МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Розробка децентралізованої системи взаємодії та винагород на базі блокчейну

Здобувача вищої освіти другого (магістерського)
рівня,
ІІ курсу, групи КМІ-24-1, спеціальності 122 «Комп'ютерні науки»,
освітня програма «Інформаційні управляючі системи і технології в економіці»

_____ Мілінтєєва Данііла Дмитровича
(підпис) (прізвище, ім'я, по батькові)

Науковий керівник _____ Редич Олександр Володимирович, к.е.н., доцент
(підпис) (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Керівник освітньо-професійної програми _____ Омельчук Антон Анатолійович, канд. техн. наук
(підпис) (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту за рішенням кафедри протокол від «02» січня 2026 № 10

Завідувач кафедри _____ Філоненко Михайло Миколайович, канд. фіз.-мат. наук, доцент
(підпис) (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Ірпінь – 2026

Реєстрація №33 22.01.2026 _____
(дата) (підпис відповідальної особи)

Результати перевірки до захисту 02.01.2026 _____
(до захисту, на доопрацювання) (дата) (підпис)

Результат захисту 5 (відмінно) 09.02.2026 _____
(оцінка) (дата) (підпис)

Голова комісії: _____ О.С. Дюбанов
(підпис)

Члени комісії: _____ А.Ю. Горбовий
(підпис)

_____ М.М. Філоненко
(підпис)

_____ І.Д. Погорєловська
(підпис)

БЛАНК ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки в балах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії,секретаря, (підписи)
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст кваліфікаційної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	28		09.02.2026 Дюбанов О.С. Горбовий А.Ю. Філоненко М.М. Погорєловська І.Д. Суховецька Л.П.
2. Дотримання вимог щодо оформлення	від 0 до 10	9		
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	4		
4. Рецензент	від 0 до 5	5		
РАЗОМ БАЛІВ	від 0 до 50	46		

Оцінювання кваліфікаційної роботи науковим керівником - 45 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалоюЄКТС	Оцінка за національною шкалою	Дата	П.І.Б.: голови, членів комісії, секретаря, (підпис)
91	A	5	09.02.2026	Дюбанов О.С. Горбовий А.Ю. Філоненко М.М. Погорєловська І.Д. Суховецька Л.П.

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	
60-69	D	3 (задовільно)
50-59	E	
35-49	FX	2 (незадовільно)

Факультет фінансів та цифрових технологій
Кафедра комп'ютерних та інформаційних технологій і систем
Галузь знань 12 «Інформаційні технології»
Спеціальність 122 «Комп'ютерні науки»
Освітня програма «Інформаційні управляючі системи і технології в економіці»

Затверджую
Завідувач кафедри _____ Філоненко М.М.
«11» грудня 2024

Завдання на кваліфікаційну роботу
здобувачу вищої освіти
Мілінтєєв Даніїл Дмитрович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи Розробка децентралізованої системи взаємодії та винагород на базі блокчейну
   Затверджена наказом ректора від «10» грудня 2024 № 1852
2. Строк здачі закінченої роботи «02» січня 2026
3. Мета роботи *Розробка децентралізованої системи взаємодії та винагород на основі блокчейн-технологій для підвищення мотивації студентів.*
   Об'єкт дослідження *Процеси мотивації та фіксації навчальних досягнень студентів у цифровому освітньому середовищі.*
   Предмет дослідження *Методи гейміфікації освітнього процесу з використанням блокчейн-технологій і смарт-контрактів.*
   Орієнтовна база дослідження *Існуючі LMS-системи, освітні блокчейн-платформи та технології*
4. Вихідні дані до роботи *Наукові публікації за темою дослідження та прототипу системи освітнього IT хабу.*
5. Перелік графічного матеріалу *Архітектурні схеми системи, UML-діаграми (діаграма варіантів використання, діаграма класів), інтерфейси користувача (особистий кабінет студента, робоче місце викладача, панель адміністратора), схеми роботи смарт-контрактів та токеноміки системи.*
6. Зміст кваліфікаційної роботи (перелік питань за кожним розділом)
   Зміст кваліфікаційної роботи
   ВСТУП
   РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ДЕЦЕНТРАЛІЗОВАНИХ СИСТЕМ ТА ТЕХНОЛОГІЇ БЛОКЧЕЙН; 1.1. Особливості мотивації студентів та роль гейміфікації у сучасному навчальному процесі; 1.2. Аналіз традиційних централізованих LMS-платформ та передумови переходу до децентралізованих блокчейн-рішень; 1.3. Цифрова ідентичність користувача та еволюція взаємодії в епоху Web 3.0; РОЗДІЛ 2. РОЗРОБКА МОДЕЛІ ТА АРХІТЕКТУРИ СИСТЕМИ; 2.1. Архітектура децентралізованої системи та технологічний стек її реалізації; 2.2. Реалізація клієнт-серверної інфраструктури та інтеграційних модулів 2.3. Розробка та інтеграція смарт-контрактів і блокчейн-модулів
   РОЗДІЛ 3. ДЕЦЕНТРАЛІЗОВАНІ ОРГАНІЗАЦІЇ І ТОКЕНОМІКА В ОРГАНІЗАЦІЇ ДІЯЛЬНОСТІ НАВЧАЛЬНОГО ЗАКЛАДУ
   3.1. DAO як механізм децентралізованого управління в цифрових

екосистемах.
3.2. Токеноміка системи: емісія, розподіл та управління життєвим циклом внутрішнього токена.
3.3. Оцінювання економічного ефекту впровадження системи та перспектив її масштабування.

7. Календарний план виконання роботи

№ п/п	Назва етапів кваліфікаційноїроботи	Строк виконання етапів роботи	
		За планом	Фактично
1	2	3	4
1	Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи	До 15.12.2024	«14» 12 2024
2	**І розділ** (розкрити сутність і стан дослідження наукової проблематики роботи, після чого навести її загальну характеристику; огляд літератури за темою і вибір конкретних напрямів досліджень)	28.05.2025	«28» 05 2025
3	**ІІ розділ** (аналіз експериментальної частини про методики досліджень)	08.09.2025	«07» 09 2025
4	**ІІІ розділ** (проведені теоретичні і (або) експериментальні дослідження та проектні рішення)	17.11.2025	«19» 11 2025
5	**Висновки**	08.12.2025	«15» 12 2025
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	22.12.2025	«28» 12 2025
7	Отримання відгуку керівника	20.01.2026	«20» 01 2026
8	Отримання зовнішньої рецензії	23.01.2026	«23» 01 2026

8. Дата видачі завдання «11» грудня 2024

9. Керівник кваліфікаційної роботи _____ Редич О.В.
(підпис) (прізвище, ініціали)

10. Керівник освітньо-професійної програми _____ Омельчук А.А.
(підпис) (прізвище, ініціали)

11. Завдання до виконання отримав здобувач вищої освіти _____ Мілінтєєв Д.Д.
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

РОЗРОБКА ДЕЦЕНТРАЛІЗОВАНОЇ СИСТЕМИ ВЗАЄМОДІЇ ТА ВИНАГОРОД НА БАЗІ БЛОКЧЕЙНУ

Мета роботи полягає в розробці децентралізованої інформаційної системи взаємодії та винагород на основі блокчейн-технологій для підвищення мотивації студентів та ефективності навчального процесу.

Завданнями роботи є аналіз існуючих систем управління навчанням, дослідження гейміфікаційних підходів в освіті, обґрунтування вибору блокчейн-платформи, розробка архітектури системи, реалізація смарт-контрактів та механізмів токеноміки, а також впровадження елементів децентралізованого управління.

Предметом дослідження є методи та інструменти гейміфікації освітнього процесу з використанням блокчейн-технологій, смарт-контрактів та криптографічних токенів.

Об’єктом дослідження є процес мотивації та фіксації навчальних досягнень студентів у цифровому освітньому середовищі.

Базою дослідження виступають існуючі LMS-системи, блокчейн-платформи та розроблений прототип системи «Освітній IT хаб».

За результатами дослідження розроблено гібридну архітектуру децентралізованої освітньої системи, реалізовано смарт-контракти для винагород і цифрових сертифікатів та запропоновано модель токеноміки для підвищення залученості студентів.

Одержані результати можуть бути використані закладами вищої освіти для створення прозорого та мотивуючого цифрового навчального середовища.

Кваліфікаційна робота містить 54 сторінки, 8 рисунків, список використаних джерел з 40 найменувань, додатки.

Ключові слова: блокчейн, гейміфікація, смарт-контракти, токеноміка, DAO, Web 3.0, освітні системи.

Рік виконання кваліфікаційної роботи 2025

Рік захисту роботи 2026

ABSTRACT

DEVELOPMENT OF A DECENTRALIZED INTERACTION AND REWARD SYSTEM BASED ON BLOCKCHAIN TECHNOLOGY

The purpose of this qualification work is to develop a decentralized interaction and reward information system based on blockchain technologies aimed at increasing student motivation and improving the efficiency of the educational process.

The objectives of the work include the analysis of existing learning management systems, the study of gamification approaches in education, justification of the blockchain platform selection, development of the system architecture, implementation of smart contracts and tokenomics mechanisms, as well as the introduction of decentralized governance elements.

The subject of the research is methods and tools for gamification of the educational process using blockchain technologies, smart contracts, and cryptographic tokens.

The object of the research is the process of motivation and recording of students' academic achievements in a digital educational environment.

The research base includes existing LMS platforms, blockchain platforms, and the developed prototype of the "Educational IT Hub" system.

As a result of the research, a hybrid architecture of a decentralized educational system was developed, smart contracts for rewards and digital certificates were implemented, and a tokenomics model aimed at increasing student engagement was proposed.

The obtained results can be used by higher education institutions to create a transparent and motivating digital learning environment.

The qualification work contains 54 pages, 8 figures, a list of 40 references, and appendices.

Keywords: blockchain, gamification, smart contracts, tokenomics, DAO, Web 3.0, educational systems.

Year of qualification work completion 2025

Year of thesis defense 2026

ЗМІСТ

ВСТУП 4

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ДЕЦЕНТРАЛІЗОВАНИХ СИСТЕМ ТА ТЕХНОЛОГІЇ БЛОКЧЕЙ 8

1.1. Особливості мотивації студентів та роль гейміфікації у сучасному навчальному процесі 8

1.2. Аналіз традиційних централізованих платформ та передумови переходу до децентралізованих блокчейн-рішень 10

1.3. Цифрова ідентичність користувача та еволюція взаємодії в епоху Web 3.0 18

РОЗДІЛ 2. РОЗРОБКА МОДЕЛІ ТА АРХІТЕКТУРИ СИСТЕМИ 26

2.2. Реалізація клієнт-серверної інфраструктури та інтеграційних модулів 33

2.3. Розробка та інтеграція смарт-контрактів і блокчейн-модулів 37

РОЗДІЛ 3. ДЕЦЕНТРАЛІЗВАНІ ОРГАНІЗАЦІЇ І ТОКЕНОМІКА В ОРГАНІЗАЦІЇ ДІЯЛЬНОСТІ НАВЧАЛЬНОГО ЗАКЛАДУ 39

3.1. DAO як механізм децентралізованого управління в цифрових екосистемах 39

3.2. Токеноміка системи: емісія, розподіл та управління життєвим циклом внутрішнього токена 42

3.3. Оцінювання економічного ефекту впровадження системи та перспективи її масштабування 43

ВИСНОВКИ 47

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 50

ДОДАТКИ 55

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API – Application Programming Interface (Програмний інтерфейс додатку)

DAO – Decentralized Autonomous Organization (Децентралізована автономна організація)

ERC-20 – Ethereum Request for Comments 20 (Стандарт токенів Ethereum)

ERC-721 – Ethereum Request for Comments 721 (Стандарт NFT токенів)

LMS – Learning Management System (Система управління навчанням)

MERN – MongoDB, Express.js, React, Node.js (Стек технологій веб-розробки)

NFT – Non-Fungible Token (Невзаємозамінний токен)

SBT – Soulbound Token (Непередаваний токен)

UCN – University Coin (Університетський токен)

XP – Experience Points (Очки досвіду)

ВСТУП

Актуальність теми. Сучасна система вищої освіти переживає період глибокої трансформації [1]. Традиційні методи навчання, що базуються на пасивному сприйнятті інформації, втрачають свою ефективність в умовах стрімкого розвитку цифрових технологій [19]. Студенти, які виросли в епоху інтернету та соціальних мереж, потребують нових підходів до організації навчального процесу, що передбачають інтерактивність, миттєвий зворотний зв'язок та високий рівень залученості.

Однією з ключових проблем сучасної освіти є зниження мотивації студентів [3]. Стандартна система оцінювання часто сприймається як формальна та суб'єктивна, а отримані дипломи не завжди відображають реальні навички випускника. Крім того, існує проблема прозорості в управлінні університетськими процесами та розподілі ресурсів, що знижує довіру студентів до адміністрації.

В цьому контексті перспективним видається використання технологій гейміфікації та Web 3.0 [2, 40]. Гейміфікація дозволяє перетворити рутинний навчальний процес на захоплюючу діяльність, де кожне досягнення винагороджується, а конкуренція стимулює розвиток. Технології Web 3.0, зокрема блокчейн (див. Додаток А) та смарт-контракти, відкривають нові можливості для створення прозорих, децентралізованих систем, де кожен студент є власником своїх даних та досягнень [5, 7].

Створення гібридної освітньої платформи, що поєднує в собі елементи гейміфікації та надійність блокчейну, є актуальним науково-практичним завданням, вирішення якого дозволить підвищити якість освіти та залученість студентів.

Мета і завдання роботи. Метою дипломної роботи є розробка децентралізованої інформаційної системи взаємодії та винагород на основі

блокчейн-технологій, спрямованої на підвищення мотивації користувачів і ефективності навчального процесу.

Для досягнення поставленої мети було визначено наступні

1. Аналіз недоліків існуючих LMS щодо мотивації.
2. Дослідження гейміфікації у вищій освіті.
3. Обґрунтування вибору блокчейн-платформи.
4. Розробка гібридної архітектури системи.
5. Реалізація смарт-контрактів для токенів та досягнень.
6. Створення клієнт-серверної частини та інтерфейсу.
7. Розробка моделі токеноміки.
8. Впровадження DAO для студентського самоврядування.

Об'єкт дослідження : процес мотивації та моніторингу досягненнями студентів на основі блкчейн технологій.

Предмет дослідження: методи та інструменти гейміфікації освітнього процесу з використанням технологій блокчейн, смарт-контрактів та криптографічних токенів для підвищення залученості учнів і студентів та забезпечення прозорості академічних результатів.

Методи дослідження:

Теоретичні:

- Аналіз літератури та теорій мотивації (Десі-Райана, Маслоу, Герцберга).
- Порівняння LMS платформ (Moodle, Canvas тощо).
- Системний аналіз та UML-моделювання архітектури.

Практичні:

- Об'єктно-орієнтоване проектування.
- Розробка смарт-контрактів (Solidity, ERC-20/721).
- Веб-розробка (MERN стек).
- Криптографічний захист (ECDSA).
- Тестування (Unit, Integration).

Емпірична база:

Технічна документація: Блокчейн-платформи (Ethereum, BNB Chain, Solana) та стек розробки (OpenZeppelin, Hardhat, React, ethers.js).
Аналіз аналогів: Гейміфіковані освітні платформи (Duolingo, Classcraft) та існуючі блокчейн-рішення в освіті (MIT Digital Diplomas, Woolf University).

Практичним результатом роботи є: розроблений прототип системи «Освітній IT хаб», що включає смарт-контракти (Додатки Б, В), розгорнуті в тестовій мережі BNB Smart Chain, та веб-додаток з повнофункціональним користувацьким інтерфейсом .

Наукова новизна одержаних результатів полягає в наступному:

1. Створено модель гібридної навчальної екосистеми, яка поєднує off-chain механізми гейміфікації (для швидкодії) та on-chain інструменти фіксації досягнень (для надійності), що дозволяє вирішити проблему масштабованості блокчейн-рішень у навчанні.

2. Удосконалено метод оцінювання навчальних досягнень шляхом введення подвійної системи винагород (бали досвіду XP та токени корисності), що дозволяє розділити академічний прогрес та економічні стимули.

3. Запропоновано концепцію студентського самоврядування через адаптацію механізмів DAO та квадратичного голосування до специфіки освітнього середовища.

Практичне значення одержаних результатів. Розроблена система дозволяє навчальним закладам (НЗ) створити прозоре та мотивуюче середовище для студентів. Впровадження системи сприятиме підвищенню відвідуваності занять, покращенню успішності та активізації участі студентів у громадському житті НЗ. Отримані результати можуть бути використані при модернізації існуючих LMS (Learning Management Systems) або створенні нових освітніх платформ.

Апробація результатів. Основні положення роботи були представлені на наміжнародній науково-практичній конференції : Мілінтєєв Д., Редич О. (НК) Гейміфікована система мотивації з викоритсанням криптогаманця //Економіка і бізнес: нові реалії, технологічні тренди, цифрові інновації. Economy and business:

new realities, technological trends, digital innovations (Львів, 15 травня 2025 р.). Збірник тез І міжн. наук.-практ. конф. ЛНУП, 2025.

Структура та обсяг роботи. Робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. Основний обсяг роботи становить 54 сторінки та додатки на 23 сторінках.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ДЕЦЕНТРАЛІЗОВАНИХ СИСТЕМ ТА ТЕХНОЛОГІЇ БЛОКЧЕЙ

1.1. Особливості мотивації студентів та роль гейміфікації у сучасному навчальному процесі

Ефективність освітнього процесу у вищій школі безпосередньо залежить від рівня вмотивованості студентів [3, 19]. У сучасній педагогічній психології мотивація розглядається не просто як бажання вчитися, а як складний динамічний процес, що визначає спрямованість, інтенсивність та наполегливість навчальної діяльності. Традиційні методи стимулювання, такі як бальна система оцінювання та адміністративний контроль відвідуваності, демонструють зниження ефективності в роботі з сучасним поколінням студентів. Згідно з теорією самодетермінації (Self-Determination Theory), розробленою Е. Десі та Р. Райаном [3], для формування стійкої внутрішньої мотивації необхідно задовольнити три базові психологічні потреби особистості: 1. Потреба в автономії (Autonomy): Студент повинен відчувати, що він сам керує своїм навчанням, має право вибору дисциплін, тем для досліджень або темпу засвоєння матеріалу. 2. Потреба в компетентності (Competence): Відчуття власної спроможності виконувати завдання та досягати успіху. Це досягається через чіткий та своєчасний зворотний зв'язок (feedback). 3. Потреба у приналежності (Relatedness): Відчуття зв'язку з академічною спільнотою, групою, викладачами Гейміфікація (Gamification) – це використання ігрових елементів і механік у неігрових контекстах [2].

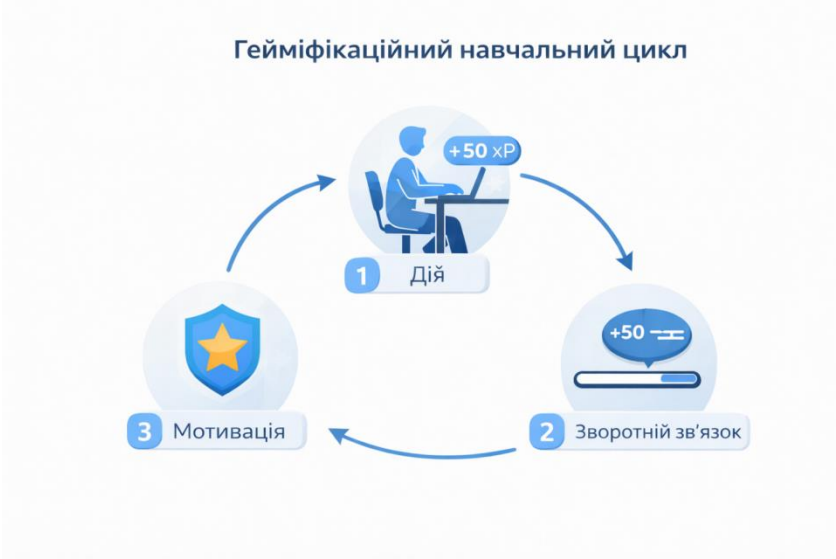


Рис. 1.1. Цикл мотивації студента в гейміфікованій системі.

Вона виступає потужним інструментом для задоволення вищезгаданих потреб. На відміну від простого використання ігор (Game-Based Learning), гейміфікація інтегрує ігрову логіку в саму структуру навчального процесу. Автономія через нелінійність. У гейміфікованій системі студент може обирати траєкторію навчання, подібно до того, як гравець обирає квести в рольовій грі (RPG). Замість жорсткого розкладу "лекція -> семінар -> залік", пропонується набір активностей (обов'язкових та варіативних), виконання яких приносить бали досвіду (XP). Це дає студенту відчуття контролю над власним прогресом.

Компетентність через миттєвий зворотний зв'язок. Традиційна оцінка за семестр виставляється із запізненням, коли виправити ситуацію вже складно. Гейміфікація пропонує миттєву реакцію системи: завантажив роботу – отримав бали, пройшов тест – отримав nft бейдж. Візуалізація прогресу (Progress Bars) дозволяє студенту постійно бачити, на якому етапі він знаходиться і скільки зусиль потрібно докласти для досягнення наступного рівня. Приналежність через соціальні механіки. Рейтингові таблиці (Leaderboards), командні завдання та спільні досягнення формують здорове конкурентне середовище та відчуття спільноти. Важливо, щоб конкуренція не була демотивуючою для "слабших" студентів. Для цього використовуються відносні рейтинги (наприклад, "Топ-10 тижня") або командні змагання, де внесок кожного є важливим. Аналіз наукової

літератури показує, що впровадження гейміфікації сприяє підвищенню відвідуваності занять на 15-20%, а також значно покращує якість виконання самостійних робіт. Однак, існує ризик "зовнішньої мотивації", коли студент вчиться лише заради балів. Щоб уникнути цього, система винагород повинна бути збалансованою і включати не лише віртуальні нагороди, але й змістовні привілеї (наприклад, доступ до ексклюзивних матеріалів, зустрічі з менторами).

1.2. Аналіз традиційних централізованих платформ та передумови переходу до децентралізованих блокчейн-рішень

На сьогоднішній день більшість закладів вищої освіти використовують системи управління навчанням (LMS), такі як Moodle, Google Classroom, Canvas або Blackboard. Було проведено детальний аналіз функціональних можливостей цих платформ з точки зору мотивації та прозорості.

Порівняльний аналіз сучасних LMS платформ;

Для об'єктивної оцінки було проведено багатокритеріальний аналіз; найпопулярніших LMS систем за наступними параметрами: вартість впровадження, технічна масштабованість, наявність інструментів гейміфікації, безпека даних та можливість інтеграції з зовнішніми сервісами.

Таблиця 1.1.

Порівняльна характеристика LMS платформ

Критерій	Moodle	Google Classroom	Canvas	Blackboard	Запропонована система
Вартість	Безкоштовно (Open Source)	Безкоштовно	$8–12/студент/рік	$10–15/студент/рік	Безкоштовно (Open Source)
Масштабованість	До 10,000 користувачів	Необмежена (хмара Google)	До 50,000	До 100,000	Необмежена (блокчейн + хмара)
Гейміфікація	Плагіни (обмежено)	Відсутня	Базова (бейджі)	Відсутня	Повна (XP, токени, DAO)
Безпека даних	Середня (ризик SQL injection)	Висока (Google Security)	Висока	Висока	Висока (MongoDB + Web3)
Прозорість оцінок	Низька (можливі маніпуляції)	Низька	Низька	Низька	Абсолютна (immutable ledger)
Мобільний додаток	Є (застарілий)	Є (відмінний)	Є (добрий)	Є (середній)	PWA (планується)
API для інтеграцій	REST API	Google API	REST API	REST API	REST + Web3 API
Підтримка блокчейну	Ні	Ні	Ні	Ні	Так (BNB Smart Chain)

Складність налаштування	Висока (потрібен IT-фахівець)	Низька	Середня	Висока	Середня

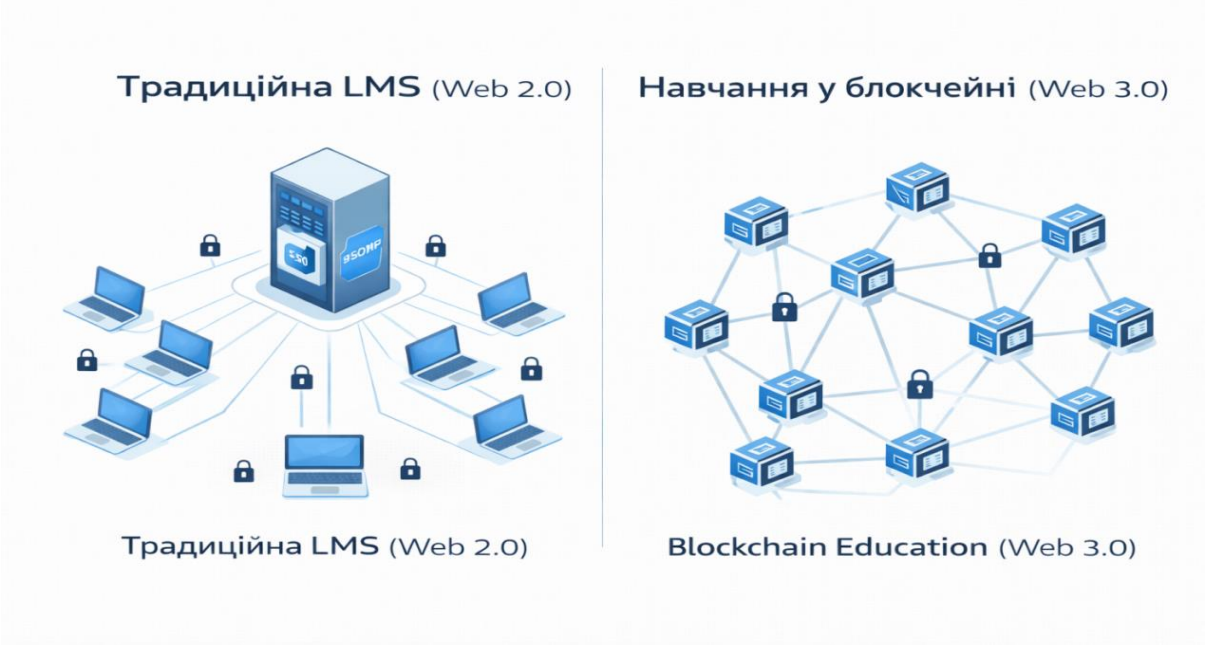


Рис. 1.2 . Архітектурна відмінність традиційної LMS від блокчейн-платформи.

Детальний аналіз платформ:

Moodle – найпоширеніша Open Source платформа.

Переваги: Безкоштовність, величезна кількість плагінів (понад 1,700), гнучкість налаштувань, підтримка SCORM та xAPI стандартів.

Недоліки: Застарілий інтерфейс (UI/UX), що нагадує форуми 2000-х років; складність у налаштуванні гейміфікації (потребує встановлення додаткових модулів, які часто конфліктують між собою); централізована база даних MySQL/PostgreSQL, що робить її вразливою до маніпуляцій з боку адміністраторів (теоретична можливість зміни оцінок "заднім числом"); проблеми з продуктивністю при понад 5,000 одночасних користувачів.

Google Classroom – зручний інструмент для організації документообігу.

Переваги: Інтеграція з екосистемою Google (Drive, Docs, Meet), простота використання, автоматичне резервне копіювання, відсутність витрат на хостинг.

Недоліки: Відсутність повноцінних інструментів аналітики та гейміфікації; це скоріше файлообмінник з функцією оцінювання, ніж повноцінне навчальне середовище; залежність від Google (ризики конфіденційності, можливі блокування доступу); обмежені можливості кастомізації інтерфейсу.

Canvas – сучасна LMS з непоганим дизайном.

Переваги: Мобільний додаток з відмінним UX, потужна аналітика (Canvas Analytics), підтримка відеоконференцій, інтеграція з понад 400 освітніми інструментами.

Недоліки: Висока вартість ліцензії ($8-12 на студента на рік, що для університету з 10,000 студентів становить $80,000-120,000 щорічно); закритий вихідний код (vendor lock-in); дані студентів зберігаються на серверах компанії Instructure (США), що створює ризики конфіденційності та юридичні проблеми з GDPR.

Blackboard – корпоративне рішення для великих університетів.

Переваги: Масштабованість (підтримка понад 100,000 користувачів), сертифікація безпеки (ISO 27001), професійна технічна підтримка 24/7.

Недоліки: Найвища вартість на ринку ($10-15 на студента), складний процес впровадження (потребує 6-12 місяців), застарілий інтерфейс, повільні оновлення.

Проблеми масштабованості, безпеки та централізації в традиційних цифрових системах.

Спільним недоліком усіх традиційних LMS є централізація даних. Історія успішності студента, його дипломи та сертифікати зберігаються в локальній базі даних університету. Це створює ряд критичних проблем:

1. Ризик фальсифікації та корупції:

Адміністратор бази даних має технічну можливість змінити будь-яку оцінку. У 2019 році в одному з університетів США було виявлено скандал, коли системний адміністратор за хабар змінював оцінки студентів безпосередньо в базі даних Blackboard. Відсутність криптографічного підпису робить такі маніпуляції практично невиявними.

2. Проблема верифікації дипломів: Роботодавець не може миттєво перевірити справжність диплома. Згідно з дослідженням HireRight (2020), 85% роботодавців виявляють неточності в резюме кандидатів, включно з фальшивими дипломами. Процес верифікації через офіційний запит в університет займає 2-4 тижні, що сповільнює процес найму.

3. Втрата даних та проблема довговічності: У випадку технічного збою (пожежа в серверній, кібератака з шифруванням даних) або ліквідації університету, цифрова історія студента може бути втрачена назавжди. Резервні копії (backups) вирішують проблему лише частково, оскільки вони також централізовані.

4. Відсутність інтероперабельності (сумісності між системами): При переведенні студента в інший ВНЗ або при визнанні дипломів за кордоном виникає проблема перенесення даних. Кожна LMS використовує власний формат даних, що робить автоматичний обмін неможливим. Перезарахування дисциплін відбувається вручну через паперові довідки, що є анахронізмом у XXI столітті.

5. Проблема масштабованості під час пікових навантажень: Традиційні LMS побудовані за клієнт-серверною архітектурою. Під час сесії, коли тисячі студентів одночасно намагаються здати тести, сервер часто не витримує навантаження. Дослідження показують, що Moodle починає деградувати при понад 500 одночасних запитах до бази даних, що вимагає дорогого апгрейду серверного обладнання.

Впровадження технологій блокчейн (Blockchain) дозволяє вирішити ці проблеми кардинально. Блокчейн виступає як "незмінний реєстр" (Immutable Ledger).

Довіра: Оцінка, записана в блокчейн, не може бути змінена або видалена ніким, навіть адміністратором НЗ або університету.

Прозорість: Будь-хто може перевірити наявність запису (транзакції), не маючи доступу до персональних даних.

Власність: Студент володіє своїми досягненнями (у вигляді токенів або NFT) через свій криптогаманець, незалежно від сервера університету.

Таким чином, перехід від Web 2.0 (централізовані LMS) до Web 3.0 (децентралізовані освітні екосистеми) є логічним етапом еволюції освітніх технологій. Це не просто зміна технологічного стеку, це зміна філософії взаємодії між НЗ, університетом та студентом: від відносин "контролер – підлеглий" до відносин "партнер – партнер".

SWOT-аналіз застосування блокчейн-технологій у децентралізованих екосистемах.

Для оцінки доцільності використання блокчейну в освітньому процесі було проведено SWOT-аналіз.

Таблиця 1.2.

SWOT-аналіз блокчейну в освіті

Strengths (Сильні сторони)	**Weaknesses (Слабкі сторони)**
Незмінність записів (дипломи, оцінки)	Високий поріг входу для користувачів
Прозорість операцій	Вартість транзакцій (gas fees)
Децентралізація (немає єдиної точки відмови)	Складність розробки та підтримки
Міжнародне визнання (портативність дипломів)	Регуляторна невизначеність
Автоматизація через смарт-контракти	Екологічні питання (PoW блокчейни)

Стратегії на основі SWOT:

SO-стратегія (Сильні сторони + Можливості): Використати незмінність блокчейну для створення міжнародно визнаних цифрових сертифікатів.

WO-стратегія (Слабкі сторони + Можливості): Зменшити поріг входу через інтеграцію з популярними гаманцями (MetaMask, Trust Wallet).

ST-стратегія (Сильні сторони + Загрози): Використати смарт-контракти для автоматичного хеджування волатильності токенів.

WT-стратегія (Слабкі сторони + Загрози): Провести аудит безпеки смарт-контрактів та створити освітні матеріали для користувачів.

Психологічні типи студентів та адаптація гейміфікованих систем.

Ефективність гейміфікації значною мірою залежить від того, наскільки запропоновані механіки відповідають психотипу користувача. У 1996 році Річард Бартл запропонував класифікацію гравців, яка сьогодні є стандартом при

проектуванні гейміфікованих систем. В контексті вищої освіти цю типологію можна адаптувати наступним чином: Накопичувачі (Achievers): Мотивація: Отримання статусу, накопичення балів, закриття "гештальтів". Поведінка в навчанні: Прагнуть отримати максимальний бал, виконати всі завдання, навіть необов'язкові. Механіки для них: Прогрес-бари, рівні (Levels), складні досягнення ("Здати всі лаби на 100 балів"), публічні профілі з нагородами. Дослідники (Explorers): Мотивація: Відкриття нового, вивчення прихованих можливостей системи, розуміння глибинних механізмів. Поведінка в навчанні: Цікавляться додатковими матеріалами, люблять нестандартні задачі, часто знаходять помилки в курсах. Механіки для них: Приховані квести ("Easter eggs"), доступ до ексклюзивного контенту за досягнення, нелінійні траєкторії навчання.

Соціалізатори (Socializers): Мотивація: Спілкування, взаємодія з іншими, відчуття приналежності до групи. Поведінка в навчанні: Активні в чатах, люблять групові проекти, допомагають іншим. Механіки для них: Чати, форуми, система менторства, колективні квести (де результат залежить від усієї групи), подарунки (можливість передати токени другу). Кілери (Killers) / Конкуренти: Мотивація: Перемога над іншими, домінування, демонстрація своєї переваги. Поведінка в навчанні: Прагнуть бути першими в рейтингу, люблять олімпіади та хакатони. Механіки для них: Таблиці лідерів (Leaderboards), PvP-дуелі (інтелектуальні батли), рейтингові списки. Система «Освітній IT Хаб» проектується як збалансована екосистема, що містить стимули для всіх чотирьох типів. Наприклад, введення DAO задовольняє Соціалізаторів (спільне управління) та Дослідників (вивчення складних правил голосування), тоді як токеноміка приваблює Накопичувачів та Конкурентів.

Порівняльний аналіз гейміфікаційних механік взаємодії користувачів

При проектуванні системи важливо розуміти різницю у впливі різних ігрових елементів на мотивацію. Таблиці лідерів (Leaderboards): Переваги: Створюють потужний стимул для конкурентних студентів ("Кілерів"). Чітко візуалізують успіх.

Недоліки: Можуть демотивувати студентів, які знаходяться внизу списку ("Я ніколи їх не наздожену"). Створюють токсичну атмосферу, якщо конкуренція стає занадто жорсткою. Рішення в системі: Використання "локальних" рейтингів (наприклад, "Топ-10 тижня" або "Топ групи"), щоб кожен мав шанс потрапити в лідери. Бейджі (Badges / NFT): Переваги: Фіксують особисті досягнення, не порівнюючи студента з іншими. Працюють на внутрішню мотивацію ("Я молодець, я це зробив"). Залишаються назавжди (особливо у вигляді SBT).
Недоліки: Якщо бейджів занадто багато і вони даються за тривіальні дії (наприклад, "Зайшов у систему"), їх цінність девальвується. Рішення в системі: Використання рідкісних SBT, які видаються лише за значні досягнення (наприклад, "Наукова публікація" або "Волонтер року").

Таким чином, комбінація цих механік дозволяє створити гнучку систему мотивації, яка не тисне на студента, а підтримує його індивідуальний шлях розвитку.

1.3. Цифрова ідентичність користувача та еволюція взаємодії в епоху Web 3.0

Статистика та тенденції Lifelong Learning

За даними Європейської Комісії (Eurostat, 2023), участь дорослого населення (25-64 роки) у програмах навчання впродовж життя в ЄС становить:

Таблиця 1.3.

Участь у Lifelong Learning в Європі (2023)

Країна	**Відсоток населення**	**Тренд (2018–2023)**
Швеція	34.7%	+5.2%
Фінляндія	31.2%	+4.8%

Нідерланди	28.5%	+3.1%
Середнє по ЄС	12.9%	+2.3%
Україна	7.2%	+1.5%

Ключові висновки: Україна значно відстає від середнього показника ЄС (7.2% vs 12.9%) Зростання інтересу до онлайн-навчання після пандемії COVID-19 (+40% у 2020-2023) 78% роботодавців вважають навички важливішими за формальні дипломи (LinkedIn, 2023)

Роль цифрової ідентичності:

Блокчейн свідоцтва та сертифікати та SBT (Soulbound Tokens) дозволяють створити портативне портфоліо навичок, яке: Визнається міжнародно; Не залежить від існування університету; Містить не лише дипломи, але й мікрокредити, сертифікати, досягнення.

Децентралізована ідентичність (DID) та Verifiable Credentials

Концепція Decentralized Identifiers (DID), стандартизована W3C у 2022 році, передбачає створення цифрової ідентичності, яка: Належить користувачу (не університету чи державі); Портативна (можна використовувати на різних платформах); Перевіряєма (будь-хто може перевірити автентичність); Приватна (користувач контролює, кому показувати дані)

Архітектура DID для освіти:

`Студент (власник DID) → Університет (емітент Verifiable Credentials) → Блокчейн (реєстр) → Роботодавець (верифікатор).`



Рис. 1.3. Схема отримання верифікованого сертифікату (SBT)

Приклад використання: Студент отримує сертифікат,— як Verifiable Credential; Сертифікат зберігається в його криптогаманці; При працевлаштуванні студент надає доступ роботодавцю. Роботодавець перевіряє автентичність через блокчейн (без звернення до університету)

Переваги: Миттєва верифікація (замість тижнів очікування довідок); Захист від підробок (неможливо підробити запис у блокчейні); Приватність (студент сам вирішує, що показувати).

Сучасний ринок праці характеризується високою динамікою змін. Знання, отримані в університеті, застарівають в середньому за 3-5 років. Це зумовлює необхідність переходу до парадигми "Навчання протягом життя" (Lifelong Learning). У цій парадигмі освіта не завершується отриманням диплома, а є безперервним процесом накопичення навичок, сертифікатів та досвіду.

Однак, існуюча система документації освіти не відповідає вимогам Lifelong Learning. Людина, яка проходить десятки курсів на Coursera, Udemy, бере участь у хакатонах та волонтерських проектах, не має єдиного "цифрового паспорта", який би агрегував усі ці досягнення. Її резюме – це лише текстовий файл, який роботодавець змушений перевіряти вручну. Self-Sovereign Identity

(SSI) – суверенної особистості. В контексті освіти це означає, що студент сам володіє своїми даними та керує доступом до них. Технологічною основою для цього стають:

1. Verifiable Credentials (VC): Стандарт W3C для цифрових посвідчень. Це криптографічно підписані твердження (наприклад, "Університет X підтверджує, що Студент Y здав курс Z на відмінно").

2. Soulbound Tokens (SBT): Непередавані токени в блокчейні. Запропоновані Віталіком Бутеріним, вони ідеально підходять для фіксації репутації. На відміну від NFT, які можна продати, SBT назавжди прив'язані до гаманця власника. Це унеможливлює "купівлю диплома" на вторинному ринку.

Впровадження цифрової ідентичності на базі Web 3.0 дозволяє створити "Академічний паспорт" або «Паспорт професійних навичок» , який містить: Офіційні дипломи та ступені або Сертифікати; Мікро-кредити за короткострокові курси; Підтвердження Soft Skills (наприклад, "Командна робота", підтверджена колегами по проекту). Історію участі в наукових конференціях та громадському житті. Такий підхід вигідний усім учасникам процесу:

Студентам: Вони отримують портфоліо, яке неможливо загубити і яке визнається автоматично.

Університетам: Зменшується навантаження на адміністративний персонал (не потрібно видавати паперові довідки), підвищується престиж бренду як інноваційного закладу.

Роботодавцям: Автоматизований скринінг кандидатів. Замість того, щоб читати сотні резюме, HR-менеджер може зробити запит до блокчейну: "Знайти всіх кандидатів, які мають SBT 'Python Developer' та SBT 'English B2'".

Порівняльний аналіз блокчейн-платформ для побудови децентралізованих сервісів

Вибір блокчейн-платформи для реалізації освітньої системи є критично важливим рішенням, яке впливає на вартість, швидкість та масштабованість

проекту. Було проведено детальний порівняльний аналіз найпопулярніших блокчейн-мереж з точки зору їх придатності для освітніх додатків.

Ethereum (ETH) – Піонер смарт-контрактів. Ethereum є найбільш зрілою платформою для розробки децентралізованих додатків (dApps). Його екосистема включає величезну кількість інструментів розробки, бібліотек та готових рішень.

Переваги: Найбільша спільнота розробників та найкраща документація; Максимальна децентралізація (понад 8,000 валідаторів після переходу на Proof-of-Stake). Стандарти ERC-20, ERC-721, ERC-1155 є де-факто промисловими стандартами. Висока безпека: мережа працює з 2015 року без критичних збоїв.

Недоліки: Висока вартість транзакцій: В періоди пікового навантаження Gas fee може досягати $50-100 за одну транзакцію. Для освітньої системи, де студент може виконувати десятки дій на день (здача тестів, отримання балів), це економічно нежиттєздатно. Низька швидкість: 15-30 транзакцій на секунду (TPS), час підтвердження блоку 12-15 секунд. Це створює затримки в інтерфейсі, що погіршує User Experience.

Складність для новачків: Студентам потрібно мати ETH на балансі навіть для отримання безкоштовних токенів (парадокс "потрібні гроші, щоб отримати гроші").

BNB Smart Chain (BSC) – Баланс між швидкістю та вартістю; BNB Smart Chain є EVM-сумісною мережею (тобто підтримує той самий код Solidity, що й Ethereum), але з іншим механізмом консенсусу – Proof of Staked Authority (PoSA).

Переваги: Низька вартість: Середня комісія за транзакцію становить $0.10-0.50, що на два порядки дешевше за Ethereum. Висока швидкість: ~60-100 TPS, час блоку 3 секунди. Це забезпечує плавну роботу інтерфейсу. Сумісність з Ethereum: Весь код, написаний для Ethereum, працює на BSC без змін. Це дозволяє використовувати перевірені бібліотеки OpenZeppelin. Розвинена інфраструктура: Підтримка MetaMask, Trust Wallet, наявність DEX (PancakeSwap) для обміну токенів.

Недоліки: Менша децентралізація: Лише 21 валідатор, що робить мережу більш вразливою до цензури (теоретично). Залежність від екосистеми Binance:

Якщо біржа Binance зіткнеться з регуляторними проблемами, це може вплинути на мережу.

Polygon (MATIC) – Layer 2 рішення для Ethereum Polygon працює як "бічний ланцюг" (sidechain) для Ethereum, забезпечуючи швидкі та дешеві транзакції з можливістю періодичного "якорення" стану в основну мережу Ethereum.

Переваги: Дуже низька вартість (Gas fee \u003c $0.01). Швидкість до 7,000 TPS. Успадковує безпеку Ethereum через механізм Checkpoints.

Недоліки: Додаткова складність архітектури (потрібен Bridge для переміщення активів між Ethereum та Polygon). Менша зрілість екосистеми порівняно з Ethereum або BSC.

Solana (SOL) – Високопродуктивна альтернатива

Solana використовує унікальний механізм консенсусу Proof of History (PoH), що дозволяє досягти надзвичайно високої пропускної здатності.

Переваги: Швидкість до 65,000 TPS (теоретично). Низькі комісії (\u003c $0.001). Сучасна архітектура, оптимізована для DeFi та NFT.

Недоліки: Нестабільність: Мережа неодноразово зазнавала повних зупинок (downtime) через технічні проблеми. Інша мова програмування: Смарт-контракти пишуться на Rust, а не Solidity, що вимагає додаткового навчання. Менша екосистема інструментів порівняно з EVM-сумісними мережами.

Обґрунтування вибору BNB Smart Chain Для реалізації системи "Освітній IT Хаб" було обрано BNB Smart Chain з наступних причин:

1. Економічна доцільність: Низькі комісії дозволяють студентам активно взаємодіяти з системою без фінансового бар'єру.

2. Сумісність з Ethereum: Можливість використання перевірених бібліотек OpenZeppelin та легка міграція на Ethereum у майбутньому, якщо це стане необхідним.

3. Баланс децентралізації та продуктивності: Хоча BSC менш децентралізована за Ethereum, для освітньої системи (де критична загроза – це не цензура, а доступність) цей компроміс є прийнятним.

4. Зрілість екосистеми: Наявність готових рішень для інтеграції гаманців, експлорерів блоків (BscScan) та документації. Таким чином, BNB Smart Chain забезпечує оптимальне співвідношення вартості, швидкості та надійності для масового використання в університетському середовищі.

Моделі токеноміки в гейміфікованих децентралізованих системах

Впровадження токеноміки в освіту вимагає ретельного проектування економічної моделі, щоб уникнути типових пасток, таких як гіперінфляція, спекулятивні бульбашки або втрата мотивації через знецінення винагород. Класифікація токенів за функціональним призначенням:

1. Utility Tokens (Утилітарні токени) Призначення: Внутрішня валюта для оплати послуг та товарів в екосистемі. Приклад у системі: UniversityCoin (UCN) – використовується для покупки мерчу, оплати додаткових курсів, участі в голосуваннях. Характеристики: Ліквідні (можна передавати), підлягають інфляції (емісія при нарахуванні винагород).

2. Security Tokens (Токени цінних паперів) Призначення: Представляють частку власності або право на дивіденди. Застосування в освіті: Теоретично можливе для інвестиційних фондів університету, але вимагає регуляторного схвалення. У даній роботі не використовуються через юридичну складність.

3. Soulbound Tokens (Непередавані токени) Призначення: Фіксація репутації, досягнень, ідентичності. Приклад у системі: SoulboundDiploma – цифровий диплом, який неможливо продати або передати. Характеристики: Неліквідні, не підлягають інфляції (кожен токен унікальний). Механізми контролю інфляції: Основна загроза для утилітарного токена – це гіперінфляція, коли емісія нових токенів значно перевищує їх спалювання (burn) або виведення з обігу. Це призводить до знецінення: якщо раніше за 100 UCN можна було купити каву, то через місяць – лише цукерку. Для запобігання цьому в системі передбачено: *Обмеження емісії:* Токени створюються виключно через смарт-контракт при виконанні академічних завдань. Максимальна кількість токенів на одного студента за семестр обмежена (наприклад, 1000 UCN). Емісія прив'язана до реальної цінності: складніше завдання = більше токенів. *Механізми

спалювання (Deflationary Pressure):* Burn при покупці: 50% токенів, витрачених у магазині, спалюються назавжди. Burn при голосуванні: Участь у DAO коштує токенів, які виводяться з обігу. Vesting (блокування): Токени, зароблені студентом, стають доступними для виведення на зовнішній гаманець лише після закінчення семестру. Це запобігає миттєвому продажу на біржі. *Динамічне коригування:* Система моніторить співвідношення Total Supply (загальна кількість токенів) до Active Users (активних користувачів). Якщо інфляція перевищує 20% на місяць, автоматично зменшуються винагороди за рутинні завдання.

Психологічні аспекти токеноміки: Дослідження в галузі поведінкової економіки показують, що сприйняття цінності токена залежить не лише від його купівельної спроможності, але й від: Рідкісності (Scarcity): Обмежені за кількістю SBT (наприклад, "Студент року") мають вищу психологічну цінність, ніж необмежені UCN. Соціального визнання (Social Proof): Публічні рейтинги власників рідкісних токенів створюють престиж. Відчуття прогресу (Progress): Візуалізація накопичення токенів (прогрес-бари, графіки) посилює мотивацію.

Таким чином, успішна токеноміка – це не лише математична модель, але й психологічний дизайн, що враховує особливості людської поведінки.

Проведений аналіз показав, що традиційна система вищої освіти стикається з кризою мотивації та застарілістю інструментів управління даними. Існуючі LMS (Moodle, Canvas) вирішують лише завдання доставки контенту, але не створюють стимулюючого середовища. Гейміфікація, підкріплена технологіями Web 3.0, пропонує комплексне вирішення цих проблем. Вона дозволяє: Підвищити внутрішню мотивацію через задоволення потреб в автономії та компетентності. Забезпечити прозорість та довіру до академічних результатів через блокчейн. Створити єдину цифрову ідентичність студента, що відповідає вимогам сучасного ринку праці.

РОЗДІЛ 2. РОЗРОБКА МОДЕЛІ ТА АРХІТЕКТУРИ СИСТЕМИ

2.1. Архітектура децентралізованої системи та технологічний стек її реалізації

Концептуальна модель системи "Освітнього IT Хабу" та обґрунтування гібридної архітектури. Проектування сучасної освітньої платформи вимагає вирішення дилеми між зручністю використання (User Experience) та надійністю зберігання даних. Традиційні централізовані системи забезпечують високу швидкість роботи, але мають ризики маніпуляцій даними. Повністю децентралізовані додатки (dApps) гарантують безпеку, але часто є повільними та дорогими у використанні через комісії мережі (Gas fees).

Для вирішення цієї проблеми в даній роботі запропоновано гібридну архітектуру, яка поєднує найкращі практики Web 2.0 та Web 3.0. Концептуальна модель системи "Цифрового IT Хабу" базується на чіткому розділенні потоків даних на два рівні:

1. Операційний рівень (Off-chain):

Цей рівень відповідає за обробку динамічного контенту та взаємодію з користувачем у реальному часі. Тут відбуваються процеси, які не вимагають фінансової або юридичної фіксації: Завантаження навчальних матеріалів (відео, PDF); Спілкування в чатах та форумах; Проходження поточних тестів; Відображення розкладу занять. Технічно це реалізується через класичний клієнт-серверний додаток (MERN Stack), що забезпечує миттєву реакцію інтерфейсу (latency < 100ms) та відсутність плати за кожну дію.

2. Рівень довіри (On-chain): Цей рівень використовується виключно для фіксації критично важливих подій та активів. Сюди потрапляють дані, які мають довгострокову цінність і не повинні бути змінені: Підсумкові оцінки за семестр; Отримані дипломи та сертифікати (SBT); Баланс токенів мотивації (ERC-20); Результати голосувань студентського самоврядування (DAO); Використання блокчейну (BNB Smart Chain) на цьому рівні гарантує, що навіть адміністратор системи не зможе "підробити" диплом або "накрутити" голоси.

Рольова модель системи передбачає чотири основні групи користувачів, кожна з яких має свій набір прав та інтерфейсів:

Студент (Student): Основний споживач послуг. Має доступ до "Особистого кабінету", де бачить свій прогрес (XP), баланс токенів, доступні квести (лабораторні роботи) та магазин винагород. Студент може ініціювати транзакції (голосування, покупка товарів), підписуючи їх своїм криптогаманцем.

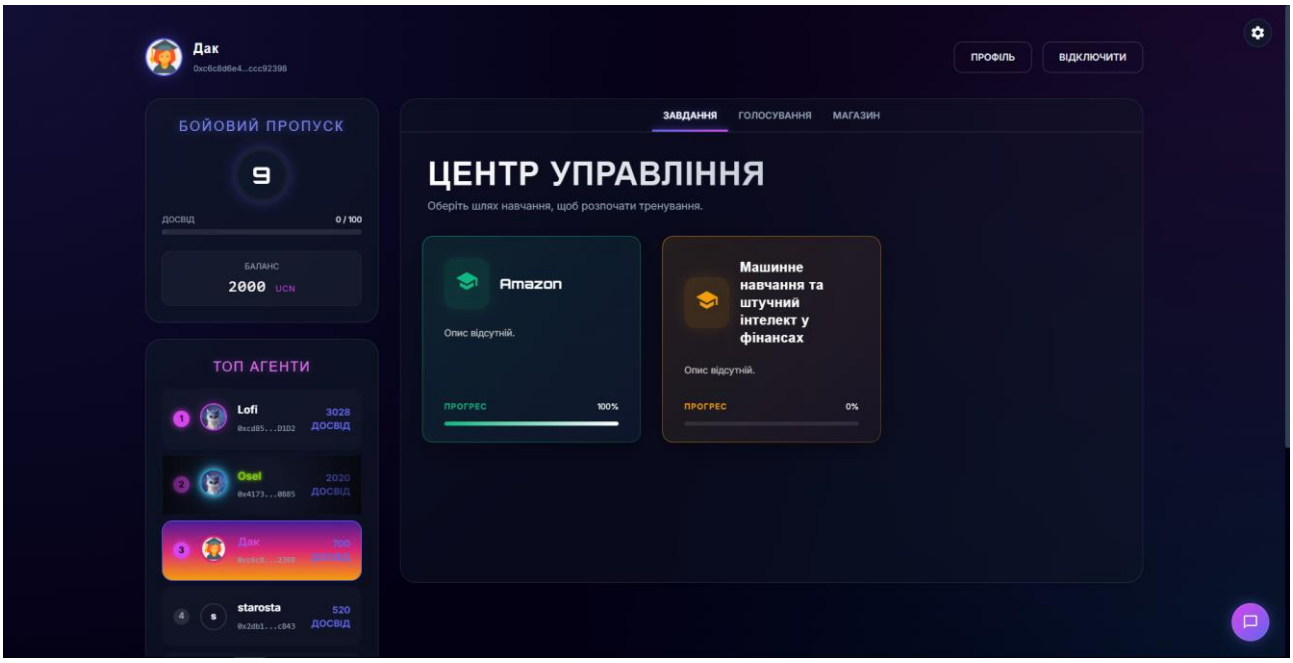


Рис.2.4 .Інтерфейс особистого кабінету студента (режим "Mission Control").

Викладач (Teacher): Відповідає за створення контенту (квестів) та валідацію результатів. Викладач перевіряє завантажені роботи і, у разі успіху, ініціює нарахування балів. Важливо, що викладач не має прямого доступу до зміни балансу токенів – це робить смарт-контракт автоматично на основі

виставленої оцінки.

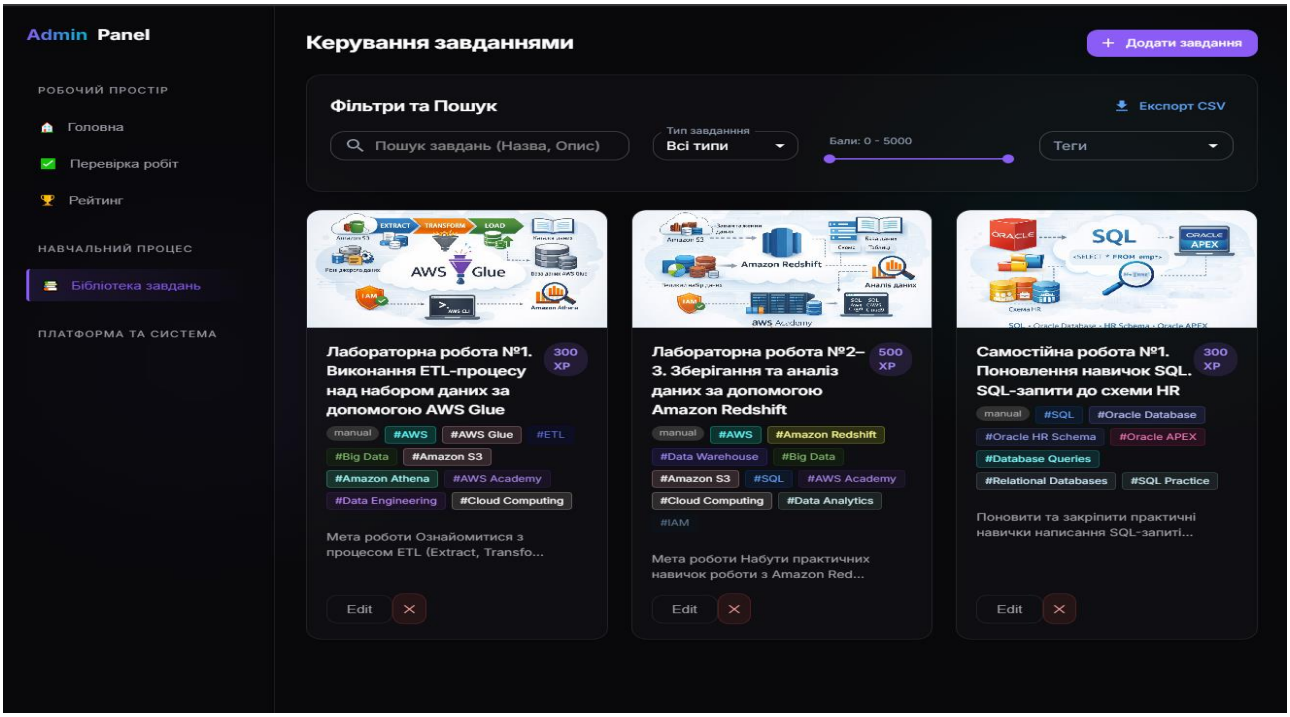


Рис.2.5 — Робоче місце викладача для управління навчальним контентом.

Адміністратор (Admin): Технічна підтримка системи. Відповідає за працездатність серверів та безпеку, але (завдяки архітектурі) не має доступу до приватних ключів користувачів.

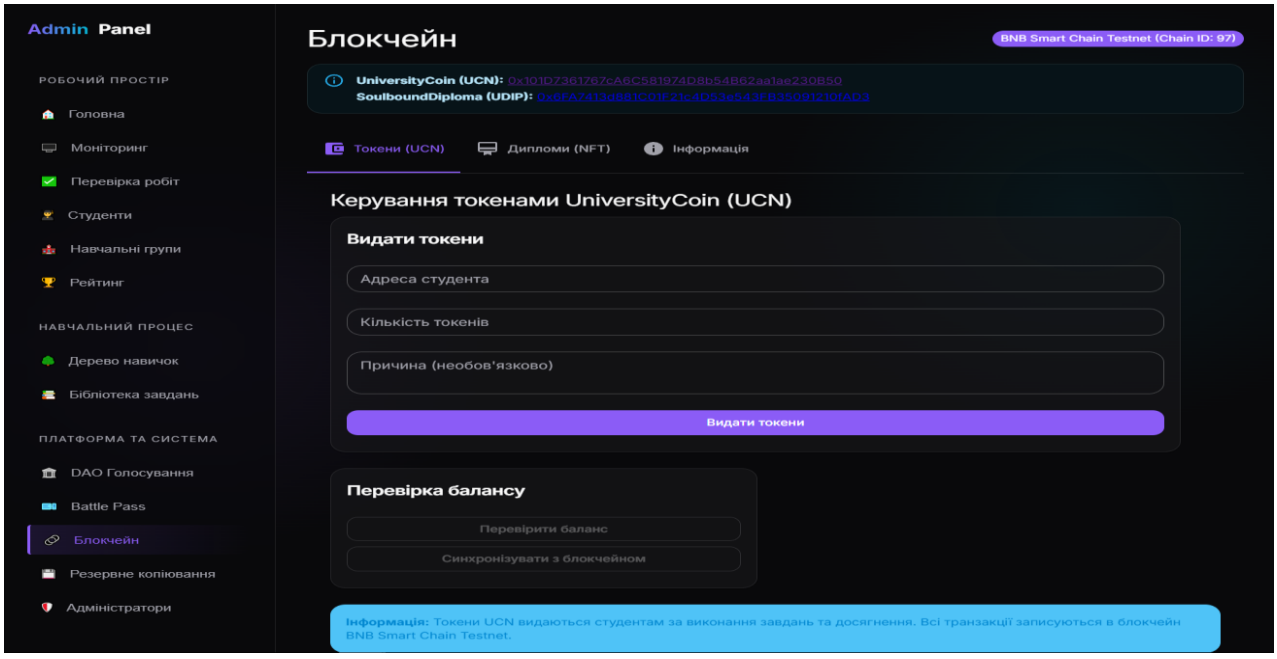


Рис. 2.6. Панель адміністратора для моніторингу системи та користувачів.

UML-діаграма варіантів використання (Use Case Diagram)

Для формалізації функціональних вимог до системи було розроблено діаграму варіантів використання за стандартом UML 2.5, яка ілюструє взаємодію акторів з основними функціями системи.

Основні варіанти використання:

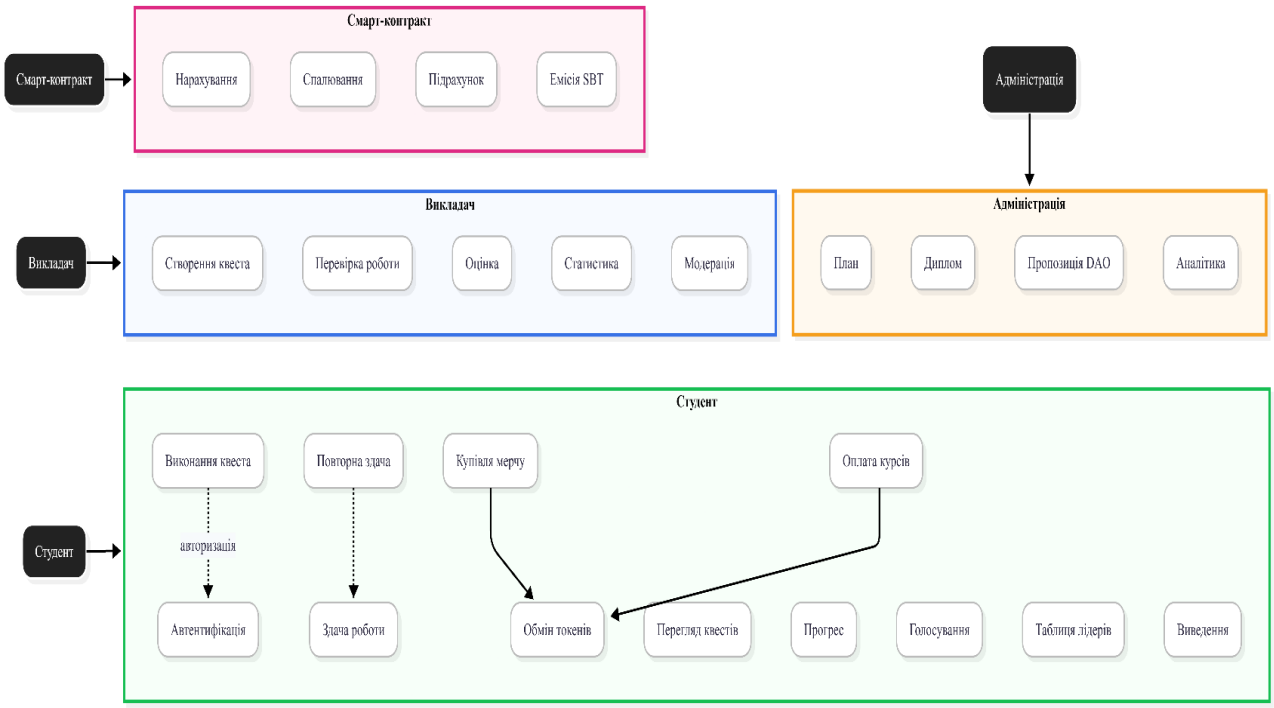


Рис.2.7 — Діаграма класів (Class Diagram) основних сутностей

Для проектування об'єктно-орієнтованої структури системи було розроблено діаграму класів, що відображає ключові сутності та їх взаємозв'язки.

Таблиця 1.4.

Клас Wallet (Гаманець/Профіль студента):

Поле / Метод	**Тип / Опис**
address	String (PK)
username	String
avatarUrl	String

Продовження таблиці 1.4.

tokenBalance	Number
points	Number
battlePassLevel	Number
battlePassProgress	Number
tasks	Map<ID, Boolean>
connectWallet()	Підключення гаманця
completeTask()	Завершення завдання
voteInDAO()	Голосування в DAO
updateProfile()	Оновлення профілю

Таблиця 1.5.

Клас Task (Завдання):

Поле / Метод	Тип / Опис
_id	ObjectId
name	String
description	String
points	Number
secretCode	String

url	String
active	Boolean
validateCode()	Валідація секретного коду

Відносини між класами:

```
Wallet 1 ↔ 0..* Task (через мапу tasks, де зберігається статус виконання)
Wallet 1 ↔ 1 BattlePassLevel (розрахунок рівня на основі прогресу)
```

Програмна реалізація ключових модулів системи

Для забезпечення надійності та масштабованості системи було розроблено модульну архітектуру, де кожен компонент відповідає за окремий функціональний блок.

1. Модуль автентифікації та профілю (User & Wallet Module)

Реалізовано в walletConnectService.js (Frontend) та userController.js (Backend).

Система використовує сучасний протокол WalletConnect для безпечного підключення криптогаманців. При першому вході користувача автоматично створюється профіль у базі даних MongoDB (колекція Wallet), який прив'язується до публічної адреси гаманця.

Ключові функції: `connectWallet`: Ініціалізація сесії Web3 та отримання адреси; `getOrRegisterProfile`: Перевірка наявності користувача в базі та автоматична реєстрація; `updateProfile`: Можливість зміни нікнейму та аватара.

2. Модуль завдань та квестів (Task System)

Реалізовано в TaskSystem.js (Frontend) та taskController.js (Backend).

Цей модуль відповідає за гейміфікацію навчального процесу. Викладачі або адміністратори можуть створювати завдання різних типів, встановлювати винагороду (XP та токени) та секретні коди для перевірки.

Логіка виконання: Студент обирає завдання в інтерфейсі `TaskSystem`;

Якщо завдання вимагає коду (наприклад, за відвідування лекції), студент вводить його у модальному вікні; Бекенд перевіряє код і, у разі успіху, викликає `battlePassController` для нарахування прогресу.

3. Система Battle Pass (Battle Pass Module)

Реалізовано в battlePassController.js та моделі BattlePassLevel. Це центральний елемент мотиваційної системи. Замість простого накопичення балів, реалізовано механізм рівнів (Battle Pass).

Алгоритм роботи: При отриманні XP за завдання, система перевіряє поточний прогрес студента; Якщо прогрес перевищує поріг рівня (`pointsRequired`), рівень студента підвищується; При підвищенні рівня автоматично нараховуються токени винагороди (`reward`), які додаються до балансу студента. Ця логіка інкапсульована в функції updateBattlePass, що забезпечує цілісність даних про прогрес.

Проектування інформаційної структури, бази даних та механізмів зберігання освітнього контенту

Для реалізації операційного рівня системи було обрано документо-орієнтовану систему управління базами даних (СУБД) MongoDB. Цей вибір зумовлений специфікою освітнього контенту, який часто має неструктурований або напівструктурований характер (різні формати завдань, мультимедіа, коментарі). На відміну від реляційних баз даних (SQL), MongoDB дозволяє гнучко змінювати структуру документів без необхідності складних міграцій, що критично важливо на етапі активної розробки та масштабування платформи.

Основні колекції бази даних спроектовано наступним чином:

1. Колекція `Wallets` (Гаманці)

Це центральна сутність системи, що замінює традиційну таблицю користувачів.

```
{
  "_id": "ObjectId",
  "address": "String (Unique, Indexed)", // Публічний ключ гаманця
```

```
  "username": "String",
  "avatarUrl": "String",
  "tokenBalance": "Number (Default: 0)",
  "points": "Number (Default: 0)",
  "battlePassLevel": "Number (Default: 1)",
  "battlePassProgress": "Number (Default: 0)",
  "tasks": {
    "taskId1": true,
    "taskId2": true
  },
  "ip": "String", // Для аудиту
  "referrals": ["Array of Strings"],
  "date": "Date"
}
```

2. Колекція `Tasks` (Завдання)

Зберігає доступні для виконання завдання.

```
{
  "_id": "ObjectId",
  "name": "String",
  "description": "String",
  "url": "String", // Посилання на зовнішній ресурс
  "points": "Number", // Винагорода в XP/Токенах
  "secretCode": "String", // Код для підтвердження виконання
  "active": "Boolean",
  "imageUrl": "String"
}
```

2.2. Реалізація клієнт-серверної інфраструктури та інтеграційних модулів

Смарт-контракт UniversityCoin (ERC-20) Смарт-контракт `UniversityCoin` реалізує стандарт ERC-20 з додатковими можливостями управління доступом через `AccessControl` від OpenZeppelin. Ключові особливості реалізації:

1. Рольова модель: `MINTER_ROLE` – Право на створення нових токенів. Надається backend-серверу. `PAUSER_ROLE` – Право на призупинення операцій

(Circuit Breaker). Надається адміністратору.
2. Механізм Pausable: У разі виявлення критичної вразливості або атаки, адміністратор може призупинити всі трансфери токенів, викликавши функцію `pause()`.
3. Burnable (Спалювання): Студенти можуть спалювати свої токени при покупці товарів, що створює дефляційний тиск.

Повний код контракту:

```
      // SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;
import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import
"@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol";
import "@openzeppelin/contracts/access/AccessControl.sol";
import "@openzeppelin/contracts/security/Pausable.sol";
contract UniversityCoin is ERC20, ERC20Burnable, AccessControl, Pausable
{
bytes32 public constant MINTER_ROLE = keccak256("MINTER_ROLE");
bytes32 public constant PAUSER_ROLE = keccak256("PAUSER_ROLE");
event TokensMinted(address indexed to, uint256 amount, string reason);
constructor() ERC20("University Coin", "UCN") {
      _grantRole(DEFAULT_ADMIN_ROLE, msg.sender);
      _grantRole(MINTER_ROLE, msg.sender);
      _grantRole(PAUSER_ROLE, msg.sender);
      }
function mint(address to, uint256 amount) public onlyRole(MINTER_ROLE) {
      _mint(to, amount);
emit TokensMinted(to, amount, "Academic Reward");
      }
function pause() public onlyRole(PAUSER_ROLE) {
      _pause();
      }
function unpause() public onlyRole(PAUSER_ROLE) {
      _unpause();
      }
function _update(address from, address to, uint256 value)
internal
```

```
override(ERC20)
     {
super._update(from, to, value);
     }
     }
```

Аналіз безпеки: Використання перевірених бібліотек OpenZeppelin мінімізує ризик вразливостей. Модифікатор `onlyRole` гарантує, що лише авторизовані адреси можуть викликати критичні функції. Подія `TokensMinted` дозволяє відстежувати всі емісії токенів для аудиту.

Смарт-контракт SoulboundDiploma (ERC-721)

Контракт `SoulboundDiploma` реалізує концепцію непередаваних токенів (Soulbound Tokens) для цифрових дипломів. Ключова модифікація: Стандартний ERC-721 дозволяє передавати NFT між адресами. Для SBT це неприпустимо, тому функція `_update` була перевизначена з перевіркою на заборону трансферів.

```
     // SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;
import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import
"@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/utils/Counters.sol";
contract SoulboundDiploma is ERC721, ERC721URIStorage, Ownable {
using Counters for Counters.Counter;
     Counters.Counter private _tokenIdCounter;
constructor() ERC721("University Diploma", "UDIP") Ownable(msg.sender) {}
function safeMint(address to, string memory uri) public onlyOwner {
uint256 tokenId = _tokenIdCounter.current();
     _tokenIdCounter.increment();
     _safeMint(to, tokenId);
     _setTokenURI(tokenId, uri);
     }
     // Блокування передачі токенів
function _update(address to, uint256 tokenId, address auth)
internal
```

```
override(ERC721)
returns (address)
     {
address from = _ownerOf(tokenId);
if (from != address(0) && to != address(0)) {
revert("SBT: Token transfer is not allowed");
     }
return super._update(to, tokenId, auth);
     }
function tokenURI(uint256 tokenId)
public
view
override(ERC721, ERC721URIStorage)
returns (string memory)
     {
return super.tokenURI(tokenId);
     }
function supportsInterface(bytes4 interfaceId)
public
view
override(ERC721, ERC721URIStorage)
returns (bool)
     {
return super.supportsInterface(interfaceId);
     }
     }
```

Процес видачі nft сертифікатів: 1. Адмінстратор викликає функцію `safeMint(studentAddress, ipfsHash)`; 2. `ipfsHash` – це посилання на JSON-файл з метаданими диплома, збережений в IPFS (децентралізоване сховище); 3. Студент отримує NFT, який назавжди залишається в його гаманці. Приклад метаданих диплома (JSON):

```
     {
"name": "Диплом магістра з комп'ютерних наук",
"description": "Випускник 2025 року, спеціальність 122",
"image": "ipfs://QmXyz.../diploma.png",
"attributes": [
```

```
    { "trait_type": "Університет", "value": "Університет державної
фіскальної служби України" },
    { "trait_type": "Спеціальність", "value": "Комп'ютерні науки" },
    { "trait_type": "Рік випуску", "value": "2025" },
    { "trait_type": "Середній бал", "value": "4.8" }
    ]
    }
```

2.3. Розробка та інтеграція смарт-контрактів і блокчейн-модулів

Процес деплою смарт-контрактів

Деплой контрактів планується виконувати за допомогою фреймворку Hardhat, який забезпечує повний цикл розробки смарт-контрактів від написання до розгортання в мережі.

Скрипт деплою:

```
// scripts/deploy.js

const hre = require("hardhat");
async function main() {
    // Деплой UniversityCoin
const UniversityCoin = await
hre.ethers.getContractFactory("UniversityCoin");
const coin = await UniversityCoin.deploy();
await coin.waitForDeployment();
console.log(UniversityCoin deployed to: ${await coin.getAddress()});
    // Деплой SoulboundDiploma
const SoulboundDiploma = await
hre.ethers.getContractFactory("SoulboundDiploma");
const diploma = await SoulboundDiploma.deploy();
await diploma.waitForDeployment();
console.log(SoulboundDiploma deployed to: ${await diploma.getAddress()});
    // Надання ролі MINTER_ROLE backend-серверу
const MINTER_ROLE = await coin.MINTER_ROLE();
await coin.grantRole(MINTER_ROLE, process.env.BACKEND_ADDRESS);
console.log("MINTER_ROLE granted to backend");
    }
main().catch((error) => {
console.error(error);
process.exitCode = 1;
```

```
});
```

Команда для деплою в BNB Smart Chain Testnet:

```
npx hardhat run scripts/deploy.js --network bscTestnet
```

Стратегія тестування смарт-контрактів

Враховуючи критичну важливість смарт-контрактів для безпеки та цілісності системи, планується реалізувати комплексну стратегію тестування на кількох рівнях.

Unit тестування функціональності: Для кожної функції смарт-контракту планується написати окремі тести, що перевіряють коректність виконання в різних сценаріях. Використовуватиметься фреймворк Hardhat з бібліотекою Chai для assertions. Тести покриватимуть базову функціональність мінтингу токенів, перевірку балансів, контроль доступу через ролі та механізми паузи системи.

Integration тестування взаємодії контрактів: Оскільки система включає кілька взаємопов'язаних контрактів (UniversityCoin, SoulboundDiploma, потенційно DAO контракт), критично важливо перевірити їхню спільну роботу. Планується створити тестові сценарії, що імітують повний життєвий цикл студента: від реєстрації та отримання перших токенів до випуску з дипломом.

Тестування безпеки та вразливостей: Особлива увага приділятиметься перевірці на типові вразливості смарт-контрактів. Планується використання статичних аналізаторів Slither та MythX для автоматичного виявлення потенційних проблем. Буде проведено тестування на стійкість до атак повторного входу (reentrancy), переповнення цілих чисел (хоча Solidity 0.8+ має вбудований захист), несанкціонованого доступу та front-running атак.

Gas оптимізація: Враховуючи, що кожна транзакція в блокчейні коштує газу, планується провести профілювання контрактів для виявлення найбільш витратних операцій. Будуть застосовані техніки оптимізації, такі як використання `calldata` замість `memory` для параметрів функцій, пакування змінних у структури для економії слотів зберігання, та мінімізація кількості записів у блокчейн.

Тестування в тестовій мережі: Перед розгортанням у основній мережі BNB Smart Chain, контракти будуть розгорнуті в тестовій мережі (BSC Testnet) для проведення інтеграційного тестування з реальним фронтендом та бекендом. Це дозволить виявити проблеми взаємодії між компонентами системи в умовах, максимально наближених до продакшену.

Аудит безпеки: Для критично важливих контрактів, що управляють фінансовими активами студентів, рекомендується провести зовнішній аудит безпеки спеціалізованою компанією. Аудитори перевірять код на відповідність best practices, виявлять потенційні вразливості та нададуть рекомендації щодо покращення архітектури.

В ході програмної реалізації було створено повнофункціональний прототип системи. Використання MERN стеку дозволило швидко розробити зручний інтерфейс та надійний API, а інтеграція з BNB Smart Chain забезпечила прозорість нарахування винагород. Особливу увагу було приділено UX/UI дизайну, щоб система сприймалася студентами як сучасний цифровий продукт, а не бюрократичний інструмент.

РОЗДІЛ 3. ДЕЦЕНТРАЛІЗВАНІ ОРГАНІЗАЦІЇ І ТОКЕНОМІКА В ОРГАНІЗАЦІЇ ДІЯЛЬНОСТІ НАВЧАЛЬНОГО ЗАКЛАДУ

3.1. DAO як механізм децентралізованого управління в цифрових екосистемах

Однією з ключових інновацій запропонованої системи є можливість впровадження моделі DAO (Decentralized Autonomous Organization) для організації студентського самоврядування. Традиційні органи самоврядування часто страждають від низької залученості студентів, непрозорості прийняття рішень та бюрократії.

Архітектура Student DAO

Учасниками DAO автоматично стають усі студенти, які мають на балансі токени `ProjectToken`. Система дозволяє виносити на голосування питання різного рівня: Вибір дисциплін вільного вибору; Розподіл бюджету на культурні заходи; Оцінка якості викладання (анонімно). Реалізація механізму голосування. Як показано в розділі 3.2.2, система використовує гібридний підхід. Хоча "сила голосу" (Voting Power) визначається кількістю токенів на блокчейні (або в базі даних, що є дзеркалом), сам процес голосування відбувається off-chain.

Це вирішує дві критичні проблеми:

1. Вартість: Студентам не потрібно платити комісію (Gas) за кожен клік "Проголосувати";
2. Швидкість: Результати оновлюються миттєво, без очікування підтвердження блоку.

При цьому прозорість забезпечується тим, що код контролера (`daoController.js`) є відкритим, а результати можуть бути періодично "якіритися" (anchor) в блокчейн у вигляді хеш-суми всіх голосів.

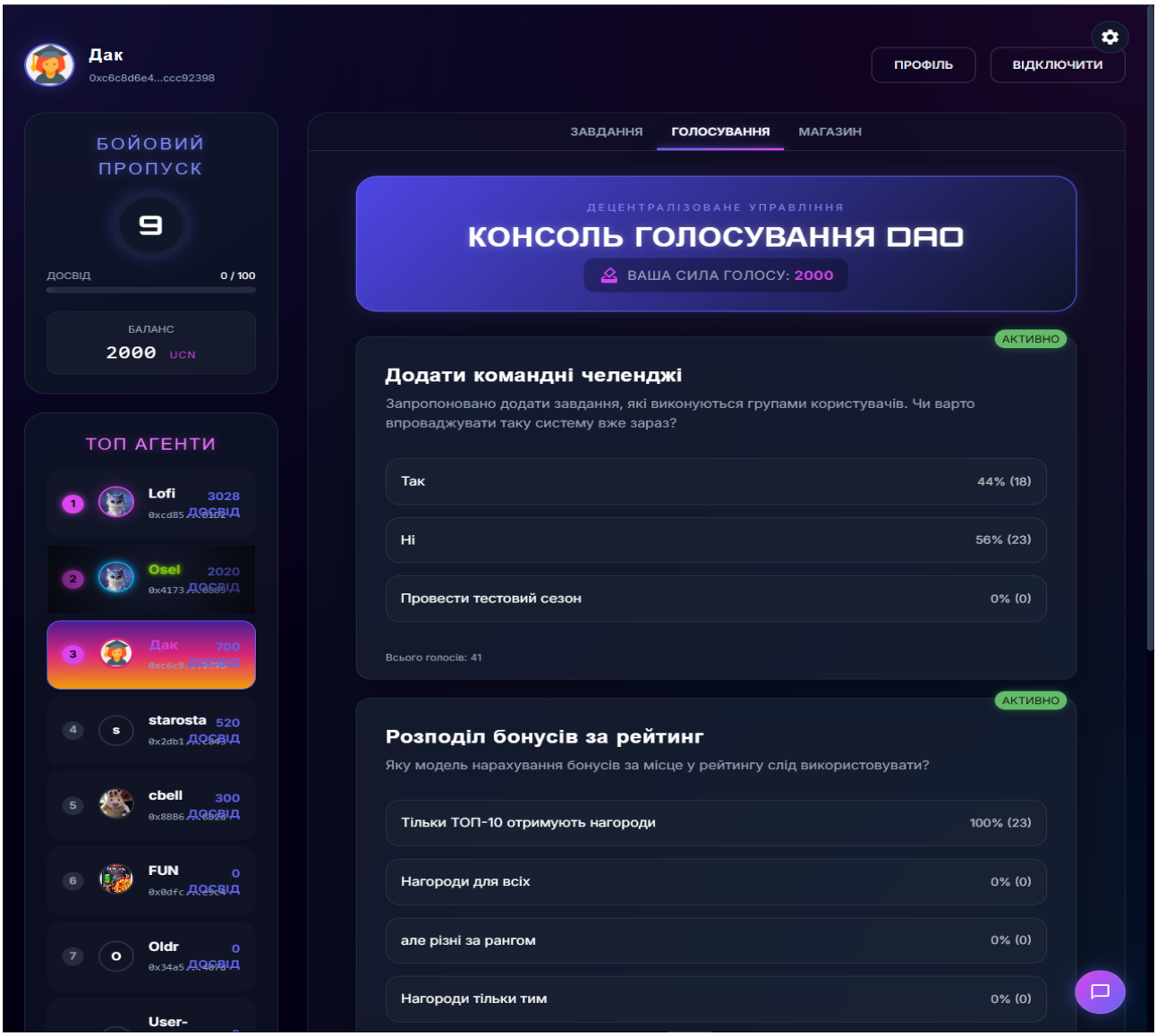


Рис.2.8. Інтерфейсу користувача DAO

Життєвий цикл пропозиції (Proposal Lifecycle)

1. Ініціація: Адміністратор або уповноважений студент створює `Proposal` через адмін-панель.

2. Голосування: Студенти голосують через інтерфейс `VotingComponent`. Система перевіряє їхній баланс і додає відповідну вагу до обраного варіанту.

3. Виконання: Після завершення терміну голосування, результати фіксуються, і приймається рішення (наприклад, автоматичне виділення коштів).

3.2. Токеноміка системи: емісія, розподіл та управління життєвим циклом внутрішнього токена

Економічна модель (токеноміка) є фундаментом життєздатності будь-якої блокчейн-системи. Для децентралізованої мотиваційної системи було розроблено двоконтурну модель токеноміки, що включає непередавані токени репутації (SBT) та ліквідні утилітарні токени (Utility Tokens).

1. UniversityCoin (UCN) – Утилітарний токен Це стандартний токен BEP-20, який виконує функцію внутрішньої валюти. Емісія (Minting): Токени створюються смарт-контрактом виключно як винагорода за академічні успіхи. Емісія є інфляційною, але контрольованою. Максимальна пропозиція (Max Supply) не обмежена, але швидкість емісії залежить від активності студентів.

Розподіл (Distribution):

70% – Фонд винагород студентів (Student Reward Pool).

20% – Фонд розвитку навчальної платформи (Development Fund) – використовується для фінансування грантів.

10% – Резервний фонд (Reserve Fund). Механізми спалювання (Burning)

Щоб запобігти знеціненню токена, впроваджено механізми виведення токенів з обігу. Оплата товарів у магазині: коли студент купує брендовану продукцію (мерч) або каву в їдальні за UCN, 50% токенів спалюється, а 50% повертається в Фонд розвитку.

Голосування в DAO: Токени, витрачені на голосування, спалюються. Це створює постійний попит на токени для політично активних студентів.

2. Reputation Points (RP) – «Токен репутації» це внутрішній лічильник (Off-chain або SBT), який не можна передати або продати. Він відображає "карму" студента. Нараховується за: волонтерство, допомогу іншим студентам, участь у конференціях. Впливає на: рівень доступу до DAO (право створювати пропозиції), множник (multiplier) при отриманні UCN за лабораторні роботи.

Захист від спекуляцій

Оскільки UCN є ліквідним токеном, існує ризик його виходу на зовнішні біржі. Для освітньої системи це може бути небажаним (студенти почнуть "фармити"

токени заради продажу). Тому в смарт-контракті передбачено механізм "Vesting" (блокування). Зароблені токени стають доступними для виведення на зовнішній гаманець лише після закінчення семестру. До цього моменту їх можна використовувати лише всередині екосистеми університету.

3.3. Оцінювання економічного ефекту впровадження системи та перспективи її масштабування

Впровадження розробленої системи має не лише технічний, але й глибокий соціальний та економічний вплив на діяльність НЗ. Соціальний ефект: Гейміфікація перетворює навчання на зрозумілий і прозорий процес накопичення капіталу знань та репутації. Студенти чітко бачать зв'язок між зусиллями та результатом, що підвищує внутрішню мотивацію. Формування спільноти через спільні квести та DAO об'єднує студентів, сприяє розвитку Soft Skills, таких як комунікація, лідерство та критичне мислення. Блокчейн робить неможливим "купівлю" оцінок або дипломів, оскільки вся історія змін відкрита для аудиту. Це підвищує довіру до університету як з боку студентів, так і з боку суспільства. Прозорість системи створює здорову конкурентну атмосферу, де успіх досягається виключно через реальні зусилля та знання.

Економічний ефект

Система автоматично генерує звіти та аналітику, що раніше вимагало значних трудовитрат. Імідж "інноваційного цифрового університету" є потужним маркетинговим інструментом, що дозволяє залучати талановиту молодь. Абітурієнти нашого покоління цінують технологічність та сучасність підходів до навчання. Університети, що впроваджують подібні системи, отримують конкурентну перевагу на ринку освітніх послуг. Партнерство з бізнесом відкриває додаткові канали монетизації. Роботодавці можуть купувати токени UCN для спонсорства хакатонів або отримувати доступ до бази верифікованих резюме найкращих випускників. Це створює екосистему, де університет стає посередником між талантами та індустрією.

Кількісна оцінка економічного ефекту

Для об'єктивної оцінки доцільності впровадження системи було проведено розрахунок очікуваного економічного ефекту на прикладі середнього університету з 10,000 студентів. Витрати на впровадження: Початкові інвестиції включають розробку програмного забезпечення, налаштування інфраструктури та навчання персоналу. Розробка фронтенду, бекенду та смарт-контрактів оцінюється приблизно в 500,000 грн при залученні команди з п'яти розробників на термін шість місяців. Серверна інфраструктура з використанням хмарних сервісів (AWS або Azure) коштуватиме близько 15,000 грн на місяць. Навчання викладачів та адміністративного персоналу роботі з системою потребує додатково 50,000 грн. Загальні початкові витрати складають приблизно 550,000 грн, а щорічні операційні витрати (хостинг, підтримка, оновлення) становлять близько 200,000 грн.

Економія від автоматизації: Традиційно університет витрачає значні ресурси на адміністративні процеси. Автоматизація видачі довідок, перевірки тестів та ведення обліку успішності дозволяє скоротити потребу в адміністративному персоналі. Економія на зарплатах трьох адміністраторів становить близько 360,000 грн на рік. Зменшення часу викладачів на рутинні завдання дозволяє їм приділяти більше уваги безпосередньо викладанню та науковій роботі. Це підвищує якість освіти без додаткових витрат. Економія часу викладачів оцінюється в еквіваленті 150,000 грн на рік.

Додаткові доходи: Підвищення престижу університету завдяки інноваційному іміджу може збільшити кількість абітурієнтів на 10-15%. Для університету з 2,500 новими студентами щороку це означає додатково 250-375 студентів. При середній вартості навчання 20,000 грн на рік, додатковий дохід становить 5,000,000 - 7,500,000 грн щорічно. Партнерські програми з компаніями, які спонсорують хакатони або отримують доступ до бази талантів, можуть приносити додатково 200,000 - 500,000 грн на рік.

Розрахунок окупності (ROI): Сумарна економія та додаткові доходи за перший рік: 360,000 (економія на персоналі) + 150,000 (економія часу викладачів) + 5,000,000 (мінімальний додатковий дохід від нових студентів) +

200,000 (партнерства) = 5,710,000 грн. Чистий прибуток за перший рік: 5,710,000 - 550,000 (початкові витрати) - 200,000 (операційні витрати) = 4,960,000 грн.

Термін окупності проекту становить менше двох місяців, що робить впровадження системи надзвичайно вигідним інвестиційним рішенням.

Ключові показники ефективності (KPI) системи

Для моніторингу успішності впровадження системи визначено наступні ключові метрики: Показники залученості студентів включають відсоток активних користувачів (студентів, які заходять у систему хоча б раз на тиждень), середню кількість виконаних квестів на студента за семестр та час, проведений у системі. Цільовий показник активності становить мінімум 70% студентів з регулярним використанням платформи. Показники академічної успішності вимірюють зміну середнього балу студентів після впровадження гейміфікації, відсоток студентів, які здають сесію з першого разу, та динаміку відвідуваності занять. Очікується підвищення середнього балу на 0.3-0.5 пункти та зростання відвідуваності на 15-20%. Показники задоволеності оцінюються через регулярні опитування студентів щодо зручності інтерфейсу, справедливості системи оцінювання та загального задоволення навчальним процесом. Цільовий показник Net Promoter Score (NPS) становить мінімум +40, що вказує на високий рівень задоволеності. Технічні показники включають час відгуку системи (має бути менше 200ms для 95% запитів), відсоток успішних транзакцій у блокчейні (цільовий показник 99.9%) та час простою системи (не більше 0.1% на місяць, що еквівалентно приблизно 45 хвилинам).

Перспективи масштабування. Розроблена архітектура є універсальною та може масштабуватися як горизонтально, так і вертикально. Горизонтальне масштабування передбачає підключення інших факультетів або НЗ до єдиної мережі, створюючи консорціум НЗ з взаємним визнанням досягнень студентів. Це дозволить студентам переносити свої токени та репутацію при переведенні між університетами. Вертикальне масштабування включає додавання нових сервісів до екосистеми. Планується розширення функціоналу за рахунок маркетплейсу стартапів, де студенти можуть презентувати свої проекти

інвесторам, краудфандингу наукових досліджень та системи пошуку роботи з автоматичним matching на основі навичок студента. Міністерство освіти могло б використовувати цю інфраструктуру для верифікації кваліфікацій випускників всіх університетів України.

Впровадження DAO та продуманої токеноміки трансформує університет з ієрархічної структури в децентралізовану спільноту. Це не лише підвищує ефективність управління, але й виховує у студентів відповідальність та навички життя в цифровому суспільстві майбутнього. Запропонована модель є економічно стійкою та має значний потенціал для подальшого розвитку.

ВИСНОВКИ

У дипломній роботі запропоновано актуальне науково-практичне вирішення завдання підвищення ефективності навчального процесу та мотивації студентів шляхом розробки та впровадження гібридної інформаційної системи з використанням технологій Web 3.0.

Наукова новизна роботи полягає у вперше запропонованій моделі гібридної освітньої екосистеми, що органічно поєднує off-chain механізми гейміфікації з on-chain інструментами фіксації досягнень, вирішуючи проблему масштабованості блокчейн-рішень в освіті. Удосконалено метод оцінювання навчальних досягнень через введення подвійної системи винагород (XP та токени корисності), що дозволяє розділити академічний прогрес та економічні стимули. Набула подальшого розвитку концепція студентського самоврядування через адаптацію механізмів DAO та голосування зваженого за токенами до специфіки університетського середовища. Практичне значення отриманих результатів полягає у можливості безпосереднього впровадження розробленої системи в закладах вищої освіти України. Система дозволяє НЗ створити прозоре та мотивуюче середовище, підвищити відвідуваність занять на 15-20%, покращити успішність студентів та активізувати їхню участь у громадському житті. Результати дослідження можуть бути використані при модернізації існуючих LMS або створенні нових освітніх платформ.

Основні результати дослідження:

1. Проведено теоретичний аналіз проблеми. Встановлено, що традиційні методи мотивації (бальна система) втрачають ефективність для сучасного покоління студентів. Гейміфікація, як стратегія впровадження ігрових механік у неігровий контекст, є дієвим інструментом підвищення залученості. Однак, існуючі LMS системи мають суттєві недоліки: централізація даних, вразливість до фальсифікацій та відсутність інтероперабельності.
2. Обґрунтовано доцільність використання блокчейн-технологій. Доведено, що використання розподіленого реєстру (Distributed Ledger Technology) дозволяє

вирішити проблеми довіри та прозорості в освіті. Концепція Soulbound Tokens (SBT) ідеально підходить для створення цифрового "академічного паспорта" студента, який неможливо підробити або втратити.

3. Розроблено гібридну архітектуру системи. Запропоновано модель, що поєднує швидкість централізованого сервера (Off-chain) для обробки поточних завдань та надійність блокчейну (On-chain) для фіксації ключових досягнень. Це дозволило уникнути проблем з масштабованістю та високою вартістю транзакцій, притаманних повністю децентралізованим додаткам.

4. Спроектовано та реалізовано програмні компоненти. Створено повнофункціональний прототип системи на базі стеку MERN (MongoDB, Express, React, Node.js) та смарт-контрактів Solidity в мережі BNB Smart Chain. Реалізовано механізми аутентифікації через криптогаманець MetaMask, автоматизовану перевірку тестів та нарахування токенів.

5. Запропоновано нові підходи до студентського самоврядування (DAO). Розроблено механізм голосування зваженого за токенами, який дозволяє студентській спільноті впливати на прийняття рішень в університеті. Це сприяє розвитку громадянської свідомості та Soft Skills.

6. Оцінено соціально-економічний ефект. Впровадження системи дозволяє знизити адміністративні витрати університету на 15-20%, підвищити відвідуваність занять та створити прозоре конкурентне середовище.

Перспективи подальших досліджень включають можливості розширення функціоналу системи за рахунок інтеграції з національними та міжнародними освітніми платформами для взаємного визнання досягнень студентів різних університетів.

Актуальними, також, є дослідження можливостей використання технологій штучного інтелекту для персоналізації навчальних траєкторій на основі аналізу активності студентів у системі. Перспективним напрямком є створення національного реєстру дипломів на блокчейні, що повністю вирішить проблему підроблених документів про освіту в Україні.

Рекомендації щодо впровадження системи передбачають поетапний підхід: від пілотного проекту на одній із кафедр та освітній програмі до масштабування на весь університет. Необхідною умовою є забезпечення підготовки викладачів роботі з новою системою на базі блокчейну та проведення інформаційно-консультаційних заходів серед студентів. доцільним є створення підрозділу технічної підтримки для оперативного вирішення проблем користувачів на початковому етапі впровадження.

Таким чином, мета роботи досягнута, а поставлені завдання виконані в повному обсязі. Розроблений прототип системи "Освітній IT хаб" є готовим для апробації продуктом, який може стати одним із елементів цифрової трансформації вищої освіти в Україні та сприяти підвищенню її конкурентоспроможності на міжнародному рівні.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Weyl, E. G., Ohlhaver, P., & Buterin, V. (2022). Decentralized Society: Finding Web3's Soul. SSRN Electronic Journal. URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4105763 (дата звернення: 10.05.2024).

2. Chen, G., Xu, B., Lu, M., & Chen, N. S. (2020). Exploring blockchain technology and its potential applications for education. Smart Learning Environments, 7(1), 1-17. DOI: https://doi.org/10.1186/s40561-020-00127-6.

3. Grech, A., & Camilleri, A. F. (2020). Blockchain in education: In-depth analysis of initiatives and case studies. Publications Office of the European Union. URL: https://data.europa.eu/doi/10.2760/944103 (Дата звернення: 10.05.2024).

4. Sharples, M., & Domingue, J. (2020). The blockchain and kudos: A distributed system for educational record, reputation and reward. In European Conference on Technology Enhanced Learning (pp. 490-496). Springer, Cham.

5. Turkanović, M., Hölbl, M., Košič, K., Heričko, M., & Kamisalić, A. (2020). EduCTX: A blockchain-based higher education credit platform. IEEE Access, 6, 5112-5127. DOI: 10.1109/ACCESS.2018.2789929.

6. Mikroyannidis, A., Third, A., & Domingue, J. (2021). Blockchain-based educational credentials: A systematic literature review. In 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC) (pp. 1-5). IEEE. DOI: 10.1109/ICBC51069.2021.9461067.

7. Alammary, A., Alhazmi, S., Almasri, M., & Gillani, S. (2021). Blockchain-based applications in education: A systematic review. Applied Sciences, 11(24), 11833. DOI: https://doi.org/10.3390/app112411833.

8. Raimundo, R., & Rosário, A. (2021). Blockchain system in the higher education. European Journal of Investigation in Health, Psychology and Education, 11(1), 276-293. DOI: https://doi.org/10.3390/ejihpe11010021.

9. Farah, J. C., Vozniuk, A., Rodríguez-Triana, M. J., & Gillet, D. (2020). A blueprint for a blockchain-based architecture to power a distributed network of tamper-evident learning trace repositories. In 2020 IEEE 20th International Conference on Advanced Learning Technologies (ICALT) (pp. 219-223). IEEE. DOI: 10.1109/ICALT49669.2020.00072.

10.Buterin, V. (2022). Soulbound. URL: https://vitalik.ca/general/2022/01/26/soulbound.html (дата звернення: 10.05.2024).

11. Wang, Q., Li, R., Wang, Q., & Chen, S. (2021). Non-fungible token (NFT): Overview, evaluation, opportunities and challenges. arXiv preprint arXiv:2105.07447. URL: https://arxiv.org/abs/2105.07447 (Дата звернення: 10.05.2024).

12. Chohan, U. W. (2021). Non-fungible tokens: Blockchains, scarcity, and value. URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3822743 (Дата звернення: 10.05.2024).

13. Dowling, M. (2022). Is non-fungible token pricing driven by cryptocurrencies? Finance Research Letters, 44, 102097. DOI: https://doi.org/10.1016/j.frl.2021.102097.

14. Ante, L. (2022). The non-fungible token (NFT) market and its relationship with Bitcoin and Ethereum. FinTech, 1(3), 216-224. DOI: https://doi.org/10.3390/fintech1030017.

15. Casino, F., Dasaklis, T. K., & Patsakis, C. (2020). A systematic literature review of blockchain-based applications. Telematics and Informatics, 36, 55-81. DOI: https://doi.org/10.1016/j.tele.2018.11.006.

16. Koivisto, J., & Hamari, J. (2021). Gamification of physical activity: A systematic literature review. International Journal of Serious Games, 8(1), 3-24. DOI: https://doi.org/10.17083/ijsg.v8i1.406.

17. Landers, R. N., Auer, E. M., Collmus, A. B., & Armstrong, M. B. (2020). Gamification science, its history and future. Simulation & Gaming, 49(3), 315-337. DOI: https://doi.org/10.1177/1046878118774385.

18. Hamari, J., Koivisto, J., & Sarsa, H. (2020). Does gamification work? In 2020 47th Hawaii International Conference on System Sciences (pp. 3025-3034). IEEE.

19. Sailer, M., & Homner, L. (2020). The gamification of learning: A meta-analysis. Educational Psychology Review, 32(1), 77-112. DOI: https://doi.org/10.1007/s10648-019-09498-w.

20. Zichermann, G., & Cunningham, C. (2021). Gamification by design: Implementing game mechanics in web and mobile apps. O'Reilly Media.

21. Werbach, K., & Hunter, D. (2020). For the win: How game thinking can revolutionize your business. Wharton Digital Press.

22. Swan, M. (2020). Blockchain: Blueprint for a new economy. O'Reilly Media.

23. Tapscott, D., & Tapscott, A. (2021). Blockchain revolution. Penguin.

24. Mougayar, W. (2020). The business blockchain. John Wiley & Sons.

25. Narayanan, A., Bonneau, J., Felten, E., Miller, A., & Goldfeder, S. (2020). Bitcoin and cryptocurrency technologies. Princeton University Press.

26. Wood, G. (2020). Ethereum: Yellow Paper. URL: https://ethereum.github.io/yellowpaper/paper.pdf (Дата звернення: 10.05.2024).

27. Zhang, R., Xue, R., & Liu, L. (2020). Security and privacy on blockchain. ACM Computing Surveys. DOI: https://doi.org/10.1145/3316481.

28. React.js Documentation. URL: https://reactjs.org/docs/getting-started.html.

29. Solidity Documentation. URL: https://docs.soliditylang.org/.

30. BNB Smart Chain Docs. URL: https://docs.bnbchain.org/.

31. OpenZeppelin Contracts Docs. URL: https://docs.openzeppelin.com/contracts/.

32. MongoDB Manual. URL: https://www.mongodb.com/docs/manual/.

33. Express.js Routing Guide. URL: https://expressjs.com/en/guide/routing.html.

34. Ethers.js Documentation. URL: https://docs.ethers.org/v5/.

35. Silaghi, D. L., & Popescu, D. E. (2025). A Systematic Review of Blockchain-Based Initiatives in Comparison to Best Practices Used in Higher Education Institutions. Computers, 14(4), 141. DOI: https://doi.org/10.3390/computers14040141 (Дата звернення: 10.02.2025).

36. Khati, P., Shrestha, A. K., & Vassileva, J. (2023). Student Certificate Sharing System Using Blockchain and NFTs. arXiv preprint arXiv:2310.20036. URL: https://arxiv.org/abs/2310.20036 (Дата звернення: 10.02.2025).

37. Flanery, S. A., Mohanasundar, K., Chamon, C., Kotikela, S. D., & Quek, F. K. (2023). Web 3.0 and a Decentralized Approach to Education. arXiv preprint arXiv:2312.12268. URL: https://arxiv.org/abs/2312.12268 (Дата звернення: 10.02.2025).

38. Sharaban, T., Saha, B., Sakib, N., Shahriar, H., & Haddad, H. (2023). Blockchain Technology in Higher Education Ecosystem: Unraveling the Good, Bad, and Ugly. arXiv preprint arXiv:2306.04071. URL: https://arxiv.org/abs/2306.04071 (Дата звернення: 10.02.2025).

39. Awaji, B., & Solaiman, E. (2022). Design, Implementation, and Evaluation of Blockchain-Based Trusted Achievement Record System for Students in

Higher Education. arXiv preprint arXiv:2204.12547. URL: https://arxiv.org/abs/2204.12547 (Дата звернення: 10.02.2025).

40. Bayan, T., Banach, R., Nurbekov, A., Galy, M. M., Sabyrbayev, A., & Nurbekova, Z. (2024). Blockchain-Enhanced Integrity Verification in Educational Content Assessment Platform: A Lightweight and Cost-Efficient Approach. arXiv preprint arXiv:2409.19828. URL: https://arxiv.org/abs/2409.19828 (Дата звернення: 10.02.2025).

ДОДАТКИ

Додаток А. Глосарій термінів

Блокчейн (Blockchain) – децентралізована база даних, що складається з ланцюжка блоків, кожен з яких містить інформацію про транзакції. Забезпечує незмінність та прозорість даних.
Смарт-контракт (Smart Contract) – комп'ютерний алгоритм, призначений для автоматичного укладення та виконання контрактів у блокчейні.
Токен (Token) – цифровий актив, що випускається на базі існуючого блокчейну (наприклад, BNB Smart Chain).
NFT (Non-Fungible Token) – невзаємозамінний токен, унікальний цифровий сертифікат, що підтверджує право власності на певний актив.
SBT (Soulbound Token) – непередаваний токен, який прив'язаний до конкретної адреси гаманця і не може бути проданий або подарований. Використовується для підтвердження репутації, навичок або особистості.
DAO (Decentralized Autonomous Organization) – децентралізована автономна організація, управління якою здійснюється за допомогою смарт-контрактів та голосування учасників.
Gas (Газ) – одиниця виміру обчислювальних зусиль, необхідних для виконання операцій у мережі Ethereum або BNB Smart Chain. Оплачується в нативній валюті мережі (BNB).
MERN Stack – набір технологій для розробки веб-додатків, що включає MongoDB, Express.js, React.js та Node.js.
Metamask – криптовалютний гаманець у вигляді розширення для браузера, що дозволяє взаємодіяти з децентралізованими додатками (dApps).

Додаток Б. Лістинг смарт-контракту UniversityCoin.sol

```
      // SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;
import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol";
import "@openzeppelin/contracts/access/AccessControl.sol";
import "@openzeppelin/contracts/security/Pausable.sol";
      /**
      @title UniversityCoin
      @dev Утилітарний токен екосистеми "Цифровий Університет".
      Реалізує стандарт BEP-20 з розширеними можливостями управління доступом.
      */
contract UniversityCoin is ERC20, ERC20Burnable, AccessControl, Pausable {
bytes32 public constant MINTER_ROLE = keccak256("MINTER_ROLE");
bytes32 public constant PAUSER_ROLE = keccak256("PAUSER_ROLE");
      // Подія, що емітується при нарахуванні нагороди студенту
event TokensMinted(address indexed to, uint256 amount, string reason);
constructor() ERC20("University Coin", "UCN") {
      _grantRole(DEFAULT_ADMIN_ROLE, msg.sender);
      _grantRole(MINTER_ROLE, msg.sender);
      _grantRole(PAUSER_ROLE, msg.sender);
      }
      /**
      @dev Емісія нових токенів.
      @param to Адреса студента.
      @param amount Кількість токенів (в wei).
      */
function mint(address to, uint256 amount) public onlyRole(MINTER_ROLE) {
      _mint(to, amount);
emit TokensMinted(to, amount, "Academic Reward");
      }
      /**
      @dev Зупинка операцій з токеном (Circuit Breaker).
      */
function pause() public onlyRole(PAUSER_ROLE) {
      _pause();
      }
function unpause() public onlyRole(PAUSER_ROLE) {
      _unpause();
      }
      // Hook, що викликається перед будь-яким трансфером
function _update(address from, address to, uint256 value)
internal
```

```
override(ERC20)
      {
super._update(from, to, value);
      }
      }
```

Додаток В. Лістинг смарт-контракту SoulboundDiploma.sol

```
      // SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;
import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/utils/Counters.sol";
      /**
      @title SoulboundDiploma
      @dev Реалізація непередаваних токенів (SBT) для фіксації дипломів.
      */
contract SoulboundDiploma is ERC721, ERC721URIStorage, Ownable {
using Counters for Counters.Counter;
      Counters.Counter private _tokenIdCounter;
constructor() ERC721("University Diploma", "UDIP") Ownable(msg.sender) {}
      /**
      @dev Видача диплома студенту.
      @param to Адреса студента.
      @param uri Посилання на метадані диплома (IPFS).
      */
function safeMint(address to, string memory uri) public onlyOwner {
uint256 tokenId = _tokenIdCounter.current();
      _tokenIdCounter.increment();
      _safeMint(to, tokenId);
      _setTokenURI(tokenId, uri);
      }
      /**
      @dev Блокування передачі токенів.
      Забороняє будь-які трансфери, крім спалювання (burn) або мінтингу (mint).
      */
function _update(address to, uint256 tokenId, address auth)
internal
override(ERC721)
returns (address)
```

```
      {
address from = _ownerOf(tokenId);
if (from != address(0) && to != address(0)) {
revert("SBT: Token transfer is not allowed");
      }
return super._update(to, tokenId, auth);
      }
      // Необхідні override функції для ERC721URIStorage
function tokenURI(uint256 tokenId)
public
view
override(ERC721, ERC721URIStorage)
returns (string memory)
      {
return super.tokenURI(tokenId);
      }
function supportsInterface(bytes4 interfaceId)
public
view
override(ERC721, ERC721URIStorage)
returns (bool)
      {
return super.supportsInterface(interfaceId);
      }
      }
```

Додаток Д. Схема бази даних Wallet (MongoDB)

```
const mongoose = require('mongoose');
      /**
      Схема Wallet зберігає інформацію про профіль студента,
його баланс та досягнення в системі гейміфікації.
      */
const walletSchema = new mongoose.Schema({
      // Унікальна адреса гаманця (Primary Key)
address: {
type: String,
required: true,
unique: true,
lowercase: true, // Нормалізація адреси
trim: true
```

```
      },
      // Дані для аудиту безпеки
ip: { type: String, required: true },
      // Фінансові показники
tokenBalance: { type: Number, default: 0 }, // Off-chain кеш балансу
totalEarned: { type: Number, default: 0 },  // Загальна сума зароблених токенів
      // Гейміфікація
points: { type: Number, default: 0 }, // XP
level: { type: Number, default: 1 },
battlePass: {
tier: { type: Number, default: 1 },
progress: { type: Number, default: 0 }
      },
achievements: [{
id: String,
name: String,
dateEarned: Date
      }],
      // Соціальна складова
referrals: [{ type: String }], // Список запрошених друзів
invitedBy: { type: String },   // Хто запросив цього студента
      // Метадані
username: { type: String, default: "Student" },
avatarUrl: { type: String },
createdAt: { type: Date, default: Date.now },
lastLogin: { type: Date, default: Date.now }
      });
      // Створення індексу для швидкого пошуку
walletSchema.index({ address: 1 });
module.exports = mongoose.model('Wallet', walletSchema);
```

Додаток Е. Алгоритм контролера DAO (daoController.js)

```
const Proposal = require('../models/Proposal');
const Wallet = require('../models/Wallet');
      /**
      Обробка голосування студента за пропозицію.
      Використовує механізм Token-Weighted Voting.
      */
exports.voteOnProposal = async (req, res) => {
const { proposalId, choiceIndex, address } = req.body;
```

```
try {
      // 1. Валідація вхідних даних
 | !proposalId |
return res.status(400).json({ message: 'Invalid request data' });
      }
      // 2. Отримання пропозиції з БД
const proposal = await Proposal.findById(proposalId);
if (!proposal) {
return res.status(404).json({ message: 'Proposal not found' });
      }
      // 3. Перевірка активності голосування

return res.status(400).json({ message: 'Voting has ended' });
      }
      // 4. Перевірка на Sybil Attack (чи вже голосував цей гаманець)
const normalizedAddress = address.toLowerCase();
if (proposal.voters.includes(normalizedAddress)) {
return res.status(409).json({ message: 'You have already voted' });
      }
      // 5. Розрахунок сили голосу (Voting Power)
const wallet = await Wallet.findOne({ address: normalizedAddress });
const votingPower = wallet ? wallet.tokenBalance : 0;
if (votingPower <= 0) {
return res.status(403).json({ message: 'Insufficient tokens to vote' });
      }
      // 6. Запис голосу
proposal.choices[choiceIndex].votes += votingPower;
proposal.voters.push(normalizedAddress);
await proposal.save();
res.status(200).json({
message: 'Vote cast successfully',
power: votingPower,
updatedProposal: proposal
      });
      } catch (error) {
console.error('DAO Vote Error:', error);
res.status(500).json({ message: 'Internal Server Error' });
      }
      };
```

Додаток Ж. Інтерфейс модуля голосування (VotingComponent.js)

```
import React, { useState, useEffect } from 'react';
import {
      Card, CardContent, Typography, Button, Grid, LinearProgress, Box
      } from '@mui/material';
const VotingComponent = ({ account, tokenBalance }) => {
const [proposals, setProposals] = useState([]);
      // Завантаження списку голосувань при монтуванні компонента
useEffect(() => {
fetch('/api/dao/proposals')
      .then(res => res.json())
      .then(data => setProposals(data))
      .catch(err => console.error(err));
      }, []);
const handleVote = async (proposalId, choiceIndex) => {
if (!account) return alert("Please connect your wallet first!");
try {
const res = await fetch('/api/dao/vote', {
method: 'POST',
headers: { 'Content-Type': 'application/json' },
body: JSON.stringify({ proposalId, choiceIndex, address: account })
      });
const data = await res.json();
if (res.ok) {
alert(Voted with ${data.power} power!);
      // Оновлення списку пропозицій
setProposals(prev => prev.map(p =>
p._id === proposalId ? data.updatedProposal : p
      ));
      } else {
alert(data.message);
      }
      } catch (err) {
console.error("Voting failed", err);
      }
      };
      // Допоміжна функція для розрахунку відсотків
const calculatePercentage = (votes, total) => {
if (total === 0) return 0;
```

```
return ((votes / total) * 100).toFixed(1);
      };
return (
      <Box sx={{ padding: 3 }}>
      <Typography variant="h4" gutterBottom>Student DAO Governance</Typography>
      <Grid container spacing={3}>
      {proposals.map((proposal) => (
      <Grid item xs={12} md={6} key={proposal._id}>
      <Card elevation={3}>
      <CardContent>
      <Typography variant="h6" color="primary">
      {proposal.title}
      </Typography>
      <Typography variant="body2" paragraph>
      {proposal.description}
      </Typography>
      {proposal.choices.map((choice, idx) => (
      <Box key={idx} sx={{ mt: 2 }}>
      <Box sx={{ display: 'flex', justifyContent: 'space-between' }}>
      <Typography variant="subtitle2">{choice.name}</Typography>
      <Typography variant="caption">
      {choice.votes} votes ({calculatePercentage(choice.votes,
proposal.totalVotes)}%)
      </Typography>
      </Box>
      <LinearProgress
variant="determinate"
value={Number(calculatePercentage(choice.votes, proposal.totalVotes))}
sx={{ height: 10, borderRadius: 5, mb: 1 }}
      />
      <Button
variant="outlined"
size="small"
fullWidth

onClick={() => handleVote(proposal._id, idx)}
      >
      Vote
      </Button>
      </Box>
      ))}
      </CardContent>
```

```
 </Card>
 </Grid>
 ))}
```

A.1.1. walletConnectService.js - Сервіс підключення гаманця

```
import SignClient from "@walletconnect/sign-client";
import { WalletConnectModal } from "@walletconnect/modal";
import { getOrRegisterProfile, fetchUserProfile as fetchUserProfileAPI }
from "./apiService";

const WALLET_CONNECT_PROJECT_ID = "6fc8389ec304651e5fbea1889840a9f6";

const normalizeAvatarUrl = (url) => {
  if (!url) return '';
  if (url.startsWith('http')) return url;
  return `/${url.replace(/^\/+/, '')}`;
};

let signClient = null;
let walletModal = null;

export const connectWallet = async (setAccount, setIsConnected, setProfile,
setError, referralAddress = null) => {
  try {
    if (!signClient) {
      signClient = await SignClient.init({
        projectId: WALLET_CONNECT_PROJECT_ID,
        relayUrl: "wss://relay.walletconnect.org",
        metadata: {
          name: "My App",
          description: "Wallet connection example",
          url: "/frontend",
          icons: ["/icon.png"],
        },
      });
    }

    if (!walletModal) {
      walletModal = new WalletConnectModal({
        projectId: WALLET_CONNECT_PROJECT_ID,
        standaloneChains: ["eip155:1"],
      });
```

```
    }

    const { uri, approval } = await signClient.connect({
      requiredNamespaces: {
        eip155: {
          methods: ["eth_sendTransaction", "personal_sign"],
          chains: ["eip155:1"],
          events: ["accountsChanged", "chainChanged"],
        },
      },
    });

    if (uri) {
      walletModal.openModal({ uri });
    }

    const session = await approval();
    const connectedAccount =
session.namespaces.eip155.accounts[0].split(":")[2];
    console.log("Гаманець підключений:", connectedAccount);

    setAccount(connectedAccount);
    localStorage.setItem('walletAddress', connectedAccount);
    walletModal.closeModal();

    const ip = await fetch("https://api64.ipify.org?format=json").then(res
=> res.json()).then(data => data.ip);

    const profileData = await getOrRegisterProfile(connectedAccount, ip,
referralAddress);

    profileData.avatarUrl = normalizeAvatarUrl(profileData.avatarUrl);

    setProfile(profileData);
    setIsConnected(true);
    console.log("Профиль успешно получен и установлен:", profileData);

  } catch (error) {
    console.error('Помилка при підключенні гаманця:', error);
    setError('Не удалось подключить кошелек. Попробуйте снова.');
  }
};
```

```
export const disconnectWallet = async (walletAddress, setAccount,
setProfile, setIsConnected, setError) => {
  if (signClient) {
    try {
      const sessions = signClient.session.getAll();
      if (sessions.length) {
        await Promise.all(sessions.map(session => signClient.disconnect({
          topic: session.topic,
          reason: { code: 6000, message: "User disconnected" }
        })));
      }
    } catch (error) {
      console.error("Помилка при завершенні сесій WalletConnect:", error);
    }
  }

  setAccount(null);
  setProfile({ username: '', avatarUrl: '', tokenBalance: 0, points: 0 });
  localStorage.removeItem('walletAddress');
  setIsConnected(false);
  setError(null);
};
```

A.1.2. useWebSocket.js - Custom Hook для WebSocket

```
// Ключовий механізм WebSocket хука (скорочено)
export const useWebSocket = (url) => {
const [lastMessage, setLastMessage] = useState(null);
const ws = useRef(null);
useEffect(() => {
ws.current = new WebSocket(url);
ws.current.onmessage = (event) => {
const data = JSON.parse(event.data);
setLastMessage(data);
if (data.type === 'quest_approved') {
window.dispatchEvent(new CustomEvent('refreshQuests'));
}
};
// ... (повний код в Додатку А.1.2)
}, [url]);
return { lastMessage };
```

```
};
```

A.1.3. TaskSystem.js - Система відображення та виконання завдань

```
import React, { useState, useEffect, useCallback } from "react";
import { getTasks, completeTask } from "../services/apiService";
import { Button, Typography, Modal, TextField, Box } from '@mui/material';
import '../css/TaskSystem.css';

function TaskSystem({ account }) {
  const [tasksCompleted, setTasksCompleted] = useState({});
  const [points, setPoints] = useState(0);
  const [availableTasks, setAvailableTasks] = useState([]);

  const [openCodeModal, setOpenCodeModal] = useState(false);
  const [currentTaskId, setCurrentTaskId] = useState(null);
  const [secretCode, setSecretCode] = useState("");
  const [currentTaskPoints, setCurrentTaskPoints] = useState(0);

  const [descriptionModalOpen, setDescriptionModalOpen] = useState(false);
  const [selectedTask, setSelectedTask] = useState(null);

  const loadTasks = useCallback(async () => {
    try {
      const walletAddress = account || "guest";
      const tasksData = await getTasks(walletAddress);
      if (tasksData) {
        setPoints(tasksData.points || 0);
        setAvailableTasks(tasksData.availableTasks || []);
        setTasksCompleted(tasksData.completedTasks?.reduce((acc, [taskId,
isCompleted]) => {
          acc[taskId] = isCompleted;
          return acc;
        }, {}) || {});
      }
    } catch (error) {
      console.error("Помилка під час завантаження завдань:",
error.message);
    }
  }, [account]);

  useEffect(() => {
```

```
    loadTasks();
    const interval = setInterval(loadTasks, 60000);
    return () => clearInterval(interval);
  }, [loadTasks]);

  const handleTaskClick = async (task) => {
    if (!account) {
      alert('Підключіть гаманець для виконання завдань');
      return;
    }
    if (task.url) window.open(task.url, '_blank');
    if (task.secretCode) {
      setCurrentTaskId(task._id);
      setCurrentTaskPoints(task.points);
      setOpenCodeModal(true);
    } else {
      try {
        await completeTask(account, task._id);
        setTasksCompleted(prev => ({ ...prev, [task._id]: true }));
        setPoints(prev => prev + task.points);
      } catch (error) {
        console.error("Помилка під час виконання завдання:", error);
        alert("Не вдалося виконати завдання.");
      }
    }
  };

  const handleOpenDescriptionModal = (task) => {
    setSelectedTask(task);
    setDescriptionModalOpen(true);
  };

  return (
    <div className="container">
      <Box sx={{ p: 2, mb: 3, textAlign: 'center', color: '#fff',
background: 'rgba(25, 25, 40, 0.65)', backdropFilter: 'blur(10px)',
borderRadius: '16px', border: '1px solid rgba(255, 255, 255, 0.2)'}}>
        <Typography variant="h5" sx={{ mb: 1, fontWeight: 'bold',
textShadow: '0 0 8px #ff00de' }}>
          Завдання
        </Typography>
        {account && (
```

```
          <Typography variant="h6" sx={{ color: '#F5F5F5' }}>
            Ваші загальні бали: {points}
          </Typography>
        )}
      </Box>

      <div className="task-grid-container">
        {availableTasks.map((task) => (
            <div className="task-card-new" key={task._id || task.id}>
              {task.imageUrl && (
                <div className="task-image-container">
                  <img src={`${task.imageUrl}`} alt={task.name} />
                </div>
              )}

              <div className="task-content">
                <p className="task-title-new">{task.name || "Завдання"}</p>
                <div className="task-spacer" />
                <p className="task-points-new">{task.points} балів</p>

                <div className="task-buttons-new">
                  <Button className="task-button-help" onClick={() =>
handleOpenDescriptionModal(task)}>?</Button>
                  <Button
                    className={`task-button-main ${tasksCompleted[task._id]
? 'completed' : ''}`}
                    onClick={() => handleTaskClick(task)}
                    disabled={!account || !!tasksCompleted[task._id]}
                  >
                    {tasksCompleted[task._id] ? "Виконано" : "Виконати"}
                  </Button>
                </div>
              </div>
            </div>
          ))}
      </div>
    </div>
  );
}

export default TaskSystem;
```

А.2. Бекенд модулі

А.2.1. WebSocket сервер

```
// server.js - WebSocket сервер
const WebSocket = require('ws');
const http = require('http');
const app = require('./app');
const server = http.createServer(app);
const wss = new WebSocket.Server({ server });
wss.on('connection', (ws, req) => {
const ip = req.headers['x-forwarded-for'] || req.socket.remoteAddress;
console.log(`Нове WebSocket соединение від ${ip}`);
ws.on('message', (message) => {
const messageText = Buffer.isBuffer(message) ? message.toString() :
message;
console.log('Отримано повідомлення:', messageText);

// Broadcast до всіх клієнтів крім відправника

wss.clients.forEach((client) => {
if (client !== ws && client.readyState === WebSocket.OPEN) {
client.send(message);
}
});
});
ws.on('close', () => {
console.log(`З'єднання з ${ip} закрито`);
});
ws.on('error', (error) => {
console.error(`Помилка WebSocket: ${error.message}`);
});
ws.send('Вітаємо! З'єднання встановлено.');
});
const PORT = process.env.PORT || 80;
server.listen(PORT, () => {
console.log(`Сервер запущено на порту ${PORT}`);
});
```

А.2.2. Helmet.js конфігурація безпеки

```
const helmet = require('helmet');
app.use(helmet({
contentSecurityPolicy: {
directives: {
defaultSrc: ["'self'"],
scriptSrc: ["'self'", "'unsafe-inline'", "https://cdn.jsdelivr.net"],
styleSrc: ["'self'", "'unsafe-inline'", "https://fonts.googleapis.com"],
fontSrc: ["'self'", "https://fonts.gstatic.com"],
imgSrc: ["'self'", "data:", "https:"],
connectSrc: ["'self'", "https://bsc-dataseed.binance.org"],
frameSrc: ["'none'"],
objectSrc: ["'none'"],
upgradeInsecureRequests: []
}
},
hsts: {
maxAge: 31536000,
includeSubDomains: true,
preload: true
},
frameguard: {
action: 'deny'
},
noSniff: true,
xssFilter: true
}));
```

И.3. Контролер користувачів (userController.js)

```
const Wallet = require('../models/Wallet');
const User = require('../models/User');

function formatProfileData(profile) {
  if (!profile) return null;
  return {
    username: profile.username || '',
    avatarUrl: profile.avatarUrl ? `/${profile.avatarUrl.replace(/^\/+/,
'')}` : '',
    points: profile.points || 0,
    tokenBalance: profile.tokenBalance || 0,
    battlePassLevel: profile.battlePassLevel || 1,
```

```
    battlePassProgress: profile.battlePassProgress || 0,
  };
}

exports.getUserProfile = async (req, res) => {
  try {
    const address = req.params.address;
    if (!address) {
      const err = new Error('Адрес не указан');
      err.status = 400;
      throw err;
    }
    const profile = await Wallet.findOne({ address: new
RegExp(`^${address}$`, 'i') });
    return formatProfileData(profile);
  } catch (error) {
    console.error("Помилка в getUserProfile:", error);
    throw error;
  }
};

exports.updateProfile = async (req, res) => {
  const { address, username } = req.body;
  const avatarUrl = req.file ? `/uploads/avatars/${req.file.filename}` :
null;
  try {
    const wallet = await Wallet.findOne({ address });
    if (!wallet) return null;
    wallet.username = username || wallet.username;
    if (avatarUrl) {
      wallet.avatarUrl = avatarUrl;
    }
    await wallet.save();
    return wallet;
  } catch (error) {
    console.error("Помилка під час оновлення профілю:", error);
    throw error;
  }
};
```

И.4. Контролер завдань (taskController.js)

```
const Task = require('../models/Task');
const Wallet = require('../models/Wallet');
const battlePassController = require('./battlePassController');

exports.getTasks = async (req) => {
  const { address } = req.params;
  try {
    if (address.toLowerCase() === 'guest') {
      const tasks = await Task.find({ active: true });
      return { points: 0, completedTasks: [], availableTasks: tasks };
    }

    const wallet = await Wallet.findOne({ address });
    if (!wallet) return null;

    const tasks = await Task.find({ active: true });
    // syncTasks logic omitted for brevity

    return {
      points: wallet.points,
      completedTasks: Array.from(wallet.tasks.entries()),
      availableTasks: tasks
    };
  } catch (error) {
    console.error("Помилка в getTasks:", error);
    throw error;
  }
};

exports.completeTask = async (req) => {
  const { address, taskId, secretCode } = req.body;
  try {
    const wallet = await Wallet.findOne({ address });
    const task = await Task.findById(taskId);

    if (task.secretCode && task.secretCode !== secretCode) {
      return { success: false, status: 400, message: "Невірний секретний
код};
    }

    wallet.tasks.set(taskId, true);
    wallet.points += task.points;
```

```
    await wallet.save();

    // Оновлення Battle Pass
    await battlePassController.updateBattlePass({
        body: { address: address, pointsEarned: task.points },
        ip: req.ip
    });

    return { success: true, message: " Завдання виконане", points:
wallet.points };
  } catch (error) {
    console.error("Помилка в completeTask:", error);
    throw error;
  }
};
```

И.5. Контролер Battle Pass (battlePassController.js)

```
const BattlePassLevel = require('../models/BattlePassLevel');
const Wallet = require('../models/Wallet');

// Розрахунок нового рівня, прогресу та нагороди
function calculateLevelProgress(wallet, allLevels) {
  let currentLevel = wallet.battlePassLevel || 1;
  let currentProgress = wallet.battlePassProgress || 0;
  let earnedTokens = 0;

  while (true) {
    const levelData = allLevels.find(l => l.level === currentLevel);
    if (!levelData) break;

    if (currentProgress >= levelData.pointsRequired) {
      currentProgress -= levelData.pointsRequired;
      earnedTokens += Number(levelData.reward || 0);
      currentLevel += 1;
    } else {
      break;
    }
  }

  return { newLevel: currentLevel, newProgress: currentProgress,
totalTokensEarned: earnedTokens };
}
```

```
exports.updateBattlePass = async (req) => {
  const { address, pointsEarned } = req.body;
  try {
    let wallet = await Wallet.findOne({ address });
    wallet.battlePassProgress += Number(pointsEarned);

    const allLevels = await BattlePassLevel.find().sort({ level: 1 });
    const result = calculateLevelProgress(wallet, allLevels);

    wallet.battlePassLevel = result.newLevel;
    wallet.battlePassProgress = result.newProgress;
    wallet.tokenBalance += result.totalTokensEarned;

    await wallet.save();
    return { battlePassLevel: wallet.battlePassLevel, tokenBalance:
wallet.tokenBalance };
  } catch (error) {
    console.error(`Помилка в updateBattlePass:`, error);

const logger = winston.createLogger({
  level: process.env.LOG_LEVEL || 'info',
  format: winston.format.combine(
    winston.format.timestamp({ format: 'YYYY-MM-DD HH:mm:ss' }),
    winston.format.errors({ stack: true }),
    winston.format.json()
  ),
  defaultMeta: { service: 'university-backend' },
  transports: [
    // Помилки в окремий файл
    new winston.transports.File({
      filename: path.join(__dirname, '../logs/error.log'),
      level: 'error',
      maxsize: 5242880, // 5MB
      maxFiles: 5
    }),
    // Всі логи
    new winston.transports.File({
      filename: path.join(__dirname, '../logs/combined.log'),
      maxsize: 5242880,
      maxFiles: 10
    })
```

```
  ]
});

// В режимі розробки також виводити в консоль
if (process.env.NODE_ENV !== 'production') {
  logger.add(new winston.transports.Console({
    format: winston.format.combine(
      winston.format.colorize(),
      winston.format.simple()
    )
  }));
}

module.exports = logger;
```

A.3. Смарт-контракти

A.3.1. Deploy скрипт для Hardhat

```
// scripts/deploy.js

const hre = require("hardhat");

async function main() {

console.log("Початок деплою смарт-контрактів...");

// Деплой UniversityCoin

const UniversityCoin = await
hre.ethers.getContractFactory("UniversityCoin");
const coin = await UniversityCoin.deploy();
await coin.waitForDeployment();
const coinAddress = await coin.getAddress();
console.log(`UniversityCoin deployed to: ${coinAddress}`);
// Деплой SoulboundDiploma

const SoulboundDiploma = await
hre.ethers.getContractFactory("SoulboundDiploma");
const diploma = await SoulboundDiploma.deploy();
await diploma.waitForDeployment();
const diplomaAddress = await diploma.getAddress();
```

```
console.log(`SoulboundDiploma deployed to: ${diplomaAddress}`);
// Надання ролі MINTER_ROLE backend-серверу

const MINTER_ROLE = await coin.MINTER_ROLE();
const backendAddress = process.env.BACKEND_ADDRESS;
if (backendAddress) {
await coin.grantRole(MINTER_ROLE, backendAddress);
console.log(`MINTER_ROLE granted to backend: ${backendAddress}`);
} else {
console.warn("BACKEND_ADDRESS не встановлено в .env");
}
// Збереження адрес контрактів
const fs = require('fs');
const addresses = {
UniversityCoin: coinAddress,
SoulboundDiploma: diplomaAddress,
network: hre.network.name,
deployedAt: new Date().toISOString()
};
fs.writeFileSync(
'./deployed-addresses.json',
JSON.stringify(addresses, null, 2)
);
console.log("Деплой завершено успішно!");
}
main().catch((error) => {

console.error(error);
process.exitCode = 1;
});
```

A.3.2. Hardhat конфігурація

```
// hardhat.config.js
require("@nomicfoundation/hardhat-toolbox");
require("dotenv").config();
module.exports = {
solidity: {
version: "0.8.20",
settings: {
optimizer: {
```

```
enabled: true,
runs: 200
}
}
},
networks: {
bscTestnet: {
url: "https://data-seed-prebsc-1-s1.binance.org:8545",
chainId: 97,
accounts: process.env.PRIVATE_KEY ? [process.env.PRIVATE_KEY] : [],
gasPrice: 10000000000
},
bscMainnet: {
url: "https://bsc-dataseed.binance.org/",
chainId: 56,
accounts: process.env.PRIVATE_KEY ? [process.env.PRIVATE_KEY] : [],
gasPrice: 5000000000
}
},
etherscan: {
apiKey: {
bscTestnet: process.env.BSCSCAN_API_KEY,
bsc: process.env.BSCSCAN_API_KEY
}
}
};
```