вибору теми та технологічного стека**

Тема бакалаврської кваліфікаційної роботи обрана відповідно до актуальних тенденцій розвитку інформаційних систем та технологій у сфері освіти. Вибір теми зумовлений кількома чинниками.

По-перше, стрімкий розвиток технології блокчейн і розширення сфер її практичного застосування за межі фінансового сектора відкриває нові можливості для інформаційних систем освіти. По-друге, зростаючі вимоги до прозорості та захищеності академічних даних формують запит на надійні технологічні рішення. По-третє, тема відповідає пріоритетним напрямкам досліджень кафедри інформаційних систем і технологій.

Для реалізації системи обрано сучасний стек технологій:

- Node.js 20 LTS — середовище виконання JavaScript на сервері з подієво-орієнтовано блокуючою архітектурою;
- Express.js 5 — мінімалістичний фреймворк для маршрутизації HTTP-запитів та побудови REST API;

- MongoDB — документоорієнтована база даних NoSQL з гнучкою схемою документів;
- SQLite — вбудована реляційна база даних без зовнішнього сервера для структурованих допоміжних даних;
- Власна реалізація блокчейну на JavaScript з SHA-256 кешуванням та алгоритмом Proof of Work;
- PDFKit — бібліотека для серверної генерації PDF-звітів без браузерного рушія;
- вбудований JavaScript/HTML5/CSS3 без важких фронтенд-фреймворків.

#### **1.4 Мета, об'єкт, предмет та завдання дослідження**

Метою бакалаврської кваліфікаційної роботи є проектування та розробка блокчейн-орієнтованої інформаційної системи контролю успішності студентів, яка дає незмінність, прозорість і криптографічний захист академічних даних.

Об'єктом дослідження є процес обліку та контролю навчальних досягнень студентів у закладах вищої освіти.

Предметом дослідження є методи і засоби застосування технології блокчейн для забезпечення цілісності та захищеності даних в інформаційних системах освіти.

#### **1.5 Висновки до першого розділу**

У першому розділі проведено аналіз предметної області та обґрунтовано актуальність теми бакалаврської кваліфікаційної роботи. Встановлено, що існуючі системи контролю успішності студентів мають недоліки з точки зору захищеності та прозорості даних. Здійснено огляд наукових праць, що підтвердив доречність власної реалізації блокчейну. Обґрунтовано вибір технологічного стека розробки.

## **РОЗДІЛ 2**

### **ПРОЕКТУВАННЯ БЛОКЧЕЙН-ОРІЄНТОВАНОЇ СИСТЕМИ**

#### **КОНТРОЛЮ УСПІШНОСТІ СТУДЕНТІВ**

##### **2.1 Архітектура системи та вибір проектних рішень**

Систему спроектовано за трирівневою клієнт-серверною структурою, що включає рівень представлення (клієнтський інтерфейс), рівень бізнес-логіки (серверна частина на Node.js/Express) та рівень збереження даних (MongoDB та SQLite та блокчейн-ланцюг у пам'яті з довготривалим JSON-файлом).

Такий підхід забезпечує чітке розмежування відповідальності між компонентами, спрощує супровід і дозволяє незалежно масштабувати кожен рівень. Клієнтська частина реалізована на нативному HTML/CSS/JavaScript без важких фронтенд фреймворків, що зменшує кількість залежностей і пришвидшує завантаження сторінок.

##### **2.2 Структурна схема системи та її основні модулі**

Система складається з таких основних програмних модулів:

Модуль автентифікації (login/session) — відповідає за перевірку облікових даних користувача, створення та знищення сесій, а також за захист маршрутів залежно від ролі.

Модуль управління оцінками (grades) — забезпечує додавання, перегляд та видалення записів про оцінки; виконує синхронізацію між базою даних MongoDB і блокчейн-ланцюгом.

Блокчейн-модуль (blockchain.js) — реалізує власну імплементацію ланцюга блоків з алгоритмом Proof of Work; містить класи Block і Blockchain з методами addBlock(), isChainValid(), exportChain() та getBlockchainStats().

Модуль бази даних SQLite (db.js) — зберігає структуровані дані про студентів і оцінки у реляційній таблиці students для швидкого пошуку і фільтрації.

Модуль Grade (Grade.js/Mongoose) — описує схему документів оцінок у MongoDB.

Модуль журналювання (logs) — фіксує всі дії в системі у JSON-файл logs.json з відміткою часу та іменем користувача.

Модуль захисту ролей (role-guard.js) — клієнтський JavaScript-модуль, що одразу приховує елементи з класом admin-only, а після запиту /api/me або показує їх (для адміністратора), або перенаправляє студента.

Модуль генерації PDF (pdfkit) — дозволяє адміністратору експортувати журнал оцінок або окремий звіт у формат PDF безпосередньо із системи.

## 2.3 Проектування бази даних

У системі застосовано гібридний підхід до зберігання даних, що поєднує дві бази даних із різними характеристиками: документоорієнтовану MongoDB та реляційну SQLite.

MongoDB використовується як основне сховище оцінок. Схема документа Grade містить такі поля: student (рядок, обов'язкове), subject (рядок, обов'язкове), grade (число, обов'язкове, діапазон 60–100), date (рядок або дата).

SQLite застосовується для зберігання допоміжних структурованих даних у таблиці students. Облікові записи користувачів та перелік дисциплін зберігаються у JSON-файлах (users.json, courses.json, students.json, course\_students.json).

Блокчейн-ланцюг зберігається у файлі studentchain.json і завантажується в пам'ять при запуску сервера. Кожен блок містить поля: index, timestamp, data, previousHash, hash, nonce.

**Таблиця 2.1 — Структура блоку блокчейн-ланцюга**

| Поле         | Тип даних       | Опис                                                          |
|--------------|-----------------|---------------------------------------------------------------|
| index        | Integer         | Порядковий номер блоку в ланцюзі                              |
| timestamp    | ISO 8601 String | Час створення блоку                                           |
| data         | JSON Object     | Корисне навантаження: об'єкт оцінки (student, subject, grade) |
| previousHash | SHA-256 hex     | Хеш попереднього блоку ("0" для genesis)                      |
| nonce        | Integer         | Лічильник для алгоритму Proof of Work                         |
| hash         | SHA-256 hex     | SHA-256 хеш усіх полів блоку + nonce                          |

## 2.4 Проектування REST API

Серверна частина системи надає набір HTTP-маршрутів, організованих за принципом REST. Маршрути поділяються на публічні (доступні без автентифікації) та захищені (потребують активної сесії та відповідної ролі).

**Таблиця 2.2 — REST API маршрути системи**

| Метод | Маршрут        | Доступ    | Опис                                             |
|-------|----------------|-----------|--------------------------------------------------|
| GET   | /login         | Публічний | Відображення сторінки входу                      |
| POST  | /login         | Публічний | Перевірка облікових даних, створення сесії       |
| GET   | /logout        | Авториз.  | Знищення сесії, перенаправлення                  |
| GET   | /api/me        | Авториз.  | JSON: ім'я, роль, username поточного користувача |
| GET   | /api/my-grades | Авториз.  | Список оцінок (студент — лише власні)            |
| POST  | /api/grades    | Адмін     | Додавання нової оцінки + запис блоку в блокчейн  |

|        |                       |          |                                          |
|--------|-----------------------|----------|------------------------------------------|
| DELETE | /api/grades/:id       | Адмін    | Видалення оцінки з MongoDB               |
| GET    | /api/blockchain       | Адмін    | Весь ланцюг блоків у форматі JSON        |
| GET    | /api/blockchain/stats | Адмін    | Статистика: кількість блоків, валідність |
| GET    | /api/logs             | Адмін    | Журнал дій користувачів                  |
| GET    | /api/export/pdf       | Адмін    | Генерація та завантаження PDF-звіту      |
| GET    | /api/courses          | Авториз. | Перелік доступних дисциплін              |
| GET    | /api/students         | Адмін    | Перелік студентів у системі              |

## 2.5 Проектування блокчейн-механізму

Блокчейн-модуль є ключовим елементом системи, що забезпечує незмінність та верифікованість академічних записів. Модуль реалізовано самостійно на JavaScript без залежностей від сторонніх бібліотек блокчейну.

Клас Block містить такі атрибути: index (порядковий номер), timestamp (час створення у форматі ISO 8601), data (об'єкт оцінки), previousHash (хеш попереднього блоку), nonce (лічильник для PoW), hash (SHA-256 хеш поточного блоку).

Метод calculateHash() обчислює хеш шляхом конкатенації всіх полів блоку та передачі рядка до крипто-хешера: `crypto.createHash('sha256').update(...).digest('hex')`. Будь-яка зміна даних у блоці призводить до зміни хешу.

Метод mineBlock(difficulty) реалізує алгоритм Proof of Work: nonce інкрементується до тих пір, поки перші difficulty символів хешу не дорівнюватимуть нулям. При складності difficulty=1 це забезпечує достатній рівень захисту при мінімальній затримці.

Метод isChainValid() перевіряє цілісність всього ланцюга: для кожного блоку починаючи з індексу 1 перевіряється, що поточний хеш відповідає

обчисленому, `previousHash` збігається з кешем попереднього блоку, а кеш відповідає умові Proof of Work.

## 2.6 Проектування людино-машинного інтерфейсу

Інтерфейс системи розроблено для двох категорій користувачів: адміністратора (викладача) та студента. Обидві ролі мають спільну базу UI, проте набір доступних елементів та сторінок різниться відповідно до прав доступу.

Дизайн-система базується на унікальних CSS-змінних, що задають єдину кольорову палітру для всього застосунку. Основні кольори: `--mint` (`#7FFFD4`) та `--teal` (`#5F9EA0`) для акцентів, `--bg` (`#EEF8F8`) для фону сторінки, `--surface` (`#FFFFFF`) для карток. Шрифт Plus Jakarta Sans (Google Fonts) використовується для тексту, JetBrains Mono — для відображення хешів і технічних кодів.

Навігація між розділами реалізована через одну HTML-сторінку з динамічним вмістом (SPA-підхід без фреймворків): JavaScript приховує/показує секції та відвантажує дані через `fetch`-запити до API.

## 2.7 Вибір та обґрунтування технологій реалізації

**Таблиця 2.3 — Порівняння та обґрунтування вибору технологій**

| Компонент            | Обраний варіант    | Альтернатива | Обґрунтування                                                            |
|----------------------|--------------------|--------------|--------------------------------------------------------------------------|
| Середовище виконання | Node.js 20 LTS     | Python/Flask | Єдина мова (JS) для сервера і блокчейн-модуля; висока продуктивність I/O |
| Веб-фреймворк        | Express.js 5       | Fastify, Koa | Широка екосистема; мінімалістичність; велика спільнота                   |
| Основна БД           | MongoDB (Mongoose) | PostgreSQL   | Гнучка документна модель; легке розширення схеми без міграцій            |

|                |                      |                      |                                                                      |
|----------------|----------------------|----------------------|----------------------------------------------------------------------|
| Допоміжна БД   | SQLite               | MySQL                | Вбудована БД без зовнішнього сервера; достатня для локального пошуку |
| Блокчейн       | Власна реалізація    | Ethereum/Hyperledger | Повний контроль над логікою; відсутність зовнішніх залежностей       |
| Генерація PDF  | PDFKit               | Puppeteer, jsPDF     | Серверна генерація без браузера; баовий API для таблиць              |
| Клієнтський UI | Нативний JS/HTML/CSS | React, Vue           | Мінімальні залежності; швидке завантаження                           |
| Автентифікація | express-session      | JWT                  | Серверні сесії простіші для монолітного застосунку                   |

## 2.8 Проектування системи журналювання та аудиту

Журналювання є важливою складовою системи, оскільки дозволяє відстежувати всі дії користувачів і забезпечує можливість аудиту. У системі реалізовано два незалежних рівні аудиту.

Перший рівень — журнал дій (logs.json). Кожна значуща операція записується у JSON-файл у вигляді об'єкта з полями: timestamp, action, username, details.

Другий рівень — блокчейн як незмінний аудит-трейл оцінок. На відміну від звичайного журналу, запис в блокчейні не може бути тихо видалений або змінений без порушення цілісності ланцюга.

## 2.9 Висновки до другого розділу

У другому розділі виконано повна розробка блокчейн-орієнтованої системи контролю успішності студентів. Система розроблена за трирівневою клієнт-серверною архітектурою з вісьми програмними модулями. Розроблено схему гібридного зберігання даних. Спроектowano REST API з рольовою авторизацією. Детально описано архітектуру блокчейн-модуля та обґрунтовано вибір стека технологій.



## РОЗДІЛ 3

# РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ БЛОКЧЕЙН-ОРІЄНТОВАНОЇ СИСТЕМИ КОНТРОЛЮ УСПІШНОСТІ СТУДЕНТІВ

### 3.1 Структура проекту та файлова організація

Проект організовано у відповідності до стандартних конвенцій Node.js-застосунків. Кореневий каталог містить такі директорії та файли:

- `blockchain/` — директорія блокчейн-модуля (`blockchain.js`);
- `database/` — модулі роботи з базами даних;
- `models/` — Mongoose-схеми (`Grade.js`);
- `public/` — статичні файли клієнтської частини (`index.html`, `login.html`, `role-guard.js`);
- `node_modules/` — залежності npm;
- `server.js` — точка входу серверного застосунку;
- `seed.js` — скрипт початкового наповнення бази даних;
- `courses.json`, `students.json`, `users.json` — конфігураційні JSON-файли;
- `course_students.json` — відображення студентів на курси;
- `data.json`, `studentchain.json` — персистентні файли блокчейн-ланцюга;
- `logs.json` — журнал дій;
- `package.json`, `package-lock.json` — конфігурація npm.

|                        |                  |                    |        |
|------------------------|------------------|--------------------|--------|
| 📁 blockchain           | 09.05.2026 18:43 | Папка с файлами    |        |
| 📁 database             | 09.05.2026 19:39 | Папка с файлами    |        |
| 📁 models               | 12.05.2026 1:45  | Папка с файлами    |        |
| 📁 node_modules         | 12.05.2026 1:44  | Папка с файлами    |        |
| 📁 public               | 17.05.2026 23:46 | Папка с файлами    |        |
| 📄 course_students.json | 22.05.2026 11:34 | Исходный файл J... | 2 КБ   |
| 📄 courses.json         | 13.05.2026 4:35  | Исходный файл J... | 1 КБ   |
| 📄 data.json            | 22.05.2026 11:36 | Исходный файл J... | 108 КБ |
| 📄 logs.json            | 22.05.2026 11:37 | Исходный файл J... | 27 КБ  |
| 📄 package.json         | 12.05.2026 1:44  | Исходный файл J... | 1 КБ   |
| 📄 package-lock.json    | 12.05.2026 1:48  | Исходный файл J... | 70 КБ  |
| 📄 seed.js              | 12.05.2026 2:18  | Исходный файл J... | 2 КБ   |
| 📄 server.js            | 17.05.2026 22:40 | Исходный файл J... | 58 КБ  |
| 📄 studentchain.json    | 22.05.2026 11:36 | Исходный файл J... | 356 КБ |
| 📄 students.json        | 12.05.2026 14:20 | Исходный файл J... | 1 КБ   |
| 📄 users.json           | 17.05.2026 23:50 | Исходный файл J... | 2 КБ   |

Рисунок 3.1 — Файлова структура проекту

### 3.2 Програмна реалізація серверної частини системи

Серверну частину системи реалізовано на платформі Node.js з використанням фреймворку Express.js версії 5. Точкою входу є файл `server.js`, який ініціалізує веб сервер, підключає базу даних MongoDB, завантажує блокчейн-ланцюг із файлу та реєструє всі HTTP-маршрути системи.

При запуску сервер виконує такі ініціалізаційні кроки: зчитує файли `users.json`, `course_students.json` та `students.json` у пам'ять; підключається до MongoDB; ініціалізує об'єкт блокчейну класу `Blockchain` та завантажує збережені блоки з файлу `data.json`. Якщо даних немає — автоматично генерує демонстраційний набір з 20 студентів і по 5 оцінок кожному.

#### Лістинг 3.1 — Ініціалізація блокчейн-ланцюга при запуску сервера

```
const studentBlockchain = new Blockchain();
```

```

studentBlockchain.chain = [studentBlockchain.chain[0]];

const DATA_FILE = 'data.json';

if (fs.existsSync(DATA_FILE)) {
  const data = fs.readFileSync(DATA_FILE);
  const parsed = JSON.parse(data);
  parsed.forEach(item => studentBlockchain.addBlock(item));
}

```

Захист маршрутів реалізовано через глобальний middleware, який перехоплює кожен вхідний запит та перевіряє наявність активної сесії. Обробку становлять лише маршрути /login та /role-guard.js.

### Лістинг 3.2 — Middleware перевірки аутентифікації

```

app.use((req, res, next) => {
  const open = ['/login', '/role-guard.js'];
  if (open.some(p => req.path.startsWith(p))) return next();
  if (!req.session.loggedIn) {
    if (req.path.startsWith('/api/')) {
      return res.status(401).json({ error: 'Не авторизовано' });
    }
    return res.redirect('/login');
  }
  next();
});

```

Функція addLog() записує в файл logs.json кожен значущий операцію в системі. Структура запису містить поля: action (опис дії), user (логін виконавця), date (рядок з локальним часом).

### Лістинг 3.3 — Функція журналювання дій

```

function addLog(action, user) {

```

```

let logs = [];
if (fs.existsSync(LOG_FILE)) {
  logs = JSON.parse(fs.readFileSync(LOG_FILE));
}
logs.push({
  action,
  user,
  date: new Date().toLocaleString()
});
fs.writeFileSync(LOG_FILE, JSON.stringify(logs, null, 2));
}

```

### 3.3 Реалізація блокчейн-модуля

Блокчейн-модуль реалізовано у файлі `blockchain/blockchain.js` і складається з двох класів: `Block` та `Blockchain`. Клас `Block` описує одиничний блок ланцюга, клас `Blockchain` керує ланцюгом у цілому.

#### Лістинг 3.4 — Метод обчислення SHA-256 хешу блоку

```

calculateHash() {
  return crypto
    .createHash('sha256')
    .update(
      this.index +
      this.previousHash +
      this.timestamp +
      JSON.stringify(this.data) +
      this.nonce
    )
    .digest('hex');
}

```

Алгоритм Proof of Work реалізовано в методі `mineBlock(difficulty)`. Значення `nonce` інкрементується в циклі до тих пір, поки перші `difficulty`

символів хешу не стануть нульовими. При поточній складності `difficulty=1` пошук займає одиниці мілісекунд.

### Лістинг 3.5 — Алгоритм Proof of Work

```
mineBlock(difficulty) {
  const target = Array(difficulty + 1).join('0');
  while (this.hash.substring(0, difficulty) !== target) {
    this.nonce++;
    this.hash = this.calculateHash();
  }
  console.log('Block mined:', this.hash);
}
```

Метод `isChainValid()` виконує повну перевірку цілісності ланцюга, перебираючи кожен блок починаючи з індексу один. Для кожного блоку перевіряються три умови: відповідність збереженого хешу обчисленому, збіг `previousHash` з хешем попереднього блоку, відповідність кешу умові Proof of Work.

### Лістинг 3.6 — Метод верифікації ланцюга `isChainValid()`

```
isChainValid() {
  for (let i = 1; i < this.chain.length; i++) {
    const cur = this.chain[i];
    const prev = this.chain[i - 1];
    if (cur.hash !== cur.calculateHash()) return false;
    if (cur.previousHash !== prev.hash) return false;
    const target = Array(this.difficulty + 1).join('0');
    if (cur.hash.substring(0, this.difficulty) !== target) {
      return false;
    }
  }
  return true;
}
```

### 3.4 Реалізація клієнтського інтерфейсу та модуля захисту ролей

Клієнтська частина системи реалізована засобами нативного HTML5, CSS3 та JavaScript без фронтенд-фреймворків. Головна сторінка (index.html) завантажується як єдиний HTML-документ, а зміна розділів відбувається динамічно.

Модуль role-guard.js реалізує клієнтський захист прав доступу у два кроки. На першому кроці синхронно ін'єктується CSS-правило, що негайно приховує всі елементи адміністратора. На другому кроці надсилається асинхронний запит до /api/me — якщо роль = admin, приховування знімається; якщо student — виконується перенаправлення з захищених сторінок.

#### Лістинг 3.7 — Синхронне приховування адмін-елементів у role-guard.js

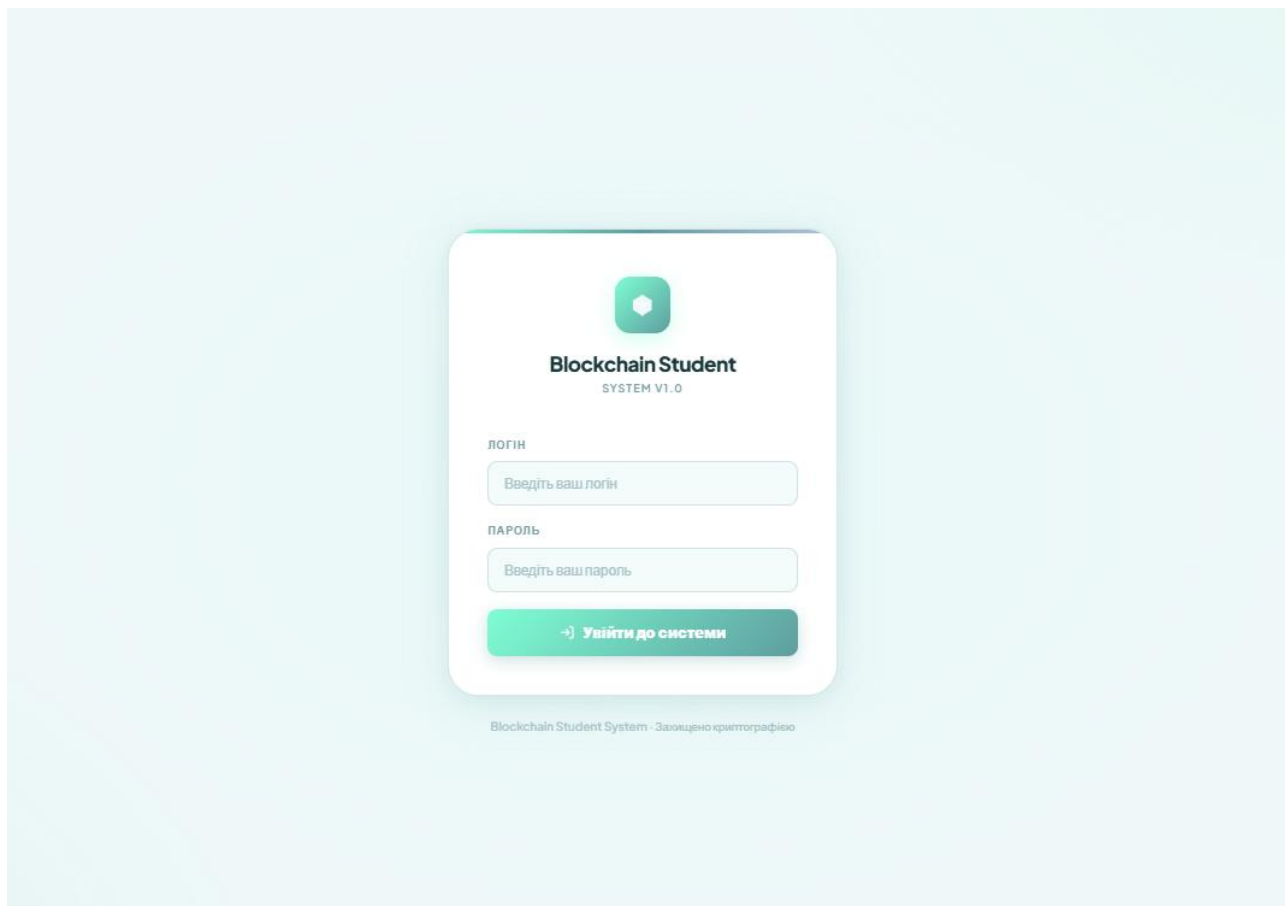
```
(function injectHideStyle() {
  const style = document.createElement('style');
  style.id = 'role-guard-style';
  style.textContent =
    '.admin-only { display: none !important; }';
  const target = document.head || document.documentElement;
  target.insertBefore(style, target.firstChild);
})();
```

Клієнт використовує fetch-API для асинхронного завантаження даних з REST API та рендерингу їх у DOM без перезавантаження сторінки. Кожна секція застосунку відповідає окремому API-маршруту.

### 3.5 Демонстрація функціональних можливостей системи

На рисунку 3.2 представлено сторінку входу до системи. Форма містить поля «Логін» та «Пароль», кнопку «Увійти до системи» та анімований фон з

плаваючими геометричними фігурами. Перевірка облікових даних відбувається на сервері шляхом пошуку в масиві `users.json`.



*Рисунок 3.2 — Сторінка входу до системи (login.html)*

На рисунку 3.3 показано панель керування адміністратора. Вона відображає загальну статистику (кількість студентів, курсів, оцінок, захищених блокчейном), список усіх доступних курсів у вигляді карток та форму додавання нового студента. Кнопка «Створити курс» відкриває модальне вікно для введення назви нового курсу.

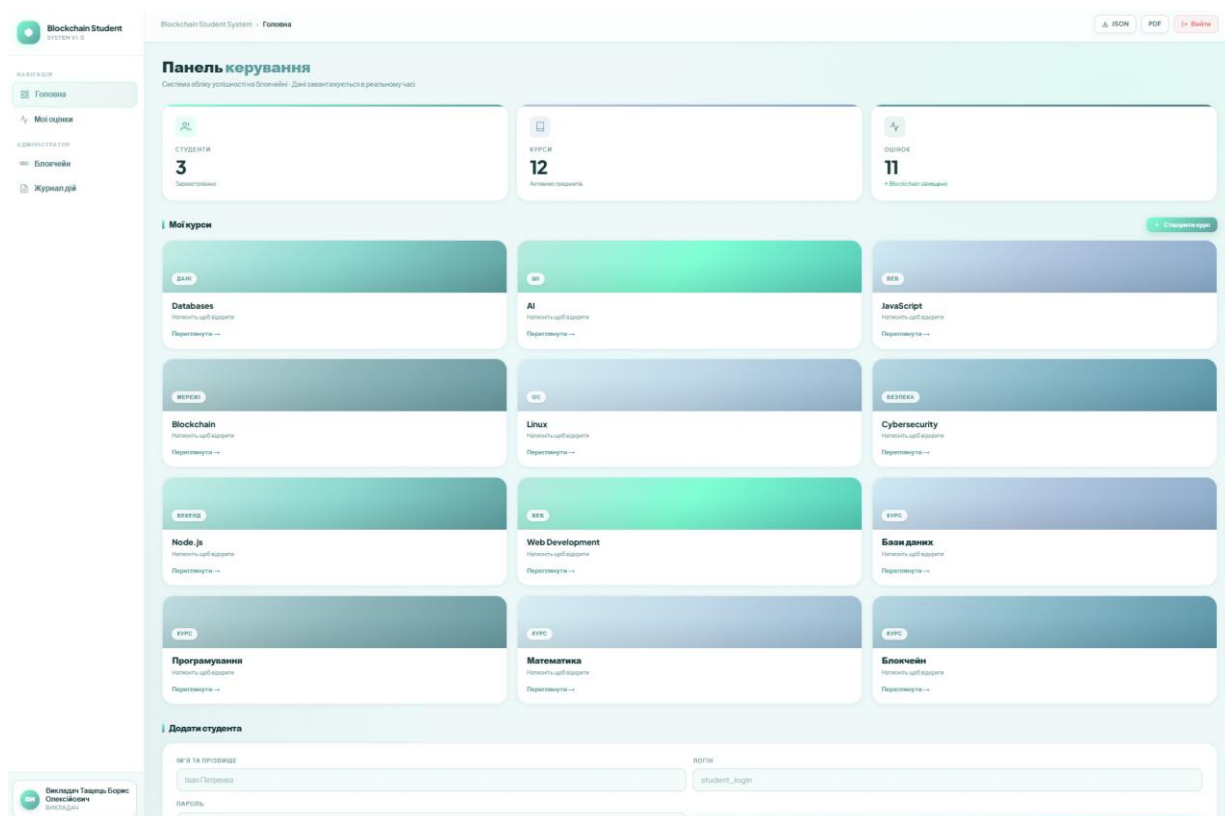
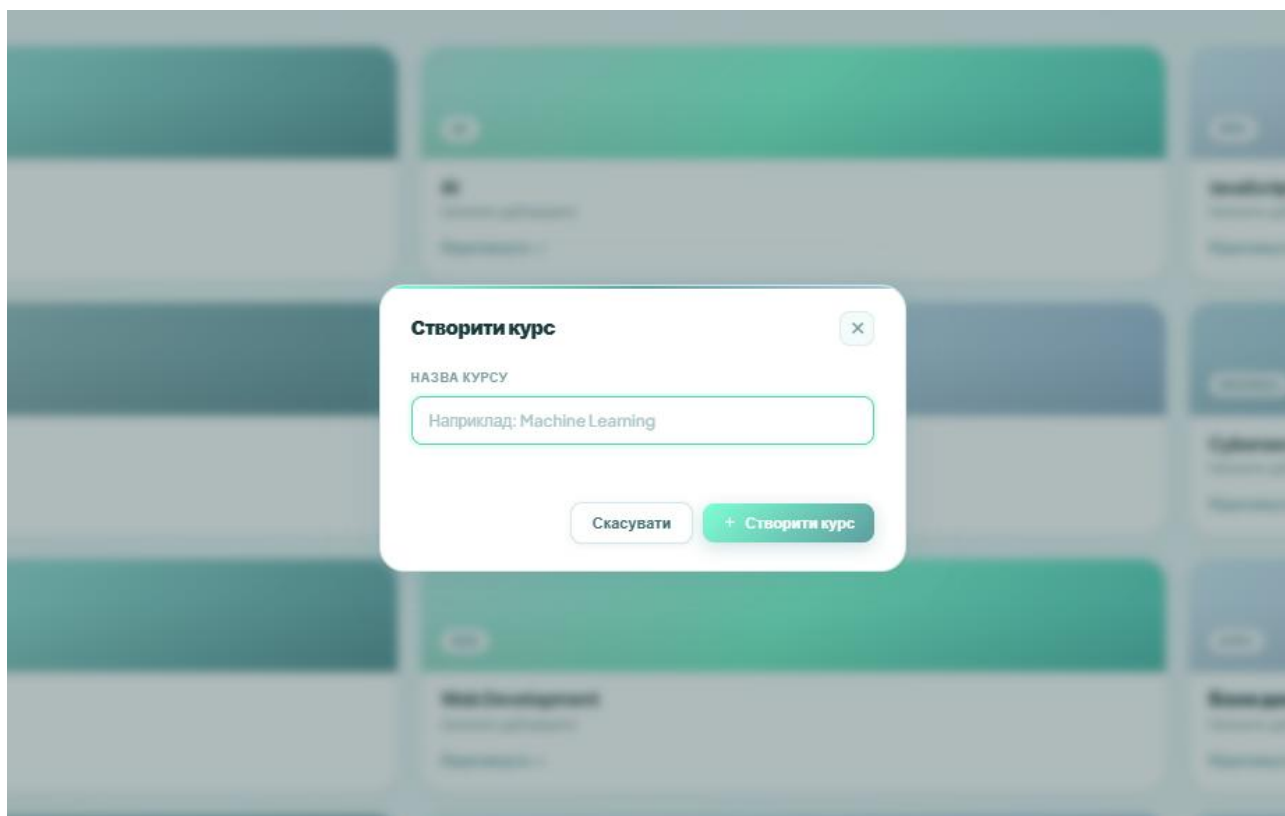


Рисунок 3.3 — Панель керування адміністратора (головна сторінка)

На рисунку 3.4 показано панель керування з відкритим модальним вікном «Створити курс», що демонструє зручний UX-підхід без переходу на окрему сторінку.



*Рисунок 3.4 — Модальне вікно створення нового курсу*

На рисунку 3.5 відображено головну сторінку студента. На відміну від адміністратора, студент бачить лише ті курси, на які його зараховано, і не має доступу до адміністративних розділів. У лівій панелі навігації відсутні пункти «Блокчейн» та «Журнал дій».

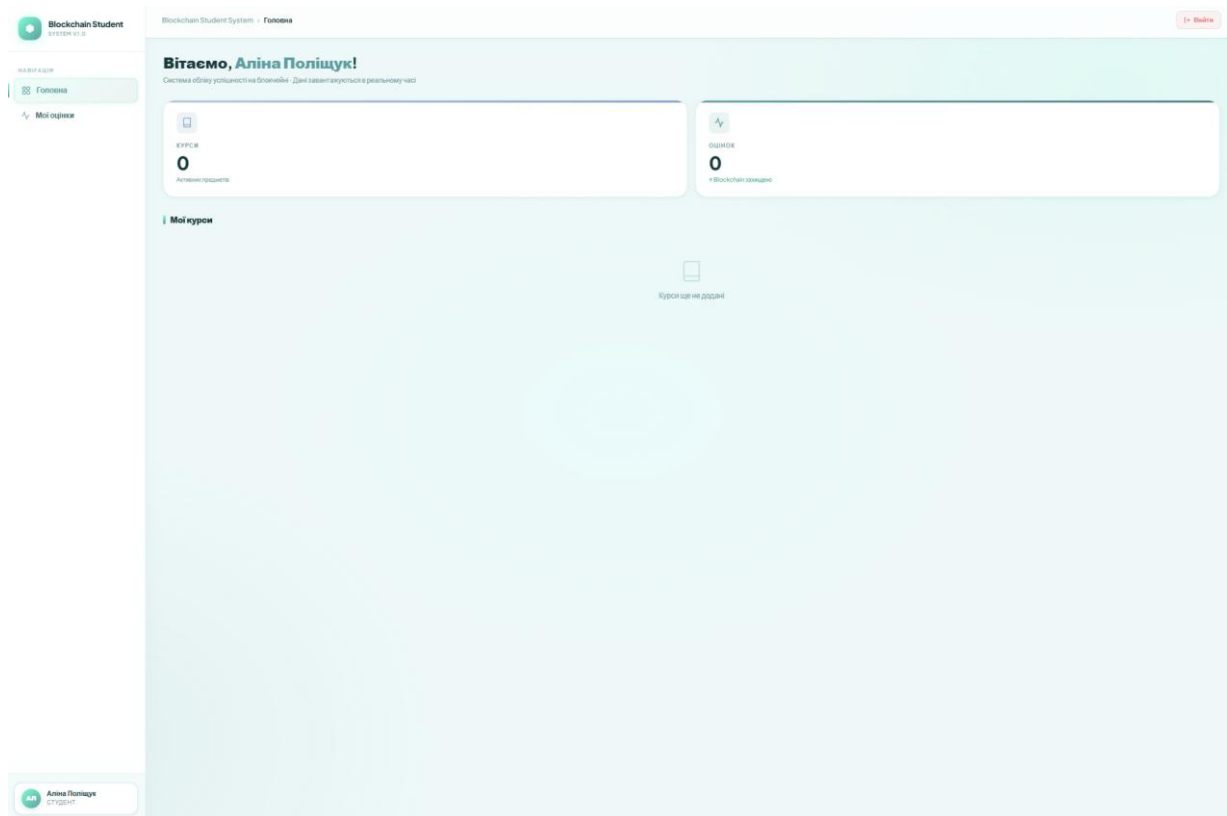


Рисунок 3.5 — Головна сторінка студента (без курсів)

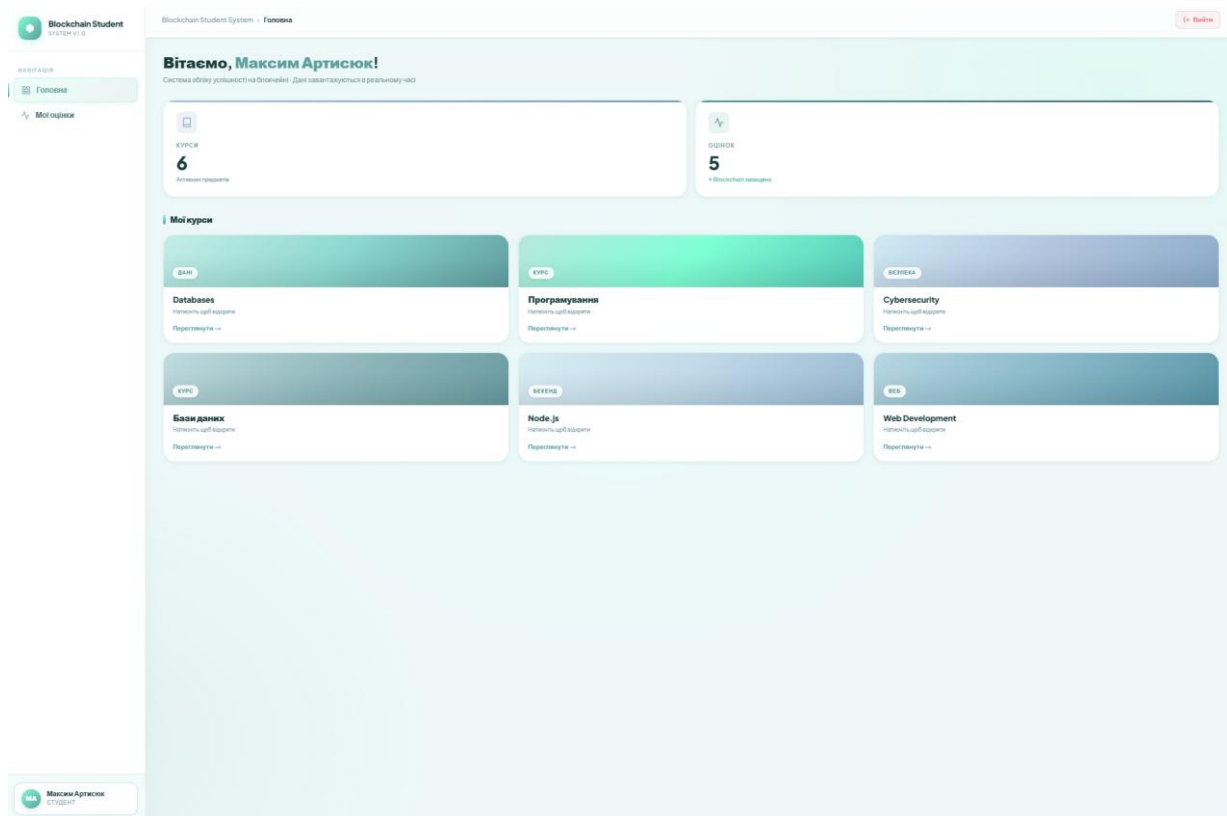


Рисунок 3.6 — Головна сторінка студента з розширеним набором курсів

На рисунку 3.7 показано сторінку управління курсом «Databases». Адміністратор може додавати лабораторні роботи (з назвою та максимальним балом), додавати студентів на курс за логіном, а також переглядати список наявних лабораторних робіт з кнопками редагування та видалення.

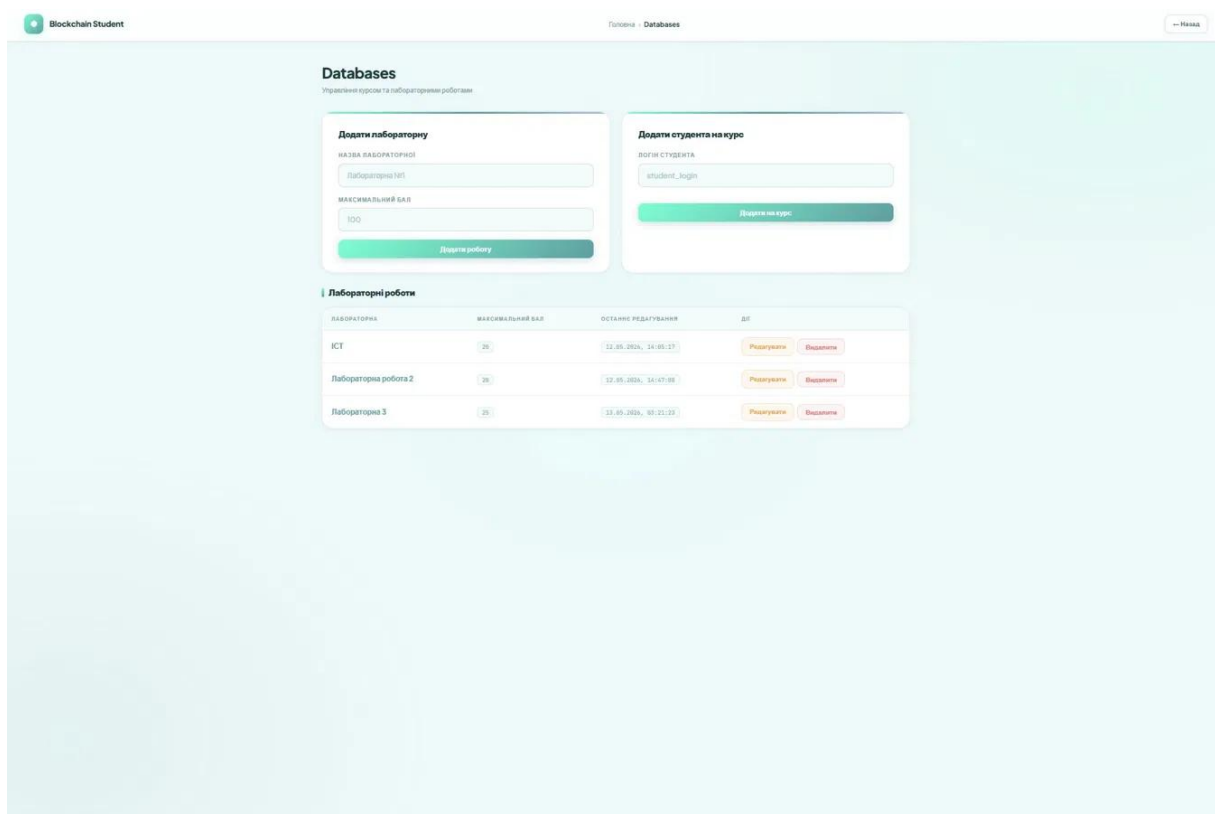
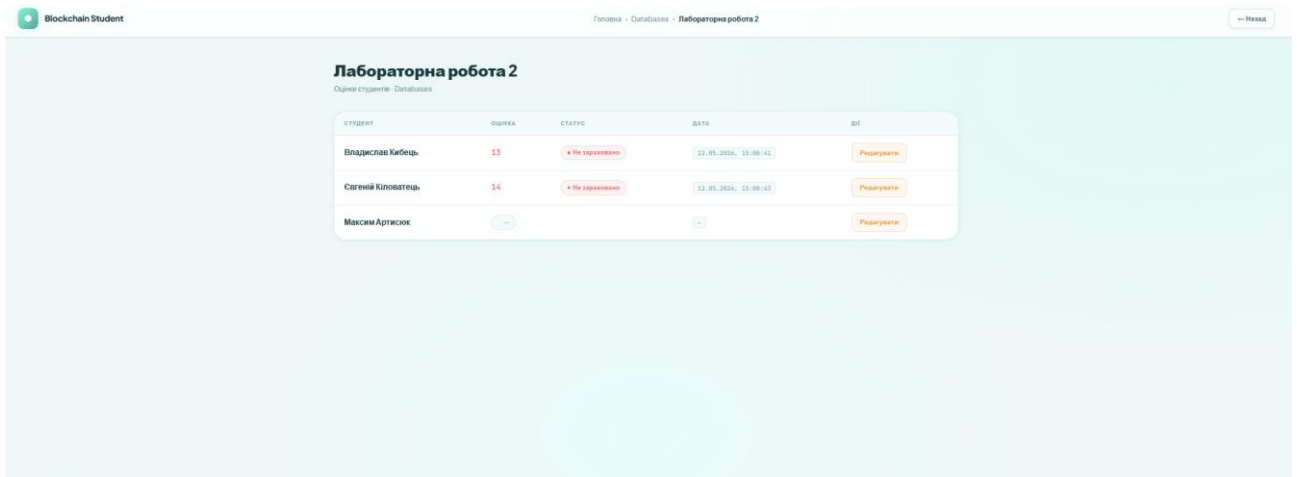


Рисунок 3.7 — Сторінка управління курсом (адміністратор)

На рисунку 3.8 показано сторінку лабораторної роботи ІСТ у курсі Databases. Таблиця відображає список студентів курсу, їхні оцінки та статус («Зараховано» / «Не зараховано»). Адміністратор має доступ до кнопки «Редагувати» для кожного рядка.



Blockchain Student

Головна > Databases > Лабораторна робота 2

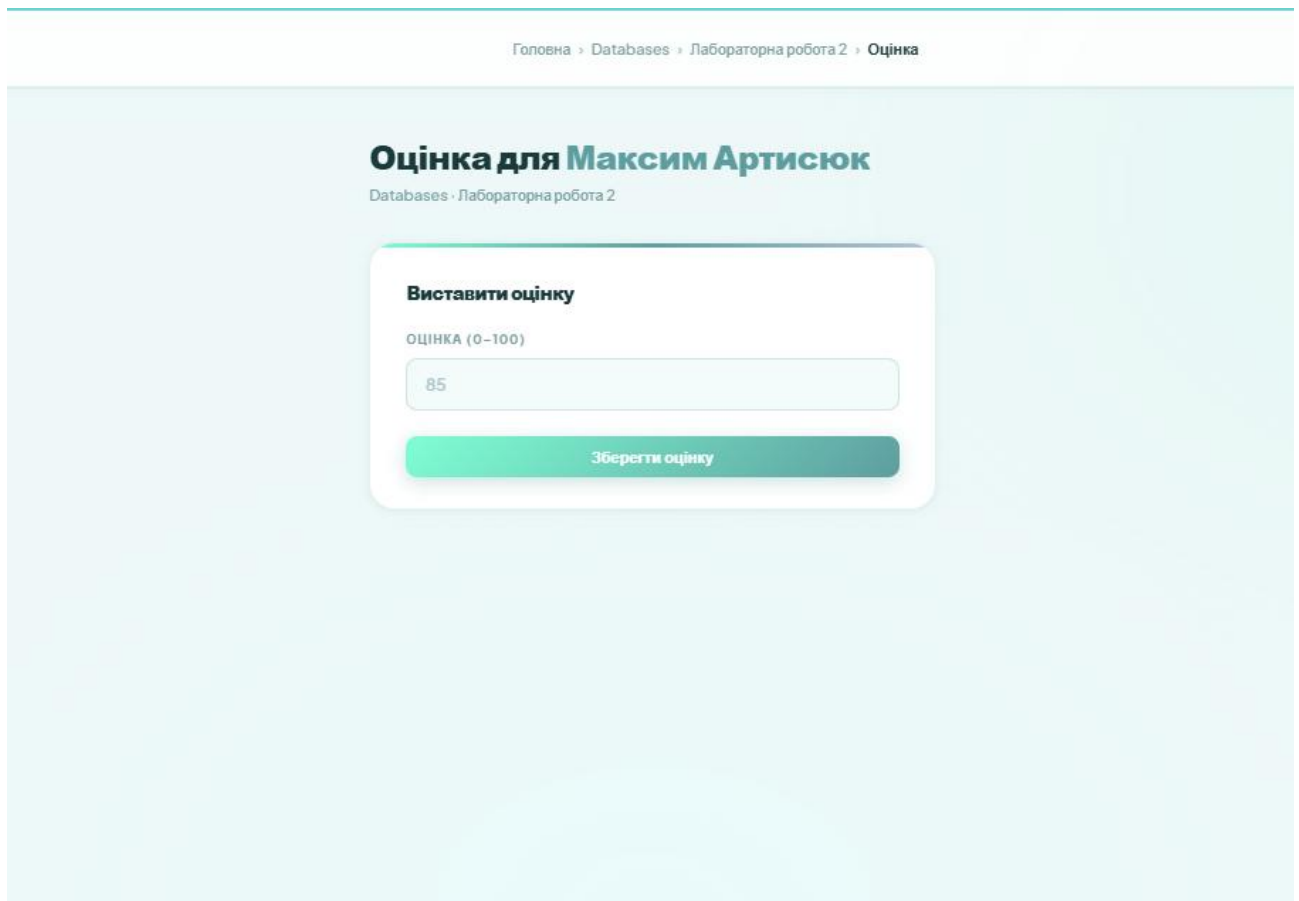
Лабораторна робота 2

Оцінки студентів: Databases

| СТУДЕНТ           | ОЦІНКА | СТАТУС       | ДАТА                 | ДІЯ        |
|-------------------|--------|--------------|----------------------|------------|
| Владислав Кибець  | 13     | Не заповнено | 12.05.2024, 15:09:41 | Редигувати |
| Степан Кішоватень | 14     | Не заповнено | 12.05.2024, 15:09:43 | Редигувати |
| Максим Артисюк    | —      | —            | —                    | Редигувати |

Рисунок 3.8 — Сторінка оцінок студентів по лабораторній роботі

На рисунку 3.9 показано форму виставлення оцінки студенту Максим Артисюк у курсі Databases, лабораторна ІСТ. Форма містить одне числове поле для введення балу (0–100) та кнопку збереження.



Головна > Databases > Лабораторна робота 2 > Оцінка

Оцінка для Максим Артисюк

Databases > Лабораторна робота 2

**Виставити оцінку**

ОЦІНКА (0–100)

85

Зберегти оцінку

Рисунок 3.9 — Форма виставлення оцінки студенту

Після збереження оцінки дані одночасно записуються до MongoDB та додаються як новий блок до блокчейн-ланцюга, після чого оновлений ланцюг серіалізується у файл studentchain.json.

На рисунку 3.10 відображено розділ «Мої оцінки» з панелі адміністратора, що показує всі записані в системі оцінки з повними метаданими, включаючи скорочений SHA-256 хеш відповідного блоку.

| СТУДЕНТ                | ПРЕДМЕТ   | ЛАБОРАТОРНА          | ОЦІНКА | ДАТА                 | HASH         | ДІЇ                       |
|------------------------|-----------|----------------------|--------|----------------------|--------------|---------------------------|
| 11 Владислав Кибець    | Databases | ІСТ                  | 12     | 12.05.2024, 15:00:38 | 888772244... | Роз. <a href="#">Вил.</a> |
| 12 Святослав Кішоватць | Databases | ІСТ                  | 12     | 12.05.2024, 15:00:31 | 807834465... | Роз. <a href="#">Вил.</a> |
| 13 Владислав Кибець    | Databases | Лабораторна робота 2 | 13     | 12.05.2024, 15:00:41 | 888346753... | Роз. <a href="#">Вил.</a> |
| 14 Святослав Кішоватць | Databases | Лабораторна робота 2 | 14     | 12.05.2024, 15:00:43 | 811327345... | Роз. <a href="#">Вил.</a> |
| 14 Святослав Кішоватць | Databases | Лабораторна робота 2 | 14     | 12.05.2024, 15:00:47 | 887475467... | Роз. <a href="#">Вил.</a> |
| 28 Владислав Кибець    | Databases | ІСТ                  | 28     | 12.05.2024, 15:01:48 | 847634567... | Роз. <a href="#">Вил.</a> |

Рисунок 3.10 — Перегляд журналу оцінок (адміністратор)

На рисунку 3.11 відображено розділ «Мої оцінки» для студента. Стовпець «Дії» прихований, і студент може лише переглядати власні оцінки без можливості редагування.

| СТУДЕНТ           | ПРЕДМЕТ         | ЛАБОРАТОРНА          | ОЦІНКА | ДАТА                 | НАВІВ     |
|-------------------|-----------------|----------------------|--------|----------------------|-----------|
| 12 Максим Артісюк | Databases       | ІСТ                  | 12     | 22.05.2024, 11:22:38 | 807654321 |
| 13 Максим Артісюк | Node.js         | Лабораторна робота 1 | 13     | 22.05.2024, 11:34:27 | 842847516 |
| 14 Максим Артісюк | Cybersecurity   | Лабораторна робота 1 | 14     | 22.05.2024, 11:34:44 | 850764182 |
| 18 Максим Артісюк | Web Development | Самостійна робота    | 18     | 22.05.2024, 11:35:48 | 899904321 |
| 13 Максим Артісюк | Databases       | Лабораторна робота 3 | 13     | 22.05.2024, 11:34:35 | 842847516 |

Рисунок 3.11 — Перегляд оцінок студентом (режим читання)

На рисунку 3.12 показано перегляд оцінок студентом у контексті конкретного курсу та лабораторних робіт. Студент бачить список лабораторних робіт, свою оцінку та дату виставлення.

| ЛАБОРАТОРНА          | ОЦІНКА | ДАТА ПОВІДОМЛЕННЯ    |
|----------------------|--------|----------------------|
| ІСТ                  | 12     | 22.05.2024, 11:22:38 |
| Лабораторна робота 2 | 13     | 22.05.2024, 11:34:27 |
| Лабораторна 3        | 14     | 22.05.2024, 11:34:44 |

Рисунок 3.12 — Оцінки студента в розрізі курсу

На рисунку 3.13 відображено Blockchain Explorer — браузер блоків ланцюга. Кожен блок відображається у вигляді картки з полями: порядковий номер, часова мітка, поточний хеш (скорочений), хеш попереднього блоку та дані оцінки. Статус «Ланцюг валідний» відображається у верхній частині сторінки.

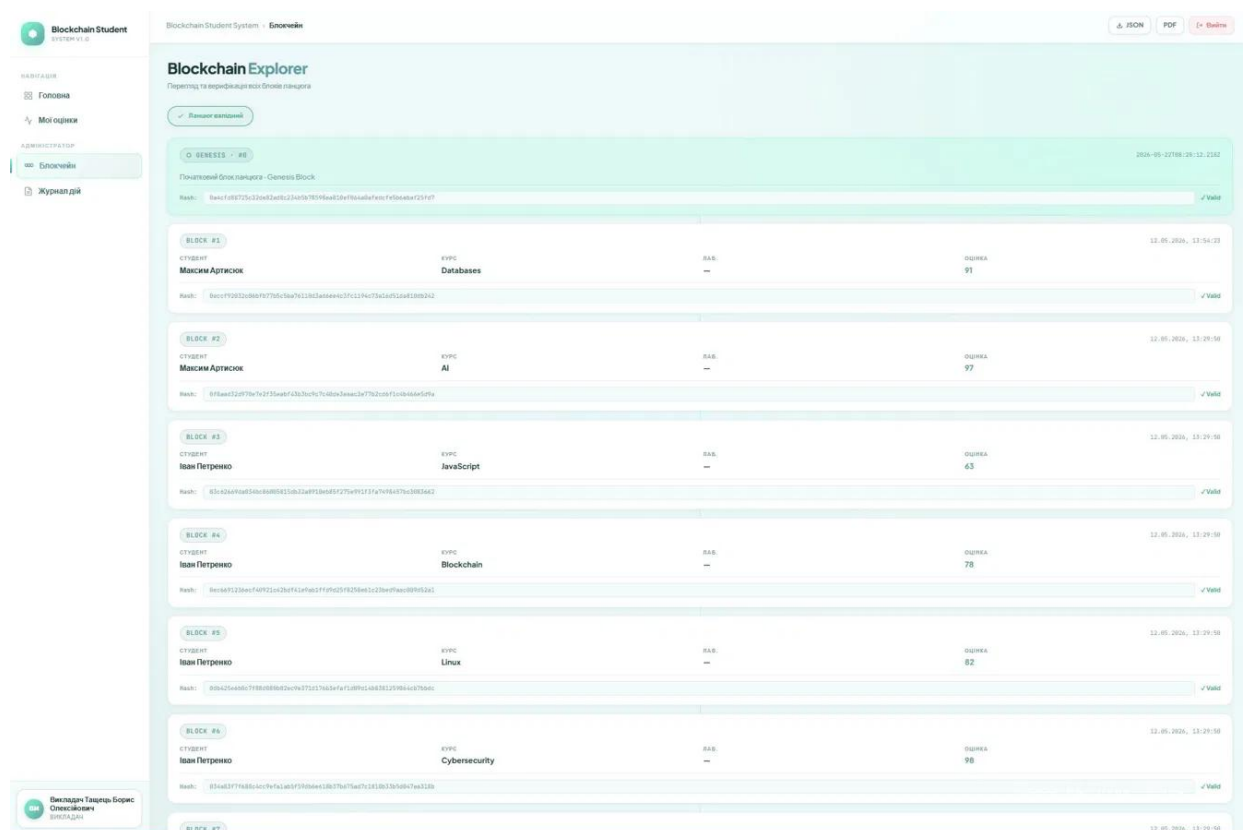


Рисунок 3.13 — Blockchain Explorer: перегляд блоків ланцюга

На рисунку 3.14 показано Журнал дій — повну хронологічну стрічку всіх операцій у системі з відображенням виконавця та часу кожної дії.

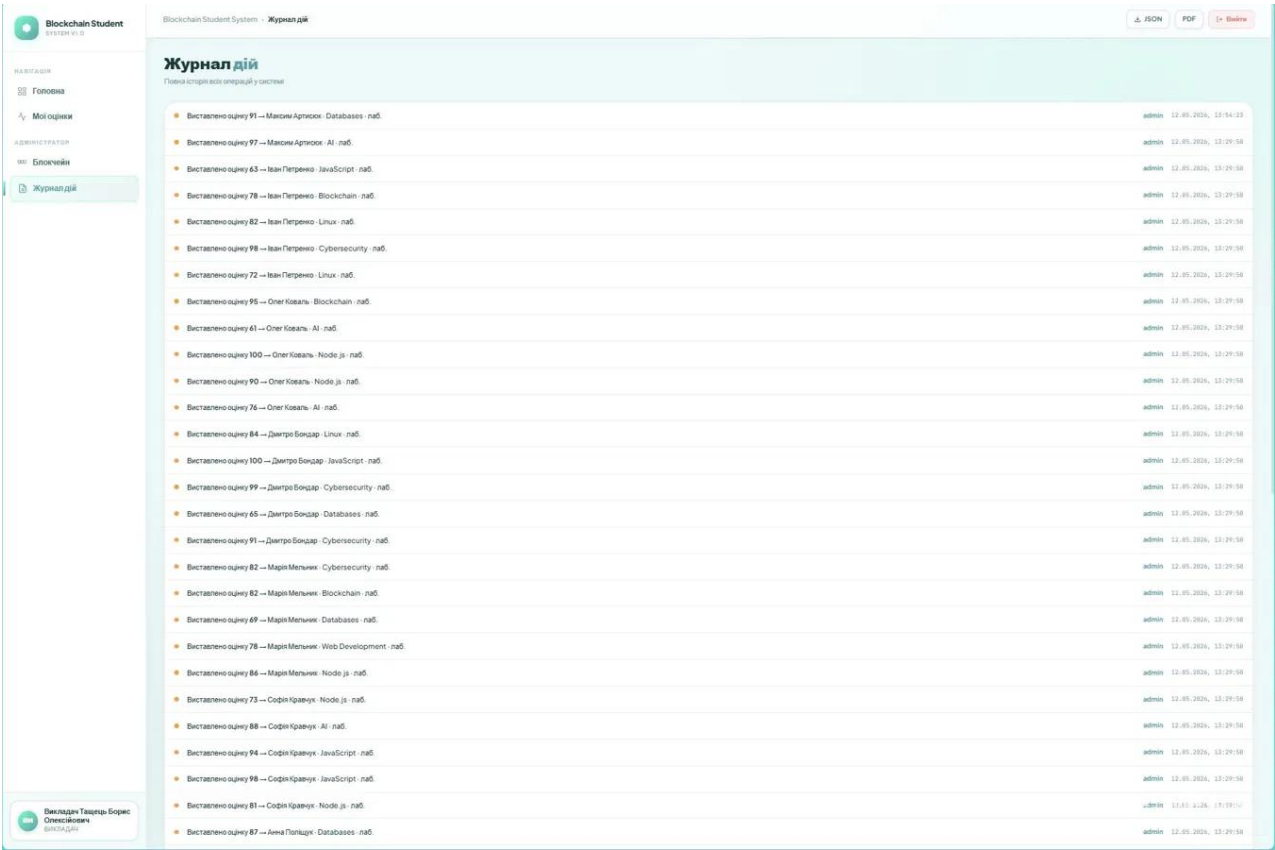


Рисунок 3.14 — Журнал дій: повна хронологія операцій

### 3.6 Технічні характеристики розробленої системи

Розроблена система має такі технічні характеристики та вимоги до середовища розгортання:

Таблиця 3.1 — Технічні вимоги до середовища розгортання

| Компонент                | Мінімальні вимоги                              |
|--------------------------|------------------------------------------------|
| Сервер ОС                | Linux (Ubuntu 20.04+), macOS або Windows 10/11 |
| Середовище виконання     | Node.js версії 18 LTS або вище                 |
| База даних               | MongoDB версії 6.0+, порт 27017 (localhost)    |
| Оперативна пам'ять       | Мінімум 512 МБ (рекомендовано 2 ГБ)            |
| Вільний дисковий простір | Мінімум 200 МБ                                 |
| Мережевий порт           | HTTP-порт 3000 (конфігурується)                |

|                 |                                               |
|-----------------|-----------------------------------------------|
| Браузер клієнта | Chrome 90+, Firefox 88+, Safari 14+, Edge 90+ |
| express         | ^5.2.1                                        |
| express-session | ^1.19.0                                       |
| mongoose        | ^9.6.2                                        |
| pdfkit          | ^0.18.0                                       |
| sqlite3         | ^6.0.1                                        |

Запуск системи виконується командою `node server.js` у кореневій директорії проекту після попереднього встановлення залежностей командою `npm install`. Початкове наповнення бази даних тестовими даними виконується скриптом `seed.js`.

### 3.7 Тестування системи

Тестування розробленої системи проводилося методом функціонального тестування на основі заздалегідь підготовлених тест-кейсів. Стратегія передбачає перевірку кожного функціонального модуля: автентифікація, управління оцінками, цілісність блокчейну, розмежування доступу, журналювання та генерація PDF.

**Таблиця 3.2 — Результати функціонального тестування системи**

| № | Назва тест-кейсу                            | Очікуваний результат                              | Результат |
|---|---------------------------------------------|---------------------------------------------------|-----------|
| 1 | Вхід адміністратора (login:admin, pass:123) | Перенаправлення на головну; роль admin у сесії    | Пройдено  |
| 2 | Вхід студента (MaksymArtysiuk1)             | Перенаправлення; відображення лише власних курсів | Пройдено  |
| 3 | Невірний пароль                             | Сторінка помилки «Невірний логін або пароль»      | Пройдено  |
| 4 | Додавання оцінки (Максим Артисюк, AI, 85)   | Новий блок у ланцюгу; запис у журналі             | Пройдено  |

|    |                                                |                                                 |          |
|----|------------------------------------------------|-------------------------------------------------|----------|
| 5  | Перевірка цілісності блокчейну (GET /validate) | Відповідь «Blockchain VALID»                    | Пройдено |
| 6  | Підробка блоку (grade=9999)                    | GET /validate повертає «Blockchain INVALID»     | Пройдено |
| 7  | Студент відкриває /blockchain                  | Перенаправлення «Доступ заборонений»            | Пройдено |
| 8  | Студент бачить лише свої оцінки                | Тільки записи з student = поточний студент      | Пройдено |
| 9  | Видалення оцінки (роль admin)                  | Блок видалено; ланцюг побудовано                | Пройдено |
| 10 | Генерація PDF (GET /export-pdf)                | Браузер завантажує students.pdf                 | Пройдено |
| 11 | Журнал дій (GET /logs)                         | Таблиця подій у зворотньому порядку             | Пройдено |
| 12 | Додавання студента на курс                     | Студент у course_students.json та таблиці курсу | Пройдено |
| 13 | Вихід зі системи (GET /logout)                 | Сесію знищено; перенаправлення на /login        | Пройдено |
| 14 | API без автентифікації                         | HTTP 401 з JSON {"error": "Не авторизовано"}    | Пройдено |

### 3.8 Аналіз результатів тестування

Проведене функціональне тестування підтверджує коректну роботу всіх основних підсистем. Особливої уваги заслуговує тест-кейс №6, що перевіряє ключову властивість блокчейну — детектування несанкціонованої модифікації даних.

При прямій зміні значення оцінки в блоці без перебудови ланцюга метод `isChainValid()` негайно повертає `false`, що підтверджує ефективність криптографічного захисту. Тест-кейс №7 підтверджує коректну роботу модуля `role-guard.js`: студент не може отримати доступ до адміністративних сторінок навіть при прямому введенні URL.

Видає не критичку особливість реалізації: при видаленні оцінки весь ланцюг перебудовується з нуля, що при великій кількості блоків може займати

помітний час. Це пов'язано з необхідністю при майнінгу всіх блоків після зміни структури ланцюга і є закономірним наслідком алгоритму Proof of Work. В умовах навчальної системи з невеликою кількістю записів ця особливість не впливає на практичну використовуваність системи.

### 3.9 Аналіз безпеки системи

Система реалізує захист на кількох рівнях, що забезпечує складну безпеку академічних даних.

Автентифікація та авторизація. Серверна сесія `express-session` зберігається в пам'яті сервера. Кожен запит до захищеного маршруту проходить перевірку наявності активної сесії та відповідної ролі (`admin/student`). Компрометація клієнтського коду не дає змоги обійти серверну перевірку ролі.

Криптографічний захист даних. Кожна оцінка, записана в блокчейн, прив'язана до хеш-ланцюга. Спроба змінити будь-яке поле будь-якого блоку порушує хеш усього наступного ланцюга. Метод `isChainValid()` виявляє будь-яку несанкціоновану модифікацію навіть після перезапуску сервера, оскільки перевіряє файл `studentchain.json`.

Клієнтський захист ролей. Модуль `role-guard.js` застосовує двоетапний підхід: попереднє приховування через CSS перешкоджає миготінню адмін-елементів у студентів, а наступна серверна перевірка ролі забезпечує кінцевий рівень захисту. Список заблокованих для студентів URL перевіряється явно.

Захист від підробки даних. Функція `saveData()` перезаписує файл `studentchain.json` при кожному додаванні блоку, що забезпечує атомарність операції. Якщо файл буде пошкоджений або відсутній, система генерує новий ланцюг з порожнім `genesis`-блоком.

### 3.10 Висновки до третього розділу

У третьому розділі описано повну програмну реалізацію та результати тестування блокчейн-орієнтованої системи контролю успішності студентів.

Реалізовано серверну частину на Node.js/Express, власний блокчейн-модуль з Proof of Work і SHA-256, клієнтський інтерфейс з рольовим розмежуванням та функцію генерації PDF.

Проведено функціональне тестування за 14 тест-кейсами — всі успішно пройдено. Підтверджено ключову властивість блокчейну — детектування несанкціонованої модифікації записів. Надано детальний опис усіх екранів інтерфейсу з ілюстраціями.

## ВИСНОВКИ

У бакалаврській кваліфікаційній роботі вирішено науково-практичну задачу проектування та розробки блокчейн-орієнтованої інформаційної системи контролю успішності студентів.

1. Проведено аналіз існуючих систем контролю успішності студентів (Moodle, Google Classroom, АСУ ВНЗ). Встановлено, що всі вони засновані на централізованій клієнт-серверній архітектурі, що робить їх вразливими до несанкціонованої модифікації академічних даних без можливості виявлення. Визначено п'ять ключових недоліків.
2. Досліджено принципи роботи технології блокчейн, зокрема механізм SHA-256 хешування, алгоритм Proof of Work та метод верифікації ланцюга. Обґрунтовано доцільність застосування власної реалізації блокчейну для навчальної системи.
3. Спроектовано архітектуру системи за тривірневою клієнт-серверною моделлю з вісьмома програмними модулями. Розроблено REST API з рольовою авторизацією та схемою гібридного зберігання даних (MongoDB + SQLite + Blockchain).
4. Розроблено серверну та клієнтську частини системи на стеку Node.js/Express.js/MongoDB/SQLite з власною реалізацією блокчейн-модуля. Реалізовано веб інтерфейс для двох категорій користувачів без залежностей від важких фронтенд-фреймворків.
5. Реалізовано блокчейн-механізм із SHA-256 хешуванням та алгоритмом Proof of Work. Метод isChainValid() забезпечує виявлення будь-якої несанкціонованої модифікації записів, що підтверджено тестуванням.
6. Проведено функціональне тестування системи за 14 тест-кейсами. Всі тести пройдено успішно. Підтверджено коректну роботу дворівневого захисту прав доступу.

Розроблена система дозволяє підвищити захищеність і прозорість академічних даних, забезпечити незалежну верифікацію оцінок студентами та сформувати криптографічно захищених аудит-трейл усіх академічних подій. Перспективними напрямками подальшого розвитку є реалізація розподіленого блокчейну з кількома вузлами, підвищення складності Proof of Work та інтеграція з університетськими інформаційними системами.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Nakamoto S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. URL: <https://bitcoin.org/bitcoin.pdf>
2. Swan M. Blockchain: Blueprint for a New Economy. Sebastopol : O'Reilly Media, 2015. 152 p.
3. Tapscott D., Tapscott A. Blockchain Revolution. New York : Portfolio/Penguin, 2016. 368 p.
4. Chen G. et al. Exploring blockchain technology and its potential applications for education. Smart Learning Environments. 2018. Vol. 5, No. 1. P. 1–10.
5. Grech A., Camilleri A. F. Blockchain in Education. Luxembourg : EU Publications Office, 2017. 136 p.
6. Lizcano D. et al. Blockchain-based approach to create a model of trust in open higher education. Journal of Computing in Higher Education. 2020. Vol. 32. P. 109–134.
7. Turkanovic M. et al. EduCTX: A Blockchain-Based Higher Education Credit Platform. IEEE Access. 2018. Vol. 6. P. 5112–5127.
8. Node.js Documentation. URL: <https://nodejs.org/en/docs>
9. Express.js Documentation. URL: <https://expressjs.com>
10. MongoDB Documentation. URL: <https://www.mongodb.com/docs>
11. SQLite Documentation. URL: <https://www.sqlite.org/docs.html>
12. PDFKit Documentation. URL: <https://pdfkit.org>
13. Antonopoulos A. M. Mastering Bitcoin. 2nd ed. Sebastopol : O'Reilly Media, 2017. 422 p.
14. Закон України «Про вищу освіту» від 01.07.2014 № 1556-VII. URL: <https://zakon.rada.gov.ua/laws/show/1556-18>
15. Moodle Documentation. URL: <https://docs.moodle.org>

16. Buterin V. A Next-Generation Smart Contract and Decentralized Application Platform. Ethereum White Paper. 2014.
17. Back A. Hashcash — A Denial of Service Counter-Measure. URL: <http://www.hashcash.org/papers/hashcash.pdf>. 2002.
18. NIST FIPS 180-4. Secure Hash Standard (SHS). National Institute of Standards and Technology. 2015.
19. MDN Web Docs — Fetch API. URL: [https://developer.mozilla.org/en-US/docs/Web/API/Fetch\\_API](https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API)
20. Mongoose ODM Documentation. URL: <https://mongoosejs.com/docs>

# ДОДАТКИ

## Додаток А

### Програмний код блокчейн-модуля (blockchain.js)

```
const crypto = require('crypto');

class Block {

  constructor(index, timestamp, data, previousHash = "") {
    this.index = index;
    this.timestamp = timestamp;
    this.data = data;
    this.previousHash = previousHash;

    this.nonce = 0;
    this.hash = this.calculateHash();
  }

  calculateHash() {
    return crypto
      .createHash('sha256')
      .update(
        this.index +
        this.previousHash +
        this.timestamp +
        JSON.stringify(this.data) +
        this.nonce
      )
      .digest('hex');
  }

  mineBlock(difficulty) {
    const target = Array(difficulty + 1).join('0');

    while (this.hash.substring(0, difficulty) !== target) {
      this.nonce++;
      this.hash = this.calculateHash();
    }
  }
}
```

```

        console.log('❑ Block mined:', this.hash);
    }
}

```

```

class Blockchain {

```

```

    constructor() {
        this.difficulty = 1;
        this.chain = [this.createGenesisBlock()];
    }

```

```

    createGenesisBlock() {
        const genesisBlock = new Block(
            0,
            new Date().toISOString(),
            'Genesis Block',
            '0'
        );

        genesisBlock.mineBlock(this.difficulty);
        return genesisBlock;
    }

```

```

    getLatestBlock() {
        return this.chain[this.chain.length - 1];
    }

```

```

    addBlock(data) {
        const newBlock = new Block(
            this.chain.length,
            new Date().toISOString(),
            data,
            this.getLatestBlock().hash
        );

```

```

        console.log('❑ Mining block...');
        const startTime = Date.now();

```

```

        newBlock.mineBlock(this.difficulty);

```

```

const endTime = Date.now();
console.log(
  `✔Mining finished in ${((endTime - startTime) / 1000)} seconds`
);

this.chain.push(newBlock);
}

isChainValid() {
  for (let i = 1; i < this.chain.length; i++) {

    const currentBlock = this.chain[i];
    const previousBlock = this.chain[i - 1];

    if (currentBlock.hash !== currentBlock.calculateHash()) {
      return false;
    }

    if (currentBlock.previousHash !== previousBlock.hash) {
      return false;
    }

    const target = Array(this.difficulty + 1).join('0');

    if (currentBlock.hash.substring(0, this.difficulty) !== target) {
      return false;
    }
  }

  return true;
}

getBlockByHash(hash) {
  return this.chain.find(block => block.hash === hash);
}

getBlockchainStats() {
  return {
    totalBlocks: this.chain.length,
    difficulty: this.difficulty,
  }
}

```

```

        valid: this.isChainValid(),
        latestHash: this.getLatestBlock().hash
    };
}

printChain() {
    console.log(JSON.stringify(this.chain, null, 2));
}

exportChain() {
    return this.chain.map(block => ({
        index: block.index,
        timestamp: block.timestamp,
        data: block.data,
        previousHash: block.previousHash,
        hash: block.hash,
        nonce: block.nonce
    }));
}

}

// EXPORT

module.exports = Blockchain;

```

## Додаток Б

### Програмний код модуля захисту ролей (role-guard.js)

```
/**
 * role-guard.js
 * Підключити у всіх HTML-сторінках: <script src="/role-guard.js"></script>
 *
 * Стратегія: admin-only елементи ОДРАЗУ приховані через CSS.
 * Якщо роль = admin — показуємо їх назад.
 * Якщо роль = student — залишають прихованими + додатковий захист.
 */

// — КРОК 1: Одразу ін'єктуємо CSS що ховає admin-only —————
// Це відбувається СИНХРОННО, до будь-якого рендеру
(function injectHideStyle() {
  const style = document.createElement('style');
  style.id = 'role-guard-style';
  style.textContent = '.admin-only { display: none !important; }';
  // вставляємо в <head> або на початок <body> якщо head ще не готовий
  const target = document.head || document.documentElement;
  target.insertBefore(style, target.firstChild);
})();

// — КРОК 2: Після завантаження — запитуємо роль і реагуємо —————
(async function applyRole() {
  try {
    const res = await fetch('/api/me');
    if (!res.ok) {
      window.location.href = '/login';
      return;
    }

    const user = await res.json();
    window.__currentUser = user;

    const isAdmin = user.role === 'admin';
    const isStudent = user.role === 'student';

    // — Відображення імені та ролі —————
    document.querySelectorAll('.user-display-name, #user-display-name')
      .forEach(el => { el.textContent = isAdmin ? 'Адміністратор' : (user.name || user.username); });
  });
}
```

```

document.querySelectorAll('.user-role, #user-role')
    .forEach(el => { el.textContent = isAdmin ? 'Викладач' : 'Студент'; });

if (isAdmin) {
    // — АДМИН: знімаємо приховування admin-only —————
    const style = document.getElementById('role-guard-style');
    if (style) style.remove();
    return;
}

// — СТУДЕНТ: додатковий захист —————

// Заблокувати прямий доступ до адмін-сторінок
const blockedPaths = [
    '/blockchain', '/blockchain-explorer',
    '/logs', '/admin-logs',
    '/export', '/export-pdf'
];
if (blockedPaths.some(p => window.location.pathname.startsWith(p))) {
    window.location.href = '/';
    return;
}

// Приховати колонку «Дії» у таблицях (заголовок + комірки)
document.querySelectorAll('table').forEach(table => {
    const headers = table.querySelectorAll('th');
    let actionsIdx = -1;
    headers.forEach((th, i) => {
        if (th.textContent.trim().toLowerCase() === 'дії' ||
            th.dataset.col === 'actions') {
            actionsIdx = i;
            th.style.display = 'none';
        }
    });
    if (actionsIdx >= 0) {
        table.querySelectorAll('tr').forEach(row => {
            const cell = row.querySelectorAll('td')[actionsIdx];
            if (cell) cell.style.display = 'none';
        });
    }
});

```

```
// Перетворити посилання /work/ у звичайний текст
document.querySelectorAll('a[href*="/work/"]').forEach(a => {
  const span = document.createElement('span');
  span.textContent = a.textContent;
  span.style.color = 'inherit';
  a.replaceWith(span);
});

} catch (err) {
  console.error('role-guard error:', err);
  window.location.href = '/login';
}
})();
```

## Додаток В

### Фрагменти програмного коду серверної частини (server.js)

```

const express = require('express');
console.log('SERVER FILE WORKS');
const fs = require('fs');
const path = require('path');
const session = require('express-session');
const mongoose = require('mongoose');
const PDFDocument = require('pdfkit');

const users = JSON.parse(fs.readFileSync('users.json'));
const courseStudents = fs.existsSync('course_students.json')
  ? JSON.parse(fs.readFileSync('course_students.json')) : {};

let students = [];
if (fs.existsSync('students.json')) {
  students = JSON.parse(fs.readFileSync('students.json'));
}
function saveStudents() {
  fs.writeFileSync('students.json', JSON.stringify(students, null, 2));
}

const Blockchain = require('./blockchain/blockchain');
const Grade = require('./models/Grade');

const app = express();
const PORT = 3000;

// SHARED CSS

function sharedCSS() {
  return `
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link
href="https://fonts.googleapis.com/css2?family=Plus+Jakarta+Sans:wght@300;400;500;600;700;800&family=JetBrains+Mono:wght@400;500;600&display=swap" rel="stylesheet">
    <style>
    :root {
      --bg:    #EEF8F8;
  
```

```

--surface: #FFFFFF;
--surface-2: #F4FBFB;
--card: #FFFFFF;
--card-2: #F0F9F9;
--mint: #7FFFD4;
--mint-dark: #5CCFB0;
--mint-light: rgba(127,255,212,0.18);
--mint-border: rgba(127,255,212,0.45);
--mint-glow: rgba(127,255,212,0.12);
--teal: #5F9EA0;
--teal-dark: #4A8A8C;
--teal-light: rgba(95,158,160,0.12);
--teal-border: rgba(95,158,160,0.25);
--steel: #B0C4DE;
--ice: #E0FFFF;
--text-1: #1A3A3C;
--text-2: #3D6B6D;
--text-3: #7A9EA0;
--text-4: #A8C4C6;
--danger: #E57373;
--danger-bg: rgba(229,115,115,0.08);
--warning: #F0A04B;
--warning-bg: rgba(240,160,75,0.08);
--success: #4CAF8A;
--success-bg: rgba(76,175,138,0.08);
--border: rgba(95,158,160,0.15);
--border-md: rgba(95,158,160,0.25);
--radius-xs: 5px;
--radius-sm: 9px;
--radius-md: 14px;
--radius-lg: 20px;
--shadow-sm: 0 1px 4px rgba(95,158,160,0.08),0 2px 12px rgba(95,158,160,0.05);
--shadow-md: 0 4px 20px rgba(95,158,160,0.12),0 2px 8px rgba(95,158,160,0.06);
--shadow-lg: 0 8px 40px rgba(95,158,160,0.16),0 2px 12px rgba(95,158,160,0.08);
}
*,*::before,*::after{box-sizing:border-box;margin:0;padding:0;}
html{scroll-behavior:smooth;}
body{font-family:'Plus Jakarta Sans',sans-serif;background:var(--bg);color:var(--text-1);font-size:14px;line-height:1.6;min-height:100vh;}
body::before{content:"";position:fixed;inset:0;background:radial-gradient(ellipse 60% 50% at 90% 10%,rgba(127,255,212,0.12) 0%,transparent 60%),radial-gradient(ellipse 50% 60% at 5% 90%,rgba(95,158,160,0.08) 0%,transparent 60%),radial-gradient(ellipse 40% 40% at 50% 50%,rgba(224,255,255,0.3) 0%,transparent 70%);pointer-events:none;z-index:0;}
.page{position:relative;z-index:1;max-width:1080px;margin:0 auto;padding:40px 28px 80px;}

```

```

.topbar {position: sticky; top: 0; z-index: 100; background: rgba(255, 255, 255, 0.88); backdrop-filter: blur(12px); border-bottom: 1px solid
var(--border); padding: 0 32px; height: 60px; display: flex; align-items: center; gap: 12px; box-shadow: 0 1px 0 var(--border), 0 2px 12px
rgba(95, 158, 160, 0.04);}

.topbar-brand {display: flex; align-items: center; gap: 10px; text-decoration: none;}

.brand-icon-sm {width: 34px; height: 34px; background: linear-gradient(135deg, var(--mint), var(--teal)); border-
radius: 10px; display: flex; align-items: center; justify-content: center; box-shadow: 0 4px 12px rgba(127, 255, 212, 0.3); flex-shrink: 0;}

.brand-icon-sm svg {width: 16px; height: 16px;}

.brand-name {font-size: 14px; font-weight: 800; color: var(--text-1); letter-spacing: -0.02em;}

.topbar-spacer {flex: 1;}

.breadcrumb {display: flex; align-items: center; gap: 6px; font-size: 12px; color: var(--text-3); font-weight: 500;}

.breadcrumb a {color: var(--text-3); text-decoration: none;}

.breadcrumb a:hover {color: var(--teal);}

.breadcrumb-sep {color: var(--text-4);}

.breadcrumb-cur {color: var(--text-1); font-weight: 700;}

.page-header {margin-bottom: 28px;}

.page-title {font-size: 26px; font-weight: 800; color: var(--text-1); letter-spacing: -0.03em; margin-bottom: 4px; line-height: 1.15;}

.page-title span {color: var(--teal);}

.page-sub {color: var(--text-3); font-size: 12px; font-weight: 500;}

.btn {display: inline-flex; align-items: center; gap: 6px; padding: 9px 18px; border-radius: var(--radius-sm); font-size: 12px; font-
weight: 700; font-family: 'Plus Jakarta Sans', sans-serif; cursor: pointer; border: none; text-decoration: none; transition: all 0.16s; white-
space: nowrap; letter-spacing: 0.01em;}

.btn-primary {background: linear-gradient(135deg, var(--mint), var(--teal)); color: #fff; box-shadow: 0 4px 14px rgba(95, 158, 160, 0.3);}

.btn-primary:hover {background: linear-gradient(135deg, var(--mint-dark), var(--teal-dark)); box-shadow: 0 6px 20px
rgba(95, 158, 160, 0.4); transform: translateY(-1px);}

.btn-ghost {background: var(--surface); color: var(--text-2); border: 1px solid var(--border-md); box-shadow: var(--shadow-sm);}

.btn-ghost:hover {background: var(--ice); color: var(--teal); border-color: var(--mint-border);}

.btn-warn {background: var(--warning-bg); color: var(--warning); border: 1px solid rgba(240, 160, 75, 0.25);}

.btn-warn:hover {background: rgba(240, 160, 75, 0.15);}

.btn-danger {background: var(--danger-bg); color: var(--danger); border: 1px solid rgba(229, 115, 115, 0.2);}

.btn-danger:hover {background: rgba(229, 115, 115, 0.15);}

.btn-sm {padding: 6px 13px; font-size: 11px;}

.card {background: var(--card); border: 1px solid var(--border); border-radius: var(--radius-lg); padding: 24px 28px; box-shadow: var(--
shadow-sm); position: relative; overflow: hidden;}

.card::before {content: ""; display: block; height: 3px; background: linear-gradient(90deg, var(--mint), var(--teal), var(--
steel)); position: absolute; top: 0; left: 0; right: 0;}

.card-title {font-size: 14px; font-weight: 800; color: var(--text-1); letter-spacing: -0.02em; margin-bottom: 18px; padding-top: 4px;}

.grid-2 {display: grid; grid-template-columns: 1fr 1fr; gap: 20px; margin-bottom: 24px;}

@media (max-width: 700px) {
.grid-2 {grid-template-columns: 1fr;}
}

.form-group {margin-bottom: 14px;}

.form-label {display: block; font-size: 10px; font-weight: 700; letter-spacing: 0.1em; text-transform: uppercase; color: var(--text-
3); margin-bottom: 6px;}

.form-input {width: 100%; padding: 11px 14px; background: var(--surface-2); border: 1px solid var(--border-md); border-radius: var(--
radius-sm); color: var(--text-1); font-family: 'Plus Jakarta Sans', sans-serif; font-size: 13px; font-weight: 500; outline: none; transition: all
0.16s;}

```

```

.form-input:focus{border-color:var(--mint-dark);box-shadow:0 0 3px rgba(127,255,212,0.18);background:#fff;}
.form-input::placeholder{color:var(--text-4);}
.table-wrap{background:var(--card);border:1px solid var(--border);border-radius:var(--radius-lg);overflow:hidden;box-shadow:var(--shadow-sm);margin-bottom:24px;}
table{width:100%;border-collapse:collapse;font-size:13px;}
thead th{background:var(--surface-2);font-size:9px;letter-spacing:0.12em;text-transform:uppercase;color:var(--text-3);font-weight:700;padding:12px 16px;text-align:left;border-bottom:1px solid var(--border);}
tbody tr{border-bottom:1px solid var(--border);transition:background 0.12s;}
tbody tr:last-child{border-bottom:none;}
tbody tr:hover{background:rgba(127,255,212,0.04);}
td{padding:12px 16px;color:var(--text-2);vertical-align:middle;}
td.name{color:var(--text-1);font-weight:700;}
.pill{display:inline-flex;align-items:center;gap:4px;padding:3px 10px;border-radius:20px;font-size:10px;font-weight:700;letter-spacing:0.04em;}
.pill::before{content:"";display:block;width:5px;height:5px;border-radius:50%;}
.pill-success{background:var(--success-bg);color:var(--success);border:1px solid rgba(76,175,138,0.2);}
.pill-success::before{background:var(--success);}
.pill-danger{background:var(--danger-bg);color:var(--danger);border:1px solid rgba(229,115,115,0.2);}
.pill-danger::before{background:var(--danger);}
.pill-warn{background:var(--warning-bg);color:var(--warning);border:1px solid rgba(240,160,75,0.2);}
.pill-warn::before{background:var(--warning);}
.mono{font-family:'JetBrains Mono',monospace;font-size:10px;color:var(--text-3);background:var(--surface-2);border:1px solid var(--border);padding:2px 7px;border-radius:var(--radius-xs);}
.action-btns{display:flex;gap:6px;align-items:center;}
.action-form{display:inline;background:none;padding:0;margin:0;}
.section-header{display:flex;align-items:center;justify-content:space-between;margin-bottom:14px;}
.section-title{font-size:14px;font-weight:800;color:var(--text-1);letter-spacing:-0.02em;display:flex;align-items:center;gap:8px;}
.section-title::before{content:"";display:block;width:4px;height:16px;background:linear-gradient(to bottom,var(--mint),var(--teal));border-radius:3px;}
.link{color:var(--teal);font-weight:700;text-decoration:none;}
.link:hover{text-decoration:underline;}
.denied{max-width:420px;margin:80px auto;text-align:center;padding:20px;}
.denied-icon{width:64px;height:64px;background:var(--danger-bg);border:1px solid rgba(229,115,115,0.2);border-radius:50%;display:flex;align-items:center;justify-content:center;margin:0 auto 18px;font-size:26px;box-shadow:var(--shadow-sm);}
.denied h1{font-size:20px;font-weight:800;color:var(--text-1);margin-bottom:8px;letter-spacing:-0.02em;}
.denied p{color:var(--text-3);margin-bottom:24px;font-size:13px;}
.alert{padding:14px 18px;border-radius:var(--radius-md);font-size:13px;font-weight:500;margin-bottom:20px;display:flex;align-items:center;gap:10px;}
.alert-danger{background:var(--danger-bg);border:1px solid rgba(229,115,115,0.2);color:var(--danger);}
.alert-success{background:var(--success-bg);border:1px solid rgba(76,175,138,0.2);color:var(--success);}
::-webkit-scrollbar{width:5px;}
::-webkit-scrollbar-track{background:transparent;}
::-webkit-scrollbar-thumb{background:var(--border-md);border-radius:5px;}
@keyframes fade-up{from{opacity:0;transform:translateY(16px);}to{opacity:1;transform:translateY(0);}}

```

```
.card{animation:fade-up 0.35s ease both;}
</style>`;
}
```

```
function topbar(breadcrumb = [], rightButtons = "") {
  const crumbs = [
    `<a href="/" class="topbar-brand">
      <div class="brand-icon-sm">
        <svg viewBox="0 0 24 24" fill="none">
          <path d="M4 8l8-5 8 5v8l-8 5-8-5V8z" fill="rgba(255,255,255,0.9)"/>
          <path d="M12 3v18M4 8l8 5 8-5" stroke="rgba(255,255,255,0.6)" stroke-width="1.5"/>
        </svg>
      </div>
      <span class="brand-name">Blockchain Student</span>
    </a>`,
    `<div class="topbar-spacer"></div>`,
    `<div class="breadcrumb">`,
    `<a href="/">Головна</a>`,
    ...breadcrumb.map((b, i) =>
      i === breadcrumb.length - 1
        ? `<span class="breadcrumb-sep">></span><span class="breadcrumb-cur">${b}</span>`
        : `<span class="breadcrumb-sep">></span><a href="#">${b}</a>`
    ),
    `</div>`,
    `<div class="topbar-spacer"></div>`,
    rightButtons
  ].join("");
  return `<div class="topbar">${crumbs}</div>`;
}
```

```
function deniedPage(msg = 'Ця сторінка вам недоступна.') {
  return `<!DOCTYPE html><html lang="uk"><head><meta charset="UTF-8">
<title>Доступ заборонений</title>${sharedCSS()}</head><body>
${topbar([], `<a href="/" class="btn btn-ghost btn-sm">На головну</a>`)}
<div class="page">
  <div class="denied">
    <div class="denied-icon">❏</div>
    <h1>Доступ заборонений</h1>
    <p>${msg}</p>
    <a href="/" class="btn btn-primary">← На головну</a>
  </div>
</div></body></html>`;
}
```

```

// MIDDLEWARE
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(session({ secret: 'secret-key', resave: false, saveUninitialized: true }));

app.use((req, res, next) => {
  const open = ['/login', '/role-guard.js'];
  if (open.some(p => req.path.startsWith(p))) return next();
  if (!req.session.loggedIn) {
    if (req.path.startsWith('/api/')) {
      return res.status(401).json({ error: 'Не авторизовано' });
    }
    return res.redirect('/login');
  }
  next();
});

app.use(express.static(path.join(__dirname, 'public')));

// TEST
app.get('/test', (req, res) => { res.send('SERVER WORKS'); });

// MAIN PAGE
app.get('/', (req, res) => {
  if (!req.session.loggedIn) return res.redirect('/login');
  res.sendFile(path.join(__dirname, 'public', 'index.html'));
});

// MONGODB

mongoose.connect('mongodb://127.0.0.1:27017/blockchainDB')
  .then(() => console.log('✔MongoDB connected'))
  .catch(error => console.log(error));

// BLOCKCHAIN

const studentBlockchain = new Blockchain();
studentBlockchain.chain = [studentBlockchain.chain[0]];

```

```

const DATA_FILE = 'data.json';
const LOG_FILE = 'logs.json';
const COURSES_FILE = 'courses.json';

if (fs.existsSync(DATA_FILE)) {
  const data = fs.readFileSync(DATA_FILE);
  const parsed = JSON.parse(data);
  parsed.forEach(item => studentBlockchain.addBlock(item));
}

if (studentBlockchain.chain.length <= 1) {
  const groups = ['KH-21','KH-22','ПІ-21','ПІ-22','ПІЗ-21'];
  const subjects = ['Blockchain','JavaScript','Node.js','Cybersecurity','Databases','AI','Linux','Web Development'];
  const demoStudents = ['Андрій Шевченко','Максим Артісюк','Іван Петренко','Олег Коваль','Дмитро Бондар','Марія Мельник','Софія Кравчук','Анна Поліщук','Юлія Ткаченко','Назар Ільчук','Богдан Левченко','Роман Савчук','Віктор Климчук','Олександр Гнатюк','Тарас Бойко','Катерина Мазур','Ірина Вовк','Владислав Романюк','Єгор Данилюк','Микола Литвин'];
  demoStudents.forEach(student => {
    for (let i = 0; i < 5; i++) {
      studentBlockchain.addBlock({
        student,
        group: groups[Math.floor(Math.random() * groups.length)],
        subject: subjects[Math.floor(Math.random() * subjects.length)],
        grade: Math.floor(Math.random() * 41) + 60,
        date: new Date().toLocaleString()
      });
    }
  });
  console.log('✓DEMO STUDENTS CREATED');
}

function saveData(data) {
  fs.writeFileSync(DATA_FILE, JSON.stringify(data, null, 2));
}

function addLog(action, user) {
  let logs = [];
  if (fs.existsSync(LOG_FILE)) logs = JSON.parse(fs.readFileSync(LOG_FILE));
  logs.push({ action, user, date: new Date().toLocaleString() });
  fs.writeFileSync(LOG_FILE, JSON.stringify(logs, null, 2));
}

```

```
function loadCoursesList() {
  if (fs.existsSync(COURSES_FILE)) {
    return JSON.parse(fs.readFileSync(COURSES_FILE));
  }
  const fromChain = [...new Set(
    studentBlockchain.chain.slice(1).map(b => b.data.subject).filter(Boolean)
  )];
  fs.writeFileSync(COURSES_FILE, JSON.stringify(fromChain, null, 2));
  return fromChain;
}
```

```
function saveCoursesList(list) {
  fs.writeFileSync(COURSES_FILE, JSON.stringify(list, null, 2));
}
```

// LOGIN PAGE

```
app.get('/login', (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'login.html'));
});
```

// LOGIN POST

```
app.post('/login', (req, res) => {
  const username = req.body.username.trim();
  const password = req.body.password.trim();
  const user = users.find(u => u.username === username && u.password === password);
  if (!user) {
    return res.send(`<!DOCTYPE html><html lang="uk"><head><meta charset="UTF-8">
<title>Помилка входу</title>${sharedCSS()}</head><body>
${topbar([], "")}
<div class="page" style="max-width:440px;">
  <div class="card" style="margin-top:60px;text-align:center;padding:40px 36px;">
    <div style="width:56px;height:56px;background:var(--danger-bg);border:1px solid rgba(229,115,115,0.2);border-
radius:50%;display:flex;align-items:center;justify-content:center;margin:0 auto 20px;font-size:24px;">❌</div>
    <div class="page-title" style="font-size:19px;margin-bottom:8px;">Невірний логін або пароль</div>
    <p style="color:var(--text-3);font-size:13px;margin-bottom:28px;font-weight:500;">Перевірте введені дані та спробуйте
ще раз.</p>
```

```

    <a href="/login" class="btn btn-primary" style="display:inline-flex;justify-content:center;width:100%;">
      ← Повернутись до входу
    </a>
  </div>
</div></body></html>`);
}
req.session.loggedIn = true;
req.session.role = user.role;
req.session.studentName = user.name;
req.session.username = user.username;
addLog('Вхід у систему', username);
res.redirect('/');
});

```

// API: CURRENT USER INFO

```

app.get('/api/me', (req, res) => {
  if (!req.session.loggedIn) return res.status(401).json({ error: 'Не авторизовано' });
  res.json({ role: req.session.role, name: req.session.studentName, username: req.session.username });
});

```

// API: MY GRADES

```

app.get('/api/my-grades', (req, res) => {
  if (!req.session.loggedIn) return res.status(403).json({ error: 'Доступ заборонений' });
  let data = studentBlockchain.chain.slice(1)
    .filter(block => block.data.type === 'grade')
    .map(block => ({
      id: block.index, student: block.data.student,
      subject: block.data.subject, lab: block.data.lab,
      grade: block.data.grade, date: block.data.date, hash: block.hash
    }));
});

```

... (наведено перші 350 рядків з 1159)