МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗАХИЩЕНОЇ РОЗПОДІЛЕНОЇ ПЛАТФОРМИ ДЛЯ УПРАВЛІННЯ ДИПЛОМАМИ ТА СЕРТИФІКАТАМИ ІЗ ВИКОРИСТАННЯМ БЛОКЧЕЙН-ТЕХНОЛОГІЙ»**

Виконав: студент 2 курсу, групи 8.1224-КН-ОНП
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми Комп'ютерні науки
(назва освітньої програми)
С.І. Дусик
(ініціали та прізвище)
Керівник доцент кафедри комп'ютерних наук, доцент, к.т.н Борю С.Ю.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент завідувач кафедри програмної інженерії, доцент, к.ф.-м.н, Лісняк А.О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти магістр

Спеціальність 122 Комп'ютерні науки
(шифр і назва)

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук, д.т.н., професор

_____ Шило Г.М.
(підпис)

" ____ " _____ 2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Дусику Сергію Івановичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Розробка програмного забезпечення захищеної розподіленої платформи для управління дипломами та сертифікатами із використанням блокчейн-технологій

керівник роботи Борю Сергій Юрійович, к.т.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 28 » лютого 2025 року № 326-с

2. Строк подання студентом роботи 01.06.2026

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Розробка системи видачі дипломів та сертифікатів.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 03.03.2025

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	12.07.2025	
2.	Збір вихідних даних.	01.09.2025	
3.	Обробка теоретичних джерел.	09.10.2025	
4.	Розробка першого та другого розділу.	21.12. 2025	
5.	Розробка третього розділу.	27.03.2026	
6.	Розробка четвертого та п’ятого розділів	12.04.2026	
7.	Оформлення та нормоконтроль	17.05.2026	
	кваліфікаційної роботи магістра.		
8.	Захист кваліфікаційної роботи.	20.06.2026	

Студент _____ С.І. Дусик
(підпис) (ініціали та прізвище)

Керівник роботи _____ С.Ю. Борю
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г. Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка програмного забезпечення захищеної розподіленої платформи для управління дипломами та сертифікатами із використанням блокчейн-технологій»: 65 с., 19 рис., 12 табл., 25 джерел.

БЛОКЧЕЙН, ВЕРИФІКАЦІЯ, ДЕЦЕНТРАЛІЗОВАНИЙ ДОДАТОК, ДИПЛОМ, СЕРТИФІКАТ, СМАРТ-КОНТРАКТ, SOLANA.

Об’єкт дослідження – процес видачі та верифікації освітніх документів у цифровому середовищі.

Предмет дослідження – методи та засоби побудови децентралізованої платформи для управління дипломами та сертифікатами з використанням блокчейн-технологій.

Мета роботи: розробка програмного забезпечення захищеної децентралізованої платформи для створення, зберігання та перевірки освітніх документів із використанням блокчейну.

Методи дослідження – аналіз існуючих рішень, порівняльний аналіз блокчейн-платформ, проєктування архітектури програмного забезпечення, методи розробки децентралізованих додатків, тестування програмного забезпечення.

У кваліфікаційній роботі розглянуто проблему централізованих систем видачі дипломів та сертифікатів, які мають обмежену прозорість, низьку надійність та залежність від єдиного центру зберігання даних. Запропоновано підхід до вирішення цих проблем шляхом використання блокчейн-технологій, що забезпечують незмінність, децентралізацію та відкриту верифікацію даних.

У роботі проведено аналіз існуючих рішень, визначено функціональні та нефункціональні вимоги до системи, обґрунтовано вибір блокчейн-платформи та технологічного стеку. Розроблено прототип децентралізованої

системи на базі блокчейну Solana, що реалізує механізми реєстрації користувачів, створення шаблонів дипломів, їх видачі та перевірки.

Результати роботи підтверджують можливість створення ефективної, надійної та масштабованої системи управління освітніми документами, що може бути використана як основа для подальшого розвитку та впровадження у реальних умовах.

SUMMARY

Bachelor's Qualifying Theses «Development of Software for a Secure Distributed Platform for Managing Diplomas and Certificates Using Blockchain Technologies»: 65 pages, 19 figures, 12 tables, 25 references.

BLOCKCHAIN, CERTIFICATE, DECENTRALIZED APPLICATION, DIPLOMA, SMART CONTRACT, VERIFICATION, SOLANA.

Object of the study – the process of issuing and verifying educational documents in a digital environment.

Subject of the study – methods and tools for building a decentralized platform for managing diplomas and certificates using blockchain technologies.

Aim of the study: development of software for a secure decentralized platform for creating, storing, and verifying educational documents using blockchain.

Methods of research – analysis of existing solutions, comparative analysis of blockchain platforms, software architecture design, decentralized application development methods, software testing.

The qualification work addresses the problem of centralized systems for issuing diplomas and certificates, which suffer from limited transparency, low reliability, and dependence on a single point of control. An approach based on blockchain technology is proposed to overcome these limitations by providing immutability, decentralization, and open verification of data.

The work includes analysis of existing solutions, definition of functional and non-functional requirements, and justification of the selected blockchain platform and technology stack. A prototype of a decentralized system based on the Solana blockchain has been developed, implementing mechanisms for user registration, diploma template creation, issuance, and verification.

The results confirm the feasibility of building an efficient, reliable, and scalable system for managing educational documents, which can serve as a foundation for further development and real-world implementation.

ЗМІСТ

Завдання на кваліфікаційну роботу……..2
Реферат 4
Summary 6
Скорочення та умовні познаки 11
Вступ..... 13
1 Огляд поточних систем видачі дипломів та сертифікатів 15
1.1 Дипломи формальної освіти 15
1.2 Сертифікати неформальної освіти 16
1.3 Блокчейн як технологічна основа вирішення проблем 16
1.3.1 Ключові властивості блокчейну 17
1.3.2 Відповідність блокчейну виявленим проблемам 18
1.4 Висновки 19
2 Постановка задачі та розробка вимог до програмного забезпечення 20
2.1 Функціональні вимоги 20
2.2 Бізнес-логіка 20
2.3 Нефункціональні вимоги 22
3 Розробка системи видачі дипломів та сертифікатів 23
3.1 Етапи розвитку програмного продукту 24
3.1.1 Вибір версії для кваліфікаційної роботи 25
3.2 Вибір блокчейну 25
3.3 Вибір фреймворку 29
3.4 Вибір технологій для клієнтської частини 29
3.5 Структура облікових записів смарт-контракту 30
3.6 Приклади коду смарт-контракту 35
3.6.1 Визначення облікового запису Diploma 35
3.6.2 Інструкція видачі диплому (issue_diploma) 36
3.6.3 Автоматизований тест видачі диплому 38

3.7 Вибір технологій для зберігання зображень (фонів) 41
3.8 Тестування програми 42
3.8.1 Місце тестування у життєвому циклі розробки ПЗ 42
3.8.2 Тестування блокчейн частини 43
4 Аналіз розробленої платформи 45
4.1 Проведення автоматичного тестування 45
4.2 Запуск додатку 46
4.2.1 Запуск локального валідатора 46
4.2.2 Компіляція та розгортання програми 46
4.3 Запуск скрипта на створення акаунту адміністратора 47
4.4 Запуск веб-додатку 47
4.5 Авторизація адміністратором 48
4.6 Використання платформи емітентом 48
4.6.1 Реєстрація акаунту 48
4.6.2 Створення шаблону диплому 49
4.6.3 Видача диплому 50
4.7 Функціонал адміністратора 50
4.7.1 Підтвердження акаунту емітента 50
4.7.2 Підтвердження шаблону 51
4.7.3 Підтвердження акаунту студента 51
4.8 Функціонал користувача 52
4.8.1 Створення акаунту 52
4.8.2 Підтвердження диплому 52
4.9 Перевірка освітнього документу за ідентифікатором 53
5 Подальший розвиток продукту 54
5.1 Розширення функціоналу управління емітентами 54
5.2 Підтримка багатомовності 55
5.3 Покращення користувацького інтерфейсу 55
5.4 Підтримка світлої та темної теми інтерфейсу 56
5.5 Розширення тестового покриття системи 56

5.6 Оптимізація структури облікових записів 57
5.7 Розробка мобільного додатку 58
5.8 Інтеграція з державними інформаційними системами 58
5.9 Загальні перспективи розвитку 59
Висновки 60
Перелік посилань 63

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

Акаунт у Solana – це базова структурна одиниця зберігання даних у мережі, яка може містити як інформацію про баланс токенів, так і стан програм (смартконтрактів)

Алгоритм консенсусу – механізм досягнення згоди між вузлами мережі щодо актуального стану блокчейну [18]

Блокчейн (Blockchain) – розподілений реєстр даних, у якому інформація зберігається у вигляді послідовності криптографічно пов’язаних блоків та підтримується мережею незалежних вузлів [11]

Валідатор – вузол блокчейн-мережі, що бере участь у підтвердженні транзакцій та створенні нових блоків

Децентралізований додаток (dApp) – програмна система, що використовує блокчейн як бекенд для виконання бізнес-логіки та зберігання даних [7]

Приватний ключ – секретний криптографічний ключ, що використовується для підпису транзакцій

Публічний ключ – криптографічний ідентифікатор користувача в мережі

Смартконтракт (Smart Contract) – програма, що виконується у блокчейн-мережі та автоматично забезпечує виконання визначених правил і умов [24]

Транзакція – операція запису або зміни стану даних у блокчейні

Хеш-функція – криптографічна функція, що перетворює вхідні дані у фіксований набір символів (хеш)

Byzantine Fault Tolerance (BFT) – модель відмовостійкості, що дозволяє системі коректно працювати навіть за наявності частини

	зловмисних або несправних вузлів
IDL (Interface Definition Language) –	формалізований опис інтерфейсу смартконтракту для інтеграції з клієнтською частиною
IPFS (InterPlanetary File System) –	децентралізована система зберігання файлів із адресацією за хешем вмісту
Off-chain –	дані або операції, що зберігаються поза блокчейном
On-chain –	дані або операції, що зберігаються безпосередньо у блокчейні
PDA (Program Derived Address) –	спеціально згенерована адреса акаунта в Solana, яка контролюється програмою

ВСТУП

У сучасних умовах стрімкого розвитку інформаційних технологій та цифровізації суспільства відбувається активна трансформація освітньої сфери. Значного поширення набувають як дистанційні форми навчання, так і різноманітні онлайн-платформи, що надають можливість отримання знань незалежно від географічного розташування користувача. У таких умовах особливої актуальності набуває питання цифрового представлення та підтвердження освітніх досягнень, зокрема дипломів та сертифікатів.

Традиційні підходи до зберігання та верифікації освітніх документів, що базуються на централізованих інформаційних системах або паперових носіях, мають низку суттєвих недоліків. Серед них можна виділити обмежену прозорість процесів, залежність від єдиного центру зберігання даних, ризики втрати або підробки документів, а також складність міжнародної перевірки їх достовірності. Крім того, існуючі рішення часто вимагають залучення третьої сторони для підтвердження інформації, що уповільнює процеси взаємодії між освітніми установами, роботодавцями та користувачами.

Паралельно з цим активно розвиваються технології розподілених систем, серед яких особливе місце займає блокчейн. Його ключові властивості, такі як децентралізація, незмінність даних, криптографічний захист та можливість відкритої верифікації, відкривають нові перспективи для побудови надійних і прозорих систем обліку інформації. Використання блокчейну дозволяє усунути єдину точку відмови, забезпечити цілісність даних та надати можливість незалежної перевірки записів без необхідності довіри до централізованих органів.

Аналіз існуючих систем видачі дипломів та сертифікатів показує, що на сьогодні відсутнє універсальне рішення, яке б одночасно забезпечувало високий рівень безпеки, зручність використання, гнучкість налаштування та доступність для широкого кола користувачів. Це обумовлює необхідність

розробки нових підходів до організації процесів створення, зберігання та перевірки освітніх документів із використанням сучасних технологій.

Метою даної дипломної роботи є розробка програмного забезпечення децентралізованої платформи для управління дипломами та сертифікатами з використанням блокчейн-технологій. Запропоноване рішення спрямоване на забезпечення прозорого, надійного та ефективного механізму видачі й верифікації освітніх документів, а також на підвищення рівня довіри до них з боку користувачів, освітніх установ і роботодавців.

1 ОГЛЯД ПОТОЧНИХ СИСТЕМ ВИДАЧІ ДИПЛОМІВ ТА СЕРТИФІКАТІВ

На сьогодні більшість систем видачі дипломів та сертифікатів побудовані за централізованою архітектурою. Дані про видані документи зберігаються у централізованих базах даних, які повністю контролюються уповноваженими органами або приватними організаціями. Такий підхід має низку недоліків, серед яких обмежена прозорість процесів, наявність єдиної точки відмови та високі ризики несанкціонованого доступу або втрати даних.

У випадку кібератаки, технічного збою або людського фактору централізований сервер може стати недоступним, що призводить до повної зупинки роботи системи перевірки дипломів та сертифікатів. Додатковою проблемою є необхідність довіри до третьої сторони – державних установ або приватних компаній, які виступають єдиними власниками та розпорядниками даних.

Окрім державних систем, значну роль відіграють сторонні видавці сертифікатів – міжнародні освітні платформи, IT-компанії та навчальні центри, які використовують власні закриті формати зберігання та верифікації документів.

1.1 Дипломи формальної освіти

У сфері формальної освіти в Україні використовується централізована державна система обліку документів про освіту [4, 5]. Вона забезпечує зберігання даних про видані дипломи, атестати та додатки до них у державному реєстрі.

Основні характеристики такої системи:

– централізоване зберігання даних;

– обмежений доступ до повної інформації;
– залежність від стабільності роботи центральної інфраструктури;
– складність інтеграції з міжнародними системами верифікації.

Процедура перевірки диплома зазвичай вимагає запиту до державного реєстру або надання паперової копії документа [6]. Це ускладнює автоматизовану перевірку та створює додаткові часові затримки для роботодавців та міжнародних установ.

1.2 Сертифікати неформальної освіти

Системи неформальної освіти представлені великою кількістю приватних платформ та організацій. До найпоширеніших належать міжнародні онлайн-платформи навчання, галузеві сертифікаційні центри та корпоративні освітні системи [14].

Такі сертифікати мають низку особливостей:

– відсутність єдиного стандарту формату;
– можливість видачі цифрових сертифікатів у вигляді PDF або веб-посилань;
– перевірка автентичності через сайти видавця або унікальні коди.

Недоліком цих систем є те, що перевірка справжності залежить від доступності сервера компанії-видавця. У разі припинення діяльності платформи або технічних проблем верифікація сертифіката стає неможливою [2].

1.3 Блокчейн як технологічна основа вирішення проблем

Блокчейн (від англ. blockchain – ланцюг блоків) – це розподілена база даних, що являє собою послідовний ланцюжок блоків, кожен з яких містить набір записів та криптографічний хеш попереднього блоку [18]. Така структура

унеможливлює непомітну зміну вже записаних даних: модифікація будь-якого блоку змінює його хеш, що робить усі наступні блоки недійсними і негайно виявляється мережею.

На відміну від традиційних баз даних, блокчейн не має єдиного центрального сервера – його копії одночасно зберігаються на тисячах незалежних вузлів по всьому світу [21]. Консенсусний механізм забезпечує узгодженість стану мережі: нові записи вважаються дійсними лише тоді, коли більшість вузлів підтвердила їх коректність.

1.3.1 Ключові властивості блокчейну

Технологія блокчейн забезпечує такі властивості:

Незмінність. Дані, записані в блокчейн, не можна змінити або видалити без перебудови всього наступного ланцюга, що потребує контролю над більшістю вузлів мережі. На практиці це робить підробку записів економічно та технічно недоцільною [11].

Децентралізація. Відсутність єдиного сервера усуває єдину точку відмови: навіть якщо частина вузлів виходить з ладу, мережа продовжує функціонувати, а дані залишаються доступними.

Прозорість і відкрита верифікація. Будь-який запис у публічному блокчейні може бути перевірений будь-ким без звернення до організації, що його зробила. Верифікація здійснюється математично – через перевірку хешів і цифрових підписів [21].

Криптографічний захист. Кожен запис підписується приватним ключем автора. Цифровий підпис гарантує автентичність – підтверджує, що запис зробила саме та особа або організація, якій належить відповідний ключ.

Смарт-контракти. Програмний код, розгорнутий безпосередньо у блокчейні, що виконується автоматично при настанні визначених умов. Дозволяє кодифікувати бізнес-логіку у вигляді незмінних і прозорих правил без участі посередника.

1.3.2 Відповідність блокчейну виявленим проблемам

Аналіз існуючих систем видачі дипломів та сертифікатів (розділ 1) виявив чотири системні проблеми. Розглянемо, яким чином властивості блокчейну усувають кожну з них (табл. 1.1).

Таблиця 1.1 – Відповідність властивостей блокчейну виявленим проблемам централізованих систем

Проблема існуючих систем	**Як блокчейн її вирішує**
Централізоване зберігання даних – єдина точка відмови. Збій або атака на центральний сервер повністю зупиняє роботу системи.	Децентралізація: дані копіюються на тисячах незалежних вузлів. Відключення будь-якої їх кількості не впливає на доступність записів.
Залежність від одного власника інфраструктури – оператор може змінити, видалити або заблокувати записи.	Незмінність: жоден суб'єкт не може одноосібно змінити запис у блокчейні. Дані захищені криптографічно та підтверджені консенсусом мережі.
Складність міжнародної верифікації – перевірка вимагає звернення до конкретного реєстру або паперового документа.	Відкрита верифікація: будь-хто у будь-якій країні може перевірити автентичність диплому за його ідентифікатором без запитів до університету чи державного органу [13].
Низький рівень автоматизації – процеси реєстрації, підтвердження та видачі виконуються вручну через адміністраторів.	Смарт-контракти: правила бізнес-логіки (реєстрація, підтвердження, видача) кодифіковані у незмінному коді та виконуються автоматично за умовами, без ручного втручання.

Таким чином, блокчейн є не просто альтернативним сховищем даних – він забезпечує принципово інші гарантії цілісності, доступності та прозорості, які недосяжні в централізованих архітектурах. Саме ці властивості роблять блокчейн обґрунтованим технологічним вибором для платформи управління дипломами та сертифікатами [9].

1.4 Висновки

Аналіз існуючих систем видачі дипломів та сертифікатів показує, що більшість з них мають спільні архітектурні риси: централізоване зберігання, наявність єдиного центру довіри та обмежену прозорість.

До переваг чинних систем можна віднести:

- простоту адміністративного контролю;
- усталені юридичні процедури;
- відносну легкість впровадження.

Основні недоліки:

- вразливість до кібератак та збоїв;
- залежність від одного власника інфраструктури;
- складність міжнародної верифікації;
- низький рівень автоматизації перевірки достовірності документів.

Отримані результати обґрунтовують доцільність розробки розподіленої платформи на основі блокчейн-технологій, яка дозволить усунути єдину точку відмови, підвищити прозорість процесів та забезпечити доступну можливість перевірки освітніх документів.

2 ПОСТАНОВКА ЗАДАЧІ ТА РОЗРОБКА ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Метою роботи є розробка програмного забезпечення захищеної розподіленої платформи для управління дипломами та сертифікатами із використанням блокчейн-технологій.

2.1 Функціональні вимоги

Це детальний опис того, що система повинна робити: які функції виконувати, як обробляти дані, які операції виконувати та як взаємодіяти з користувачем (наприклад, реєстрація, пошук, розрахунки, звітність) [16].
На початковому етапі передбачено такі функціональні вимоги:

- реєстрація користувачів;
- реєстрація емітентів (з підтвердженням адміністратором);
- створення шаблонів (підтвердження шаблонів адміністратором);
- видача диплому;
- підтвердження диплому користувачем (користувач може вирішити не приймати диплом);
- перевірка диплому за унікальним ідентифікаційним номером.

2.2 Бізнес-логіка

У системі є адміністратори, емітенти та користувачі. Емітенти – це навчальні заклади, які видають дипломи та сертифікати. Користувачі – це ті хто може отримати диплом. Адміністратори приймають участь у реєстрації

емітентів (затверджують акаунти) та підтвердженні шаблонів для дипломів. Взаємодія між ролями наведена на рисунку 2.1.

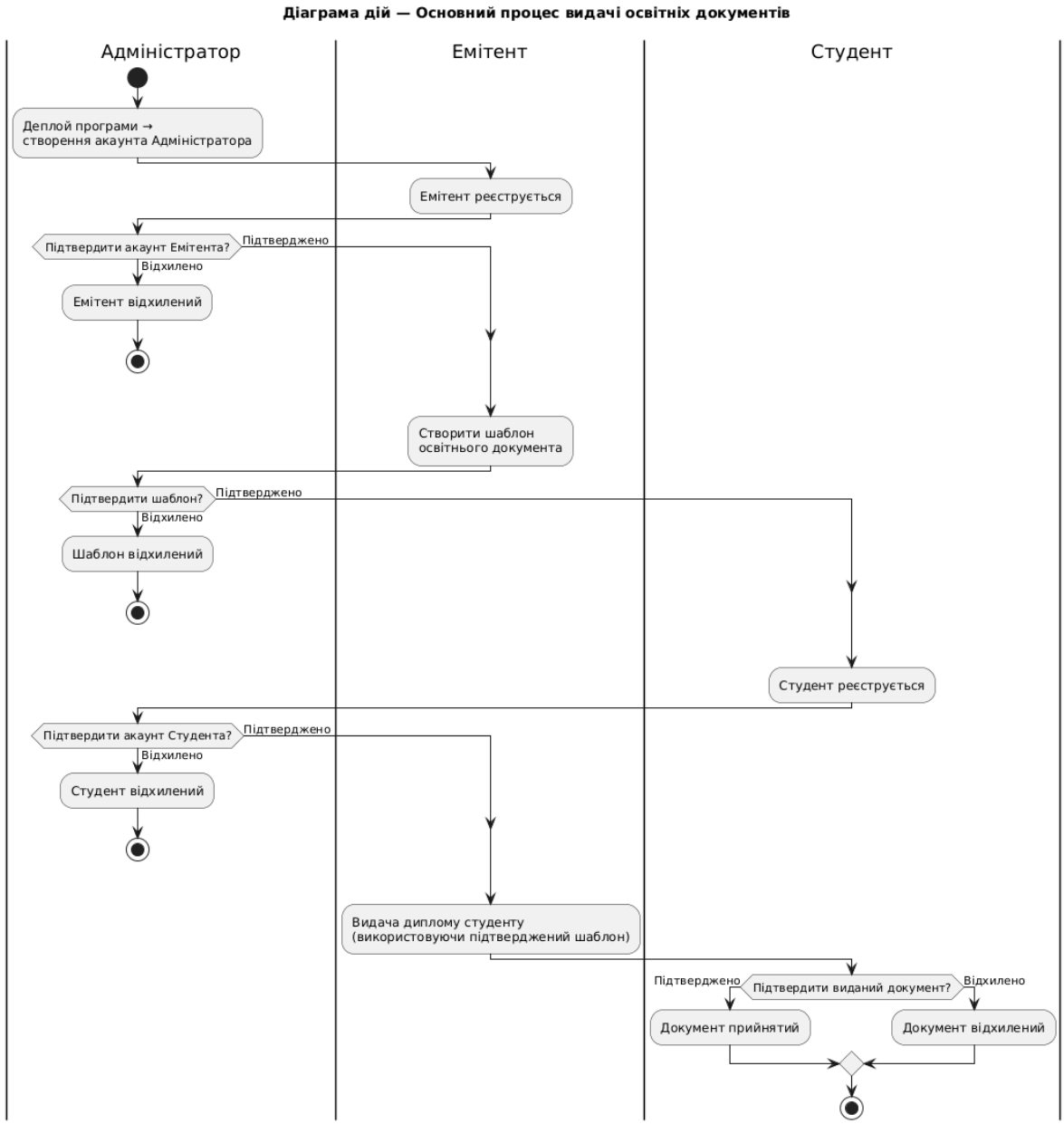


Рисунок 2.1 – Діаграма дій платформи

Користувач – зареєстрована приватна особа, яка може отримати диплом або сертифікат.

Емітент – зареєстрований та підтверджений адміністратором навчальний заклад, що може видавати дипломи або сертифікати.

Адміністратор – офіційний керівний акаунт платформи. Використовується для підтверджень емітентів, шаблонів та облікових записів користувачів.

2.3 Нефункціональні вимоги

Ці вимоги є інструкціями, які описують як система повинна працювати, а не що вона має робити, визначаючи атрибути її якості: швидкість, безпеку, зручність використання, надійність, масштабованість, якість програмного коду тощо, забезпечуючи ефективність та задоволеність користувачів [16].

Надійність – здатність системи стабільно та коректно функціонувати протягом визначеного часу без збоїв і помилок. Надійність забезпечується шляхом використання механізмів обробки помилок, резервування та автоматизованого тестування [1].

Захищеність – властивість системи забезпечувати захист даних і ресурсів від несанкціонованого доступу, модифікації чи знищення. Включає автентифікацію, авторизацію, шифрування даних і захист від типових кіберзагроз.

Швидкодія – здатність системи обробляти запити та виконувати операції в допустимий час за заданого навантаження. Характеризується часом відгуку, пропускною здатністю та ефективним використанням обчислювальних ресурсів.

Доступність – здатність системи бути доступною для користувачів у будь-який необхідний момент часу. Визначається відсотком часу безперервної роботи системи та стійкістю до відмов.

Зручність використання – міра того, наскільки легко користувачам навчатися роботі з системою, ефективно її використовувати та отримувати задоволення від взаємодії з інтерфейсом. Включає інтуїтивність інтерфейсу, зрозумілість навігації та мінімізацію помилок користувача.

3 РОЗРОБКА СИСТЕМИ ВИДАЧІ ДИПЛОМІВ ТА СЕРТИФІКАТІВ

Розробка децентралізованих додатків (Decentralized Applications, dApp) має свої особливості порівняно з класичними клієнт-серверними системами. Життєвий цикл розробки таких систем зазвичай включає аналіз предметної області, проєктування архітектури смарт-контрактів, вибір блокчейн-платформи, реалізацію бізнес-логіки, інтеграцію з користувацьким інтерфейсом та тестування в умовах розподіленого середовища.

В данному випадку децентралізований додаток буде складається з декількох ключових компонентів:

- смарт-контрактів, що зберігаються та виконуються безпосередньо в блокчейні й відповідають за основну бізнес-логіку та правила системи;
- децентралізованого або напівдецентралізованого сховища даних, у якому можуть зберігатися метадані або великі файли (наприклад, IPFS);
- бекенд-інфраструктури, яка забезпечує взаємодію між блокчейном і зовнішніми сервісами (авторизація, індексація подій, кешування);
- фронтенд-інтерфейсу, що надає користувачам доступ до функціоналу системи.

У контексті платформи управління дипломами та сертифікатами блокчейн використовується як незмінний та публічно перевірюваний реєстр, у якому фіксуються факти видачі та перевірки документів, що підвищує рівень довіри та захищеності системи.

3.1 Етапи розвитку програмного продукту

Розробка сучасних програмних систем зазвичай відбувається поетапно. Такий підхід дозволяє поступово перевіряти ідеї, оцінювати технічну реалізованість та зменшувати ризики під час створення складних програмних рішень. У практиці розробки програмного забезпечення часто виділяють три основні етапи розвитку продукту: Proof of Concept (PoC)**,** Minimum Viable Product (MVP) та Production-версія системи [16]**.**

Proof of Concept (PoC) – це початковий етап, метою якого є підтвердження технічної можливості реалізації запропонованої ідеї. На цьому етапі створюється експериментальний прототип, який демонструє працездатність ключових механізмів системи. PoC зазвичай має обмежений функціонал, не орієнтований на кінцевих користувачів і може не відповідати вимогам до якості коду, безпеки або масштабованості. Основне завдання цього етапу полягає у відповіді на питання: чи можливо реалізувати запропоноване рішення з використанням обраних технологій**.**

Minimum Viable Product (MVP) – мінімально життєздатна версія продукту. MVP містить лише базовий набір функцій, необхідних для демонстрації цінності системи та її використання першими користувачами. Головною метою цього етапу є перевірка практичної доцільності розробки та отримання зворотного зв’язку від користувачів. MVP дозволяє оцінити попит на продукт, визначити необхідні покращення та скоригувати подальший напрямок розвитку системи.

Завершальним етапом є **Production-версія системи**, яка призначена для повноцінного використання у реальних умовах. На цьому етапі особлива увага приділяється стабільності роботи, безпеці, продуктивності та масштабованості програмного забезпечення. Production-версія включає повний функціонал системи, проходить тестування та оптимізацію, а також інтегрується з інфраструктурними компонентами, такими як системи моніторингу, резервного копіювання та автоматизованого розгортання.

3.1.1 Вибір версії для кваліфікаційної роботи

У межах даної кваліфікаційної роботи було реалізовано етап Proof of Concept (PoC). Основною метою розробки було підтвердження технічної можливості створення децентралізованої системи видачі та верифікації освітніх документів із використанням сучасних технологій розподілених систем та блокчейн-інфраструктури. Реалізований прототип демонструє працездатність ключових компонентів системи, зокрема механізмів створення, збереження та перевірки дипломів, а також взаємодію між серверною частиною, блокчейн-мережею та веб-інтерфейсом.

Створена система покрита автоматизованими тестами, що дозволяє перевіряти коректність роботи основних функцій. Крім того, розроблено простий веб-інтерфейс, який забезпечує базову взаємодію користувача із системою та демонструє можливість практичного використання розробленого рішення.

Таким чином, розроблений прототип повністю відповідає цілям етапу PoC та дозволяє підтвердити доцільність подальшого розвитку системи до рівня MVP, а в подальшому – до повноцінної production-версії.

3.2 Вибір блокчейну

Сучасний ринок блокчейн-технологій представлений великою кількістю платформ, які відрізняються архітектурою, механізмами консенсусу та технічними характеристиками. Найбільш популярними серед розробників децентралізованих додатків є Ethereum, Solana, Polkadot та Cosmos.

Ethereum є однією з найпоширеніших платформ із розвиненою екосистемою, проте має відносно високі комісії та обмежену продуктивність. Альтернативні рішення, зокрема Solana, орієнтовані на забезпечення високої

швидкодії та низької вартості транзакцій. Polkadot і Cosmos реалізують підхід взаємодії між різними блокчейн-мережами, що дозволяє підвищити масштабованість і гнучкість систем.

Таким чином, кожна з платформ має свої особливості, що зумовлює необхідність їх порівняння за ключовими критеріями (табл. 3.1).

Таблиця 3.1 – Порівняльний аналіз блокчейн-платформ за ключовими критеріями

Назва	**Мова програмування**	**Консенсус**	**Комісія**	**Повна фіналізація**
Ethereum	Solidity, Vyper	Proof of Stake	~0.1%–5%+	~1–2 хв
Solana	Rust, C, C++	Proof of History + PoS	< 0.1%	~1–2 с
Polkadot	Rust, ink!	Nominated PoS	0.2%–2%	~30 с
Cosmos	Rust, Go	Tendermint	0.1%–1%	~2–10 с

З-поміж усіх критеріїв найбільш критичними для платформи є швидкодія та розмір комісії, оскільки кожна операція (видача, підтвердження, верифікація) генерує транзакцію в блокчейні.

Розмір комісії в Solana є найнижчим, навіть найбільша можлива комісія нижча ніж у найближчого конкурента – Cosmos.

Швидкодія блокчейн-систем є критично важливою характеристикою, оскільки вона визначає час очікування користувача між ініціюванням операції та її підтвердженням у мережі. Історично перші блокчейни мали низьку пропускну здатність і значні затримки через обмежений розмір блоків і складні алгоритми консенсусу. З розвитком технологій з’явилися нові підходи, спрямовані на підвищення продуктивності: оптимізація мережевої взаємодії, паралельна обробка транзакцій та використання нових механізмів консенсусу. У сучасних блокчейнах, таких як Solana, час підтвердження транзакцій може становити менше однієї секунди, що робить їх придатними для інтерактивних децентралізованих додатків [20, 24]. Не останню

роль відіграє мова програмування: безпека пам’яті, статична типізація, ідеальним варіантом, який забезпечує всі ці вимоги є Rust [15].

Також варто звернути увагу на **алгоритм консенсусу**, оскільки саме він визначає спосіб досягнення згоди між вузлами мережі щодо стану блокчейну. Алгоритм консенсусу безпосередньо впливає на рівень безпеки, децентралізації, швидкодії та енергоспоживання мережі. Найбільш поширеними підходами є Proof of Work, Proof of Stake та їх похідні. Proof of Stake і його варіації забезпечують значно вищу енергоефективність і швидше підтвердження транзакцій, що робить їх більш придатними для сучасних децентралізованих платформ. Комбіновані підходи, такі як Proof of History у Solana або Tendermint у Cosmos, дозволяють досягати високої продуктивності без суттєвого зниження рівня безпеки [25].

Висновок. За результатами порівняльного аналізу для реалізації платформи управління дипломами та сертифікатами було обрано блокчейн Solana. Вибір обґрунтований трьома ключовими факторами.

По-перше, Solana забезпечує найвищу пропускну здатність серед розглянутих платформ – до 65 000 транзакцій на секунду – завдяки механізму Proof of History (PoH), який дозволяє вузлам мережі встановлювати хронологічний порядок подій без додаткових раундів комунікації. Для платформи видачі дипломів це означає миттєве підтвердження операцій без затримок, помітних для користувача.

По-друге, комісія за транзакцію у мережі Solana є найнижчою серед конкурентів і становить у середньому менше 0,001 USD. У контексті платформи, де кожна видача диплому, підтвердження шаблону або реєстрація користувача є окремою транзакцією, низька комісія безпосередньо впливає на операційну вартість системи для навчальних закладів та їхніх студентів.

По-третє, Solana має зрілу екосистему інструментів для розробки, зокрема фреймворк Anchor, який суттєво спрощує написання, тестування та розгортання смарт-контрактів порівняно з нативним Rust API. Anchor забезпечує

декларативну валідацію акаунтів, автоматичне обчислення розмірів структур та зручний TypeScript-клієнт для взаємодії з програмою.

Таким чином, Solana є оптимальним вибором з огляду на продуктивність, економічність та зручність розробки, що відповідає як технічним, так і бізнес-вимогам платформи.

3.3 Вибір фреймворку

Написання децентралізованого додатку на Solana за допомогою мови програмування Rust можливе двома шляхами: нативно, без використання фреймворків та за допомогою фреймворку Anchor (табл. 3.2) [10].

Таблиця 3.2 – Порівняльний аналіз підходів до розробки смарт-контрактів на Solana

Критерій	Нативно	Anchor
Рівень абстракції	Низький (повний контроль)	Високий (DSL над Solana)
Об’єм boilerplate-коду	Дуже багато	Мінімальний
Опис акаунтів	Ручна перевірка AccountInfo, owner, signer	Декларативно за допомогою макросів
Серіалізація даних	Ручна	Автоматична (Borsh за замовчуванням)
Безпека	Високий ризик помилок	Менше помилок завдяки constraints
Інтеграція з frontend	Ручна	Дуже проста (IDL + client)
Розмір програми (BPF)	Менший	Трохи більший
Продуктивність	Максимальна	Близька до максималь ної

3.4 Вибір технологій для клієнтської частини

Клієнтська частина платформи забезпечує взаємодію користувачів зі смарт-контрактом через веб-браузер. Хоча на стадії розробки програмного забезпечення PoC інтерфейс користувача у більшості випадків не вимагається, він буде чудовим інструментом для демонстрації роботи системи широкій аудиторії. Оскільки платформа є децентралізованим додатком, до фронтенд-фреймворку висуваються специфічні вимоги: наявність бібліотек для роботи з крипто-гаманцями Solana, підтримка реактивного управління станом для відображення даних із блокчейну, а також зрілість екосистеми для швидкої розробки прототипу [19]. Для вибору оптимального рішення проведено порівняльний аналіз популярних JavaScript-фреймворків, які підтримують

типізацію за допомогою TypeScript для запобігання помилкам на етапі розробки (табл. 3.3) [22].

Таблиця 3.3 – Порівняльний аналіз технологій клієнтської частини

Технологія	**Підтримка Solana / Web3**	**Продуктивність**	**Складність навчання**
React	Найвища	Висока	Середня
Vue.js	середня	Висока	Середня
Svelte	добра	Дуже висока	Низька – Середня
Solid.js	погана	Дуже висока	Середня
Angular	погана	Середня	Висока
HTML/JS	добра	Середня	Низька

За результатами порівняльного аналізу для розробки клієнтської частини обрано бібліотеку React 19 у поєднанні зі збірником Vite [23]. Вибір обґрунтований найширшою екосистемою готових рішень для Solana: офіційний пакет @solana/wallet-adapter-react забезпечує підключення гаманців, @tanstack/react-query – кешування та синхронізацію даних із блокчейном. Для стилізації компонентів використано Tailwind CSS v4 та бібліотеку Material UI, що дозволяє швидко реалізовувати інтерфейс без написання стилів вручну.

3.5 Структура облікових записів смарт-контракту

У блокчейні Solana дані зберігаються у вигляді облікових записів (акаунтів) [20]. Фреймворк Anchor дозволяє декларативно описувати структури таких облікових записів за допомогою атрибуту #[account]. Для забезпечення детермінованої адресації використовуються PDA (Program Derived Address) – адреси, що обчислюються із заздалегідь відомих шматочків даних (seeds), а не генеруються випадково.

Розроблений смарт-контракт визначає сім типів облікових записів, що зберігаються безпосередньо у блокчейні. Нижче наведено опис кожної сутності та перелік її полів.

Admin (Адміністратор) – єдиний системний обліковий запис, що ініціалізується при розгортанні контракту та зберігає дані про керівний акаунт платформи). PDA-адреса обчислюється з seed рядка “admin” (табл. 3.4).

Таблиця 3.4 – Поля облікового запису Admin

Поле	Тип	Призначення
Authority	Pubkey	Публічний ключ (адреса гаманця) адміністратора – єдиного акаунту, що має право затверджувати або відхиляти емітентів, шаблони та користувачів
created_at_timestamp	u64	Unix-мітка часу ініціалізації облікового запису адміністратора

Емітент (Issuer) – акаунт навчального закладу або організації, що має право видавати дипломи та сертифікати. Реєструється самостійно, після чого очікує підтвердження від адміністратора. PDA-адреса обчислюється з seed рядка “issuer” та публічного ключа емітента (табл. 3.5).

Таблиця 3.5 – Поля облікового запису Issuer

Поле	Тип	Призначення
authority	Pubkey	Публічний ключ гаманця емітента – ідентифікує власника облікового запису
name	String	Назва організації-емітента (від 5 до 50 символів)
government_country_code	Option<CountryCode>	Необов’язковий код країни реєстрації емітента (підтримуються: UA, US, CA)
status	IssuerStatus	Поточний статус облікового запису: AwaitingApproval (очікує розгляду), Approved (підтверджений), Rejected(String) (відхилений із зазначенням причини)
registered_at_timestamp	u64	Unix-мітка часу реєстрації емітента в системі
reviewed_at_timestamp	Option<u64>	Unix-мітка часу, коли адміністратор розглянув заявку емітента

Користувач (User) – представляє фізичну особу, яка може отримувати дипломи та сертифікати. Реєструється самостійно та потребує підтвердження адміністратором. PDA-адреса обчислюється з рядку “user” та публічного ключа користувача (табл. 3.6).

Таблиця 3.6 – Поля облікового запису User

Поле	Тип	Призначення
authority	Pubkey	Публічний ключ гаманця користувача – ідентифікує власника облікового запису
first_name	String	Ім’я користувача (від 2 до 30 символів)
middle_name	Option<String>	По батькові користувача (необов’язкове, від 2 до 30 символів)
last_name	String	Прізвище користувача (від 2 до 30 символів)
status	UserStatus	Поточний статус: AwaitingApproval, Approved або Rejected(String)
external_id	u64	Зовнішній числовий ідентифікатор із офчейн-системи (наприклад, ID студента в університетській базі даних)
registered_at_timestamp	u64	Unix-мітка часу реєстрації користувача в системі
reviewed_at_timestamp	Option<u64>	Unix-мітка часу розгляду заявки адміністратором

Шаблон диплому (DiplomaTemplate) – визначає тип або форму диплому, який емітент може видавати. Перед використанням потребує підтвердження адміністратором. Один емітент може мати кілька шаблонів (наприклад, окремі для бакалаврів і магістрів). PDA-адреса обчислюється з seed рядка “diploma-template” та числового ідентифікатора шаблону (табл. 3.7).

Таблиця 3.7 – Поля облікового запису DiplomaTemplate

Поле	Тип	Призначення
id	u64	Унікальний числовий ідентифікатор шаблону (присвоюється автоматично через глобальний лічильник)
authority	Pubkey	Публічний ключ гаманця емітента, який створив шаблон
name	String	Назва шаблону (від 5 до 50 символів)

Продовження таблиці 3.7

Поле	Тип	Призначення
background_hash	Option<String>	CIDv1-хеш IPFS-ресурсу із зображенням фону диплому (59 символів); необов'язкове поле
status	DiplomaTemplateStatus	Статус шаблону: AwaitingApproval, Approved або Rejected(String)
expiration_time	Option<u64>	Необов'язковий термін дії диплому у секундах (від 1 до 10 років); якщо вказано – передається до кожного виданого диплому
country_code	Option<CountryCode>	Код країни, успадкований від облікового запису емітента
created_at_timestamp	u64	Unix-мітка часу створення шаблону
reviewed_at_timestamp	Option<u64>	Unix-мітка часу розгляду шаблону адміністратором

Лічильник шаблонів (DiplomaTemplateCounter) – допоміжний глобальний обліковий запис для автоматичного присвоєння унікальних ідентифікаторів шаблонам дипломів. PDA-адреса обчислюється з рядку "diploma-template-counter" (табл. 3.8).

Таблиця 3.8 – Поля облікового запису DiplomaTemplateCounter

Поле	Тип	Призначення
data	u64	Поточне значення лічильника; збільшується на 1 при кожному створенні нового шаблону

Диплом (Diploma) – основна сутність системи, конкретний виданий документ, що підтверджує освітнє досягнення. Формується емітентом на основі затвердженого шаблону та прив'язується до конкретного користувача. Користувач повинен підтвердити або відхилити отриманий диплом. PDA-адреса обчислюється з рядку "diploma" та числового ідентифікатора диплому (табл. 3.9).

Таблиця 3.9 – Поля облікового запису Diploma

Поле	Тип	Призначення
id	u64	Унікальний числовий ідентифікатор диплому (присвоюється автоматично через глобальний лічильник)
name	String	Назва диплому (від 5 до 50 символів), успадкована із шаблону
issuer	Pubkey	Публічний ключ облікового запису емітента, що видав диплом
user	Pubkey	Публічний ключ облікового запису користувача – отримувача диплому
template	Pubkey	Публічний ключ шаблону, на основі якого видано диплом
background_hash	Option<String>	CIDv1-хеш IPFS-ресурсу із зображенням фону, успадкований із шаблону
status	DiplomaStatus	Статус диплому: AwaitingApproval (виданий, очікує підтвердження), Approved (підтверджений користувачем), Rejected(String) (відхилений із причиною)
valid_until	Option<u64>	Unix-мітка часу, до якої дійсний диплом (= час видачі + expiration_time шаблону); None означає безстроковий диплом
country_code	Option<CountryCode>	Код країни, у якому діє диплом, успадкований із шаблону
metadata	Option<String>	Необов’язкові додаткові відомості (від 5 до 50 символів), наприклад спеціальність або середній бал
issued_at_timestamp	u64	Unix-мітка часу видачі диплому
reviewed_at_timestamp	Option<u64>	Unix-мітка часу, коли користувач підтвердив або відхилив диплом

Лічильник дипломів (DiplomaCounter) – аналогічний до лічильника шаблонів, забезпечує унікальність числових ідентифікаторів дипломів у межах усієї платформи. PDA-адреса обчислюється з рядка “diploma-counter” (табл. 3.10).

Таблиця 3.10 – Поля облікового запису DiplomaCounter

Поле	Тип	Призначення
data	u64	Поточне значення лічильника; збільшується на 1 при кожній видачі нового диплому

Зв’язки між обліковими записами реалізовані через публічні ключі (Pubkey), що відповідає моделі даних Solana та забезпечує можливість незалежної верифікації будь-якого запису безпосередньо в блокчейні без звернення до централізованих сервісів.

3.6 Приклади коду смарт-контракту

Для ілюстрації практичної реалізації платформи нижче наведено три характерні фрагменти коду: визначення структури облікового запису, логіка інструкції смарт-контракту та автоматизований тест. Усі приклади стосуються ключової операції системи – видачі диплому.

3.6.1 Визначення облікового запису Diploma

У фреймворку Anchor кожен обліковий запис описується як структура Rust з атрибутом #[account]. Атрибут #[derive(InitSpace)] автоматично обчислює розмір, необхідний для виділення пам’яті у блокчейні, а макроси #[max_len(...)] обмежують довжину рядкових полів. Нижче наведено визначення структури Diploma – основної сутності системи (рис. 3.1).

```
#[account]
#[derive(InitSpace, Debug)]
pub struct Diploma {
    #[max_len(DIPLOMA_NAME_MAX_LENGTH)]
    pub name: String,          // назва диплому (з шаблону)

    pub id: u64,               // унікальний ідентифікатор

    pub issuer:   Pubkey,      // PDA облікового запису емітента
    pub user:     Pubkey,      // PDA облікового запису користувача
```

Рисунок 3.1 – Структура облікового запису Diploma (diploma.rs)

```
    pub template: Pubkey,      // PDA шаблону, за яким видано диплом

    #[max_len(CID_V1_LENGTH)]
    pub background_hash: Option<String>,  // IPFS CIDv1 хеш фону

    pub status: DiplomaStatus, // AwaitingApproval | Approved | Rejected

    pub valid_until: Option<u64>,         // термін дії (Unix timestamp)

    pub country_code: Option<CountryCode>,

    #[max_len(DIPLOMA_METADATA_MAX_LENGTH)]
    pub metadata: Option<String>,         // довільні метадані

    pub issued_at_timestamp:   u64,
    pub reviewed_at_timestamp: Option<u64>,
}
```

Рисунок 3.1, аркуш 2

3.6.2 Інструкція видачі диплому (issue_diploma)

Смарт-контракт містить інструкцію issue_diploma, яку викликає емітент. Перед збереженням даних виконується валідація: перевіряється довжина метаданих. Решта обмежень – статус емітента, шаблону та користувача – перевіряються декларативно у структурі контексту IssueDiploma за допомогою атрибуту #[account(constraint = ...)]. Після перевірок обліковий запис диплому заповнюється даними із пов’язаних записів (рис. 3.2).

```
pub fn issue_diploma(
    ctx: Context<IssueDiploma>,
    metadata: Option<String>,
) -> Result<()> {
    // валідація необов’язкових метаданих
    if let Some(metadata) = metadata.clone() {
        require!(
            validate_string_length(
                &metadata,
```

Рисунок 3.2 – Інструкція issue_diploma (lib.rs)

```
                DIPLOMA_METADATA_MIN_LENGTH,
                DIPLOMA_METADATA_MAX_LENGTH,
            ),
            DiplomaError::InvalidMetadata
        );
    }

    let diploma  = &mut ctx.accounts.diploma;
    let counter  = &mut ctx.accounts.counter;
    let template = &ctx.accounts.template;
    let issuer   = &ctx.accounts.issuer;
    let user     = &ctx.accounts.user;

    let timestamp = Clock::get()?.unix_timestamp as u64;

    **diploma = Diploma {
        name:             template.name.clone(),
        id:               counter.data,
        issuer:           issuer.key(),
        user:             user.key(),
        template:         template.key(),
        background_hash:  template.background_hash.clone(),
        status:           DiplomaStatus::default(), // AwaitingApproval
        valid_until:      template.expiration_time
                              .map(|ttl| timestamp + ttl),
        country_code:     template.country_code.clone(),
        metadata,
        issued_at_timestamp:   timestamp,
        reviewed_at_timestamp: None,
    };

    counter.data += 1; // інкремент глобального лічильника

    Ok(())
}
```

Рисунок 3.2, аркуш 2

Після успішного виклику у блокчейні створюється новий PDA-акаунт з незмінними даними про виданий диплом. Статус диплому встановлюється у

AwaitingApproval – користувач повинен окремо підтвердити або відхилити документ.

3.6.3 Автоматизований тест видачі диплому

Для перевірки коректності інструкції issue_diploma написано інтеграційний тест мовою Rust з використанням бібліотеки anchor-client та tokio. Тест відтворює повний бізнес-цикл: реєстрація та підтвердження емітента, створення та підтвердження шаблону, реєстрація та підтвердження користувача, і нарешті – видача диплому. Після виклику тест зчитує обліковий запис із блокчейну та перевіряє кожне поле (рис. 3.3).

```
#[tokio::test]
#[serial]
async fn should_issue_diploma() {
    let context = initialize().await;
    let admin   = context.admin.clone();
    let user    = create_user().await;

    let client  = Client::new(Cluster::Localnet, user.clone());
    let program = client.program(diploma::ID).unwrap();

    // --- підготовка: емітент ---
    request(&program, create_issuer_accounts(user.pubkey()),
        instruction::CreateIssuer {
            name: VALID_ISSUER_NAME_NAME.to_string(),
            government_country_code: None,
        }, user.clone()).await;

    request(&program,            approve_issuer_accounts(admin.pubkey(),
user.pubkey()),
        instruction::ApproveIssuer {}, admin.clone()).await;

    // --- підготовка: шаблон ---
    let template_counter: DiplomaTemplateCounter =
```

Рисунок 3.3 – Інтеграційний тест should_issue_diploma (tests/src/issue_diploma.rs)

```
        get_account(&program,
create_diploma_template_counter_pda()).await;
    let                          template_pda                          =
create_diploma_template_pda(template_counter.data);

    request(&program,            create_template_accounts(user.pubkey(),
template_counter.data),
        instruction::CreateTemplate {
            name:            "Test diploma".to_string(),
            background_hash: Some(VALID_CID.to_string()),
            expiration_time: Some(DIPLOMA_EXPIRATION_MIN_SECONDS),
        }, user.clone()).await;

    request(&program,          approve_template_accounts(admin.pubkey(),
template_counter.data),
        instruction::ApproveTemplate {}, admin.clone()).await;

    // --- підготовка: користувач ---
    request(&program, create_user_accounts(user.pubkey()),
        instruction::CreateUser {
            first_name:  VALID_USER_FIRST_NAME.to_string(),
            middle_name: None,
            last_name:   VALID_USER_LAST_NAME.to_string(),
            external_id: 0,
        }, user.clone()).await;

    request(&program,       approve_user_account_accounts(admin.pubkey(),
user.pubkey()),
        instruction::ApproveUser {}, admin.clone()).await;

    // --- видача диплому ---
    let diploma_counter: DiplomaCounter =
        get_account(&program, create_diploma_counter_pda()).await;
    let diploma_pda = create_diploma_pda(diploma_counter.data);

    let start_ts = get_timestamp();
    request(&program, accounts::IssueDiploma {
        diploma:        diploma_pda,
        issuer:         create_issuer_account_pda(user.pubkey()),
        template:       template_pda,
        user:           create_user_account_pda(user.pubkey()),
```

Рисунок 3.3, аркуш 2

```
        counter:        create_diploma_counter_pda(),
        authority:      user.pubkey(),
        system_program: system_program::ID,
    }, instruction::IssueDiploma {
        metadata: Some(VALID_DIPLOMA_METADATA.to_string()),
    }, user.clone()).await;
    let end_ts = get_timestamp();

    // --- перевірка полів ---
    let diploma: Diploma = get_account(&program, diploma_pda).await;
    let     template:     DiplomaTemplate     =     get_account(&program,
template_pda).await;

    assert_eq!(diploma.name,            template.name);
    assert_eq!(diploma.issuer,
create_issuer_account_pda(user.pubkey()));
    assert_eq!(diploma.user,
create_user_account_pda(user.pubkey()));
    assert_eq!(diploma.template,        template_pda);
    assert_eq!(diploma.id,              diploma_counter.data);
    assert_eq!(diploma.background_hash, template.background_hash);
    assert_eq!(diploma.status,
DiplomaStatus::AwaitingApproval);
    assert_eq!(diploma.valid_until,
               Some(diploma.issued_at_timestamp                          +
DIPLOMA_EXPIRATION_MIN_SECONDS));
    assert_eq!(diploma.metadata,
               Some(VALID_DIPLOMA_METADATA.to_string()));
    assert!(check_timestamp(start_ts,        diploma.issued_at_timestamp,
end_ts));
}
```

Рисунок 3.3, аркуш 3

Тест є інтеграційним: він взаємодіє з реальним локальним валідатором Solana, а не з імітованим середовищем. Це гарантує, що перевіряється фактична поведінка смарт-контракту в умовах, максимально наближених до production.

3.7 Вибір технологій для зберігання зображень (фонів)

У процесі розробки системи зберігання зображень (фонів) важливим етапом є вибір відповідної технології. Існує кілька основних підходів до зберігання даних: централізовані рішення (бази даних), децентралізовані системи (IPFS), а також зберігання безпосередньо у блокчейні [12].

Кожен із підходів має свої переваги та недоліки, які необхідно враховувати залежно від вимог до системи, таких як вартість, простота реалізації та рівень надійності. Для порівняння основних характеристик було сформовано таблицю.

Таблиця 3.11 – Порівняльний аналіз технологій зберігання зображень

Система	**Ціна**	**Простота**	**Надійність**
IPFS	Низька / середня	Середня (потрібна інтеграція)	Висока (розподілене зберігання)
База даних	Низька	Висока	Найнижча
Блокчейн	Висока	Низька	Найвища

На основі проведеного аналізу можна зробити висновок, що використання класичної бази даних є найбільш простим та економічно вигідним рішенням, проте має обмеження з точки зору надійності та централізації. Зберігання даних безпосередньо у блокчейні забезпечує найвищий рівень надійності та незмінності інформації, однак є дорогим і складним у реалізації, особливо для великих обсягів даних, таких як зображення.

Оптимальним компромісним варіантом є використання IPFS, який поєднує відносно низьку вартість із високою надійністю завдяки децентралізованій природі [12]. Саме тому для реалізації системи доцільно використовувати IPFS у поєднанні з блокчейном, де у блокчейні зберігатиметься лише хеш (посилання) на зображення.

3.8 Тестування програми

Тестування програмного забезпечення є невід'ємною частиною життєвого циклу розробки програмних систем. Його основною метою є забезпечення відповідності розробленого програмного продукту поставленим вимогам, виявлення помилок, дефектів та потенційних вразливостей до моменту введення системи в експлуатацію. Для захищених розподілених платформ, зокрема таких, що використовують блокчейн-технології для управління дипломами та сертифікатами, тестування має особливе значення, оскільки будь-яка помилка може призвести до втрати даних, порушення цілісності інформації або зниження рівня довіри до системи [17].

Тестування дозволяє:

– перевірити коректність реалізації функціональних вимог;

– оцінити надійність, стабільність та масштабованість системи;

– забезпечити інформаційну безпеку та захист від несанкціонованого доступу;

– підвищити якість програмного забезпечення та зменшити витрати на його супровід.

3.8.1 Місце тестування у життєвому циклі розробки ПЗ

Життєвий цикл розробки програмного забезпечення включає етапи аналізу вимог, проєктування, реалізації, тестування, впровадження та супроводу. Тестування не є окремим фінальним етапом, а повинно проводитися протягом усього циклу розробки.

Для даної платформи доцільно використовувати ітеративну або інкрементну модель розробки, у якій тестування виконується після реалізації кожного функціонального модуля. Це дозволяє на ранніх етапах виявляти дефекти та оперативно їх усувати [16].

Особливо важливим є раннє тестування таких компонентів:

– модулів автентифікації та авторизації користувачів;
– механізмів взаємодії з блокчейном;
– API для роботи з дипломами та сертифікатами.

3.8.2 Тестування блокчейн частини

Блокчейн-частина системи відповідає за зберігання даних дипломів і користувачів та забезпечення незмінності і прозорості даних. Її тестування має специфічні особливості, пов’язані з розподіленою природою технології.

Смарт-контракти є ключовим елементом блокчейн-платформи. Їх тестування включає:

– перевірку коректності бізнес-логіки;
– тестування граничних випадків;
– перевірку стійкості до зловмисних дій.

У процесі розробки блокчейн-частини системи було реалізовано набір тестових випадків, спрямованих на перевірку коректності роботи смарт-контрактів (програми), а також забезпечення надійності та безпеки виконання основних операцій.

Тестування охоплює ключові бізнес-процеси системи, зокрема створення сутностей, їх підтвердження або відхилення, а також безпосередню видачу дипломів. Для цього було реалізовано окремі тестові сценарії, кожен з яких відповідає певній функціональній можливості програми.

Було реалізовано наступні тестові випадки:

– **ініціалізація системи (initialize)** – Перевіряється коректність початкового налаштування програми, створення адміністративних облікових записів та базових параметрів;

– **створення користувача (create_user_account)** – Тестується можливість створення нового облікового запису користувача з відповідними правами доступу;

– **підтвердження користувача (approve_user_account)** – Перевіряється логіка підтвердження користувача уповноваженим суб’єктом;

– **відхилення користувача (reject_user_account)** – Тестується сценарій відмови у реєстрації користувача;

– **створення емітента (create_issuer)** – Перевіряється можливість створення організації (емітента), що має право видавати дипломи;

– **підтвердження емітента (approve_issuer)** – Тестується процес надання прав емітенту після перевірки;

– **відхилення емітента (reject_issuer)** – Перевіряється сценарій відмови у реєстрації емітента;

– **створення шаблону диплома (create_template)** – Перевіряється створення шаблону диплома, який використовується при генерації документів;

– **підтвердження шаблону (approve_template)** – Тестується логіка підтвердження шаблону перед його використанням;

– **відхилення шаблону (reject_template)** – Перевіряється можливість відхилення некоректного або невалідного шаблону;

– **видача диплома (issue_diploma)** – Один із ключових тестів, що перевіряє процес створення диплома, запису його у блокчейн та прив’язки до користувача;

– **підтвердження диплома (approve_diploma)** – Перевіряється механізм валідації диплома після його створення;

– **відхилення диплома (reject_diploma)** – Тестується сценарій скасування або відхилення диплома.

4 АНАЛІЗ РОЗРОБЛЕНОЇ ПЛАТФОРМИ

4.1 Проведення автоматичного тестування

Автоматичне тестування є найпростішим способом перевірити коректну роботу додатку перед його запуском. Воно не вимагає витрачання часу на ручні перевірки та окремої людини (інженера з якості) для цього.

Фреймворк Anchor має вбудований скрипт для запуску автоматичних тестів, він включає в себе запуск локального валідатора та самих тестів. Це команда “anchor test”.

Після завершення виконання тестів можна побачити загальну кількість проведених тестів, кількість успішних та провалених тестів, час виконання команди та розподіл тестів за типами (рис. 4.1).

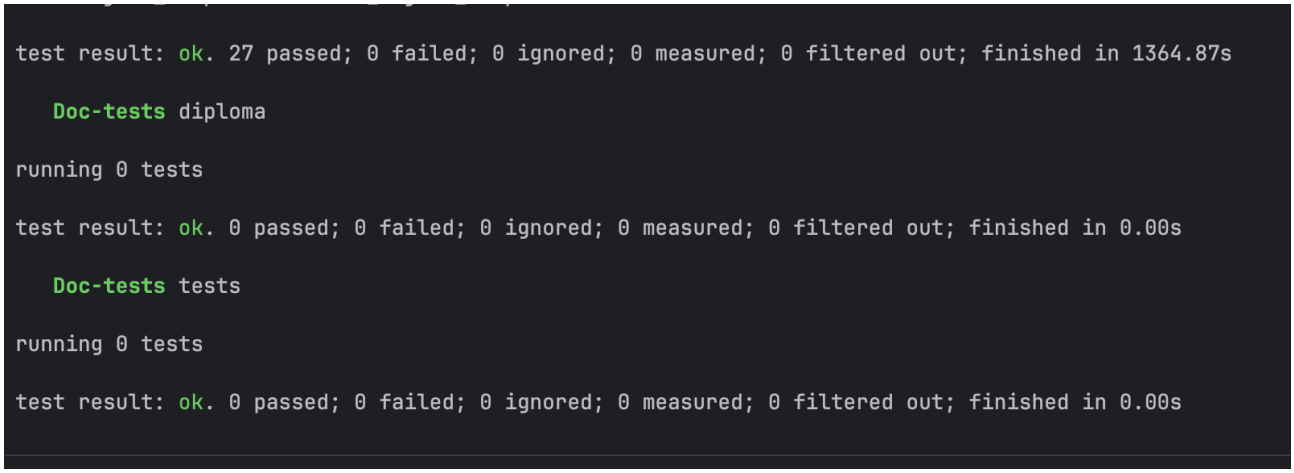


Рисунок 4.1 – Результати виконання автоматичних тестів

4.2 Запуск додатку

4.2.1 Запуск локального валідатора

Для роботи смарт-контракту у локальному середовищі запускається вбудований валідатор Solana за допомогою команди solana-test-validator. Рисунок 4.2 демонструє успішний старт валідатора: відображається адреса RPC-сервера, ідентифікатор мережі та початок генерації блоків.

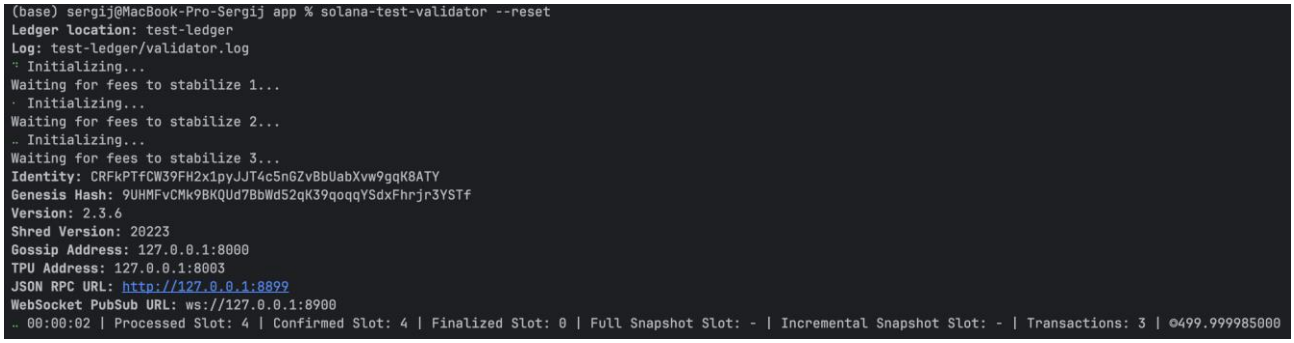


Рисунок 4.2 – Запуск локального валідатора Solana

4.2.2 Компіляція та розгортання програми

Після запуску валідатора смарт-контракт компілюється та розгортається командою anchor deploy. На рисунку 4.3 видно успішне завершення компіляції та підтвердження розгортання із зазначенням Program ID контракту у мережі.

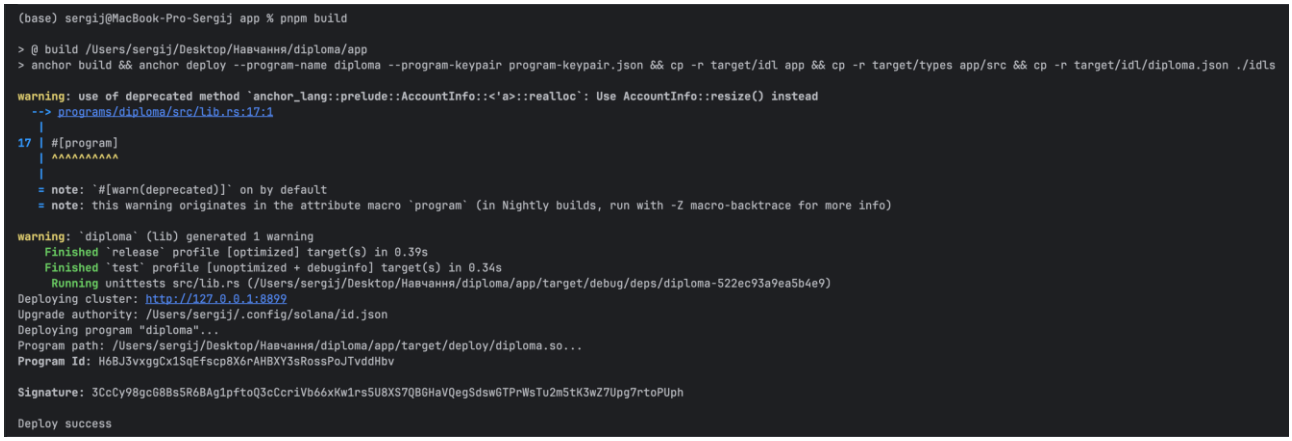


Рисунок 4.3 – Компіляція та розгортання смарт-контракту

4.3 Запуск скрипта на створення акаунту адміністратора

Після розгортання контракту виконується скрипт ініціалізації, що створює обліковий запис адміністратора та виводить в консоль приватний його ключ, що є не зовсім правильним, але на поточному етапі розвитку продукту – прийнятно. Рисунок 4.4 показує результат виконання скрипта з підтвердженням успішного запису транзакції у блокчейн.



Рисунок 4.4 – Ініціалізація облікового запису адміністратора

4.4 Запуск веб-додатку

Фронтенд-додаток запускається командою pnpm dev, що стартує сервер розробки Vite. Рисунок 4.5 підтверджує успішний запуск: додаток доступний за локальною адресою, виводиться час компіляції та адреса для підключення.

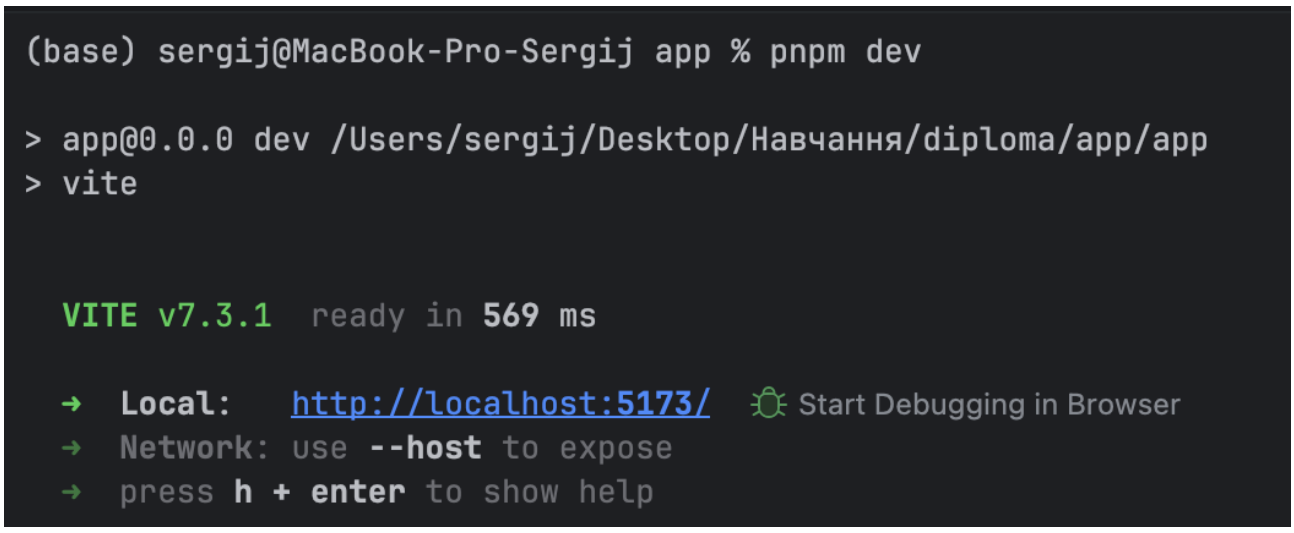


Рисунок 4.5 – Запуск веб-додатку

4.5 Авторизація адміністратором

Взаємодія з платформою здійснюється через крипто-гаманець Solana (в даному випадку Phantom wallet). На рисунку 4.6 показано процес підключення гаманця адміністратора до веб-інтерфейсу: система розпізнає обліковий запис зчитуючи приватний ключ адміністратора та відображає відповідний функціонал керування платформою.

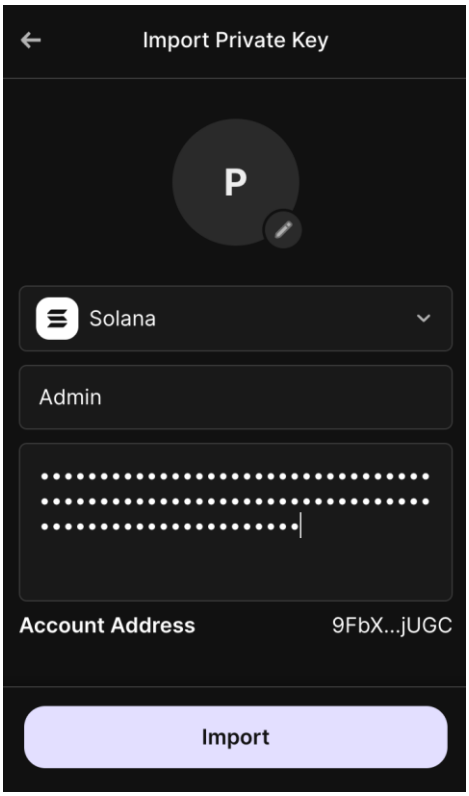


Рисунок 4.6 – Авторизація з правами адміністратора

4.6 Використання платформи емітентом

4.6.1 Реєстрація акаунту

Емітент реєструється у системі, заповнивши форму з назвою організації та країною (у випадку коли він прив’язаний до конкретної країни). Рисунок 4.7 демонструє інтерфейс реєстрації та відображення щойно створеного облікового

запису зі статусом AwaitingApproval, що означає очікування підтвердження адміністратором.

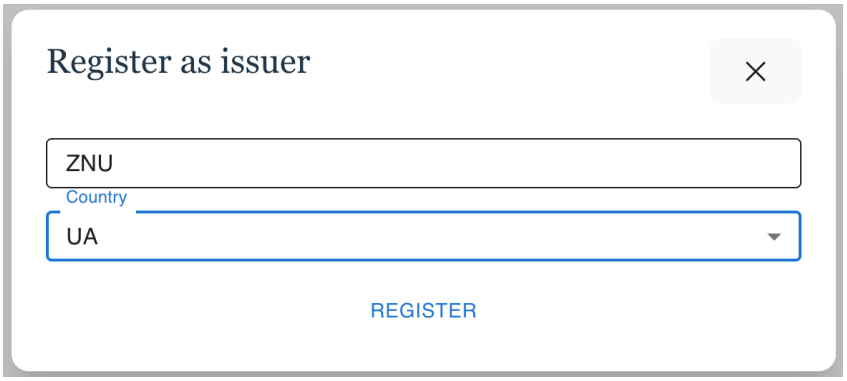


Рисунок 4.7 – Реєстрація облікового запису емітента

4.6.2 Створення шаблону диплому

Після підтвердження емітент може створювати шаблони дипломів. На рисунку 4.8 показано форму створення шаблону: назва, завантаження необов'язкового зображення фону диплому (хеш зберігається в IPFS), необов'язковий термін дії. Щойно створений шаблон отримує статус AwaitingApproval.

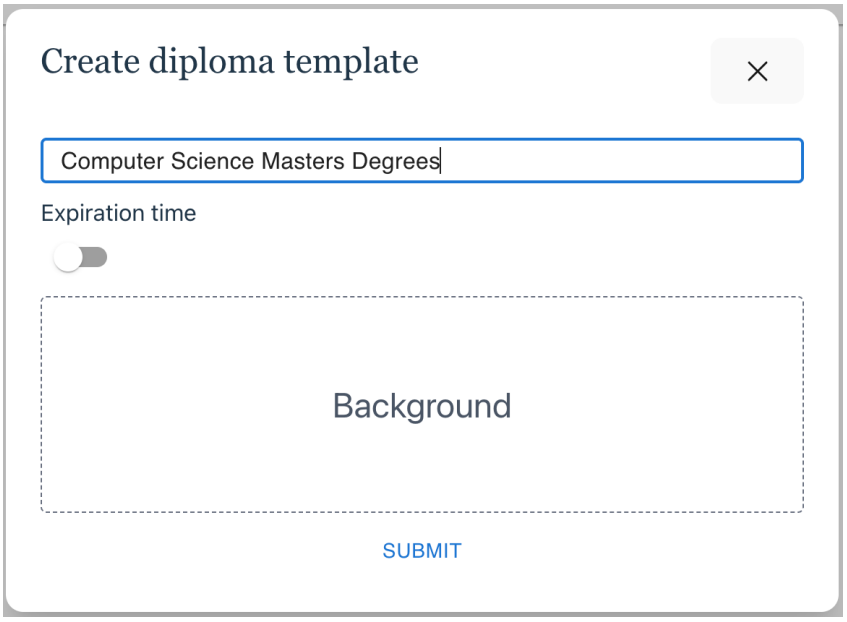


Рисунок 4.8 – Створення шаблону диплому

4.6.3 Видача диплому

На рисунку 4.9 показано інтерфейс видачі диплому після затвердження шаблону. Емітент обирає затверджений шаблон і підтвердженого користувача, опціонально додає метадані. Після підтвердження транзакції диплом з'являється у системі зі статусом AwaitingApproval.

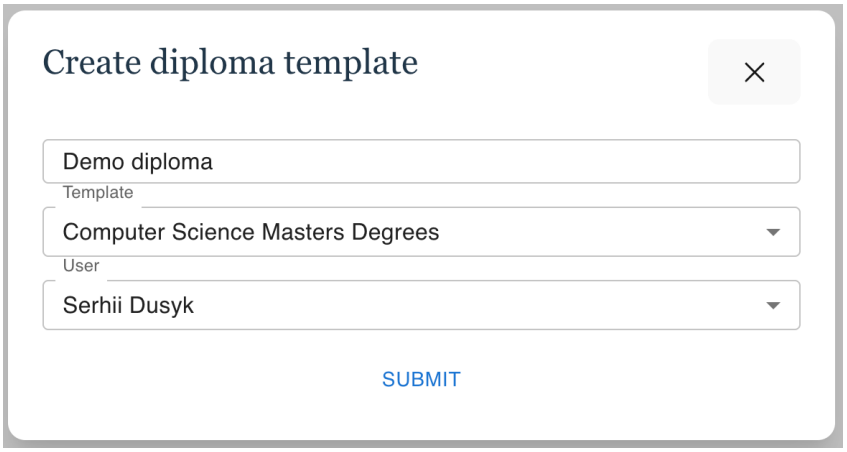


Рисунок 4.9 – Видача диплому

4.7 Функціонал адміністратора

4.7.1 Підтвердження акаунту емітента

Адміністратор переглядає список емітентів, що очікують підтвердження. Рисунок 4.10 демонструє відповідний розділ адміністративного інтерфейсу та результат підтвердження: статус емітента змінюється на Approved.



Рисунок 4.10 – Підтвердження облікового запису емітента адміністратором

4.7.2 Підтвердження шаблону

Аналогічно до підтвердження емітента, адміністратор розглядає заявки на шаблони дипломів. На рисунку 4.11 показано форму перегляду шаблону та зміну його статусу на Approved після підтвердження.



Рисунок 4.11 – Підтвердження шаблону диплому адміністратором

4.7.3 Підтвердження акаунту студента

Рисунок 4.12 ілюструє процес підтвердження облікового запису студента. Адміністратор переглядає дані користувача та підтверджує його реєстрацію, після чого студент отримує можливість приймати дипломи.

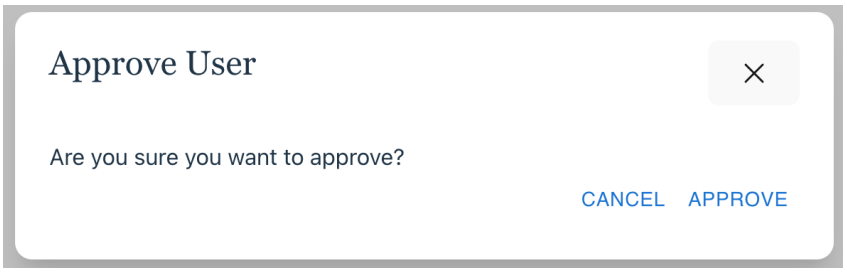


Рисунок 4.12 – Підтвердження облікового запису студента адміністратором

4.8 Функціонал користувача

4.8.1 Створення акаунту

Рисунок 4.13 ілюструє процес створення облікового запису студента. Студент має змогу ввести свої дані: ім’я, опційне ім’я по батькові, прізвище та унікальний ідентифікатор (наприклад ІПН або номер паспорту). Після створення акаунту він буде у статусі AwaitingApproval до підтвердження адміністратором.

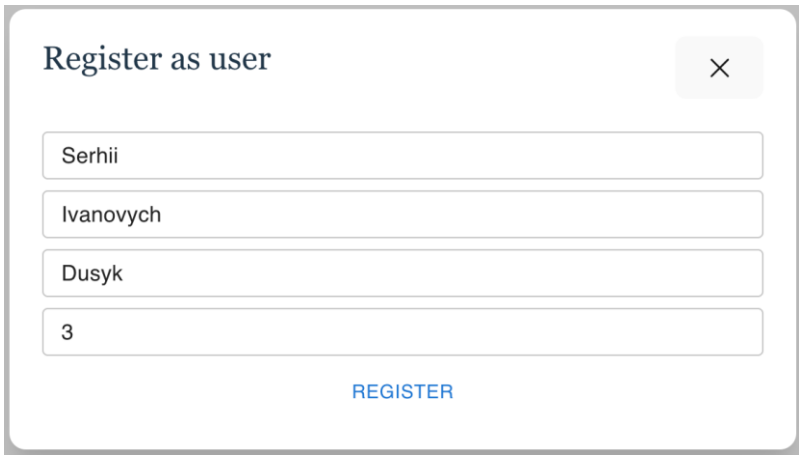


Рисунок 4.13 – Створення облікового запису користувача

4.8.2 Підтвердження диплому

Після видачі диплому студент повинен його підтвердити або відхилити. Рисунок 4.14 показує відповідний інтерфейс: студент бачить деталі виданого диплому та може підтвердити отримання, змінивши статус на Approved.

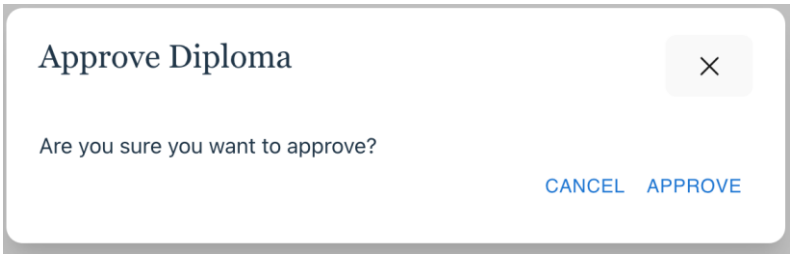


Рисунок 4.14 – Підтвердження диплому користувачем

4.9 Перевірка освітнього документу за ідентифікатором

Будь-який користувач може перевірити автентичність диплому за його числовим ідентифікатором без авторизації. Рисунок 4.15 ілюструє відповідну форму пошуку та результат: система відображає дані диплому безпосередньо з блокчейну, що гарантує їх незмінність та достовірність.

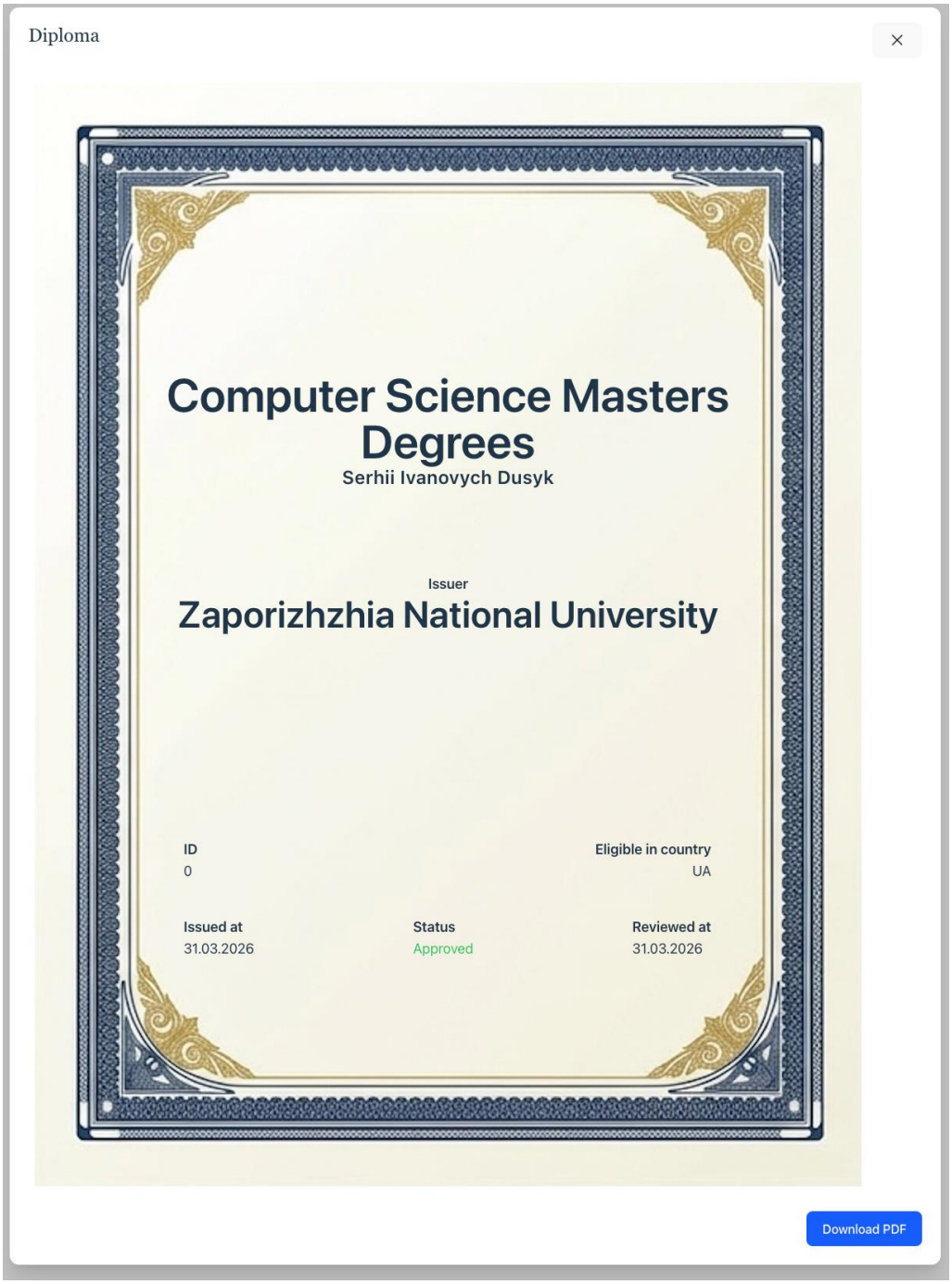


Рисунок 4.15 – Перевірка освітнього документа за ідентифікатором

5 ПОДАЛЬШИЙ РОЗВИТОК ПРОДУКТУ

Розроблена платформа управління дипломами та сертифікатами на основі блокчейн-технологій демонструє можливість створення прозорої, надійної та захищеної системи зберігання освітніх документів. Використання розподіленого реєстру дозволяє забезпечити незмінність даних, спростити процес перевірки документів та усунути проблему централізованої точки відмови.

Разом з тим, створена система є прототипом, який реалізує базовий функціонал управління дипломами та сертифікатами. Подальший розвиток платформи може бути спрямований на розширення її можливостей, підвищення зручності використання та інтеграцію з існуючими освітніми та державними інформаційними системами.

5.1 Розширення функціоналу управління емітентами

Одним із напрямків розвитку платформи є вдосконалення механізмів управління емітентами дипломів та сертифікатів. У поточній реалізації емітенти проходять процедуру реєстрації та підтвердження адміністратором системи, після чого отримують можливість створювати шаблони документів та видавати дипломи користувачам.

У подальшому доцільно реалізувати механізм деактивації або блокування емітентів. Така функція може знадобитися у випадках:

- припинення діяльності освітньої установи;
- виявлення порушень правил використання платформи;
- компрометації облікового запису емітента;
- помилкової реєстрації організації.

Деактивація емітента повинна здійснюватися таким чином, щоб уже видані дипломи залишалися доступними для перевірки та зберігали свою юридичну силу. При цьому емітент втрачає можливість створювати нові документи або редагувати існуючі шаблони.

5.2 Підтримка багатомовності

Поточна реалізація платформи підтримує лише англійську мову інтерфейсу. Водночас потенційна аудиторія системи є міжнародною – навчальні заклади різних країн та їхні студенти. Підтримка багатомовності дозволить значно розширити охоплення платформи та підвищити зручність її використання.

Реалізацію багатомовності доцільно здійснити на рівні фронтенд-додатку за допомогою стандартних бібліотек локалізації, зокрема react-i18next. Усі рядки інтерфейсу виносяться до окремих файлів перекладу (наприклад, uk.json, en.json), що спрощує додавання нових мов без зміни коду компонентів. На першому етапі доцільно додати підтримку української мови як основної для вітчизняних навчальних закладів.

Дані самих дипломів (імена, назви закладів) зберігаються у блокчейні як введені користувачем рядки без прив’язки до мови. Таким чином, локалізація торкається виключно інтерфейсу, а не структури смарт-контракту.

5.3 Покращення користувацького інтерфейсу

Іншим важливим напрямком розвитку системи є вдосконалення веб-інтерфейсу платформи. Зручність використання програмного забезпечення є одним із ключових факторів його успішного впровадження.

Подальше покращення інтерфейсу може включати:

– оптимізацію структури навігації;
– покращення візуального оформлення сторінок;
– спрощення процесу створення та перегляду дипломів;
– додавання інтерактивних підказок та інформаційних повідомлень.

Особливу увагу слід приділити інтерфейсу перевірки дипломів, оскільки ним можуть користуватися роботодавці, навчальні установи або інші організації, які не мають спеціальних технічних знань. Процес перевірки повинен бути максимально простим та зрозумілим.

5.4 Підтримка світлої та темної теми інтерфейсу

Сучасні веб-додатки часто підтримують декілька режимів відображення інтерфейсу, зокрема світлу та темну тему. Реалізація такої функції дозволяє підвищити комфорт використання системи та адаптувати її до різних умов роботи.

Темна тема може бути особливо корисною при роботі у вечірній час або в умовах низького освітлення. Крім того, використання темного режиму може зменшувати навантаження на зір користувача при тривалій роботі з системою.

Реалізація перемикання тем може здійснюватися на рівні фронтенд-додатку із використанням сучасних CSS-технологій та механізмів збереження налаштувань користувача.

5.5 Розширення тестового покриття системи

Важливим напрямком подальшого розвитку є розширення автоматизованого тестування програмного забезпечення. У поточній реалізації основна увага приділена тестуванню блокчейн-частини системи, зокрема смарт-контрактів.

Для підвищення якості програмного продукту доцільно:

- збільшити покриття тестами веб-клієнта;
- додати інтеграційні тести для перевірки взаємодії між фронтендом та блокчейном;
- впровадити автоматичне тестування інтерфейсу користувача;
- використовувати інструменти безперервної інтеграції (CI).

Автоматизоване тестування дозволить швидше виявляти помилки під час подальшої розробки системи та зменшить ризик появи критичних дефектів у робочій версії платформи.

5.6 Оптимізація структури облікових записів

Ще одним напрямком удосконалення системи є оптимізація структури зберігання даних у блокчейні. У поточній реалізації для зберігання інформації використовуються програмно визначені адреси (Program Derived Address, PDA), які дозволяють створювати детерміновані акаунти для різних типів даних.

Подальша оптимізація структури PDA може дозволити:

- спростити пошук інформації у блокчейні;
- зменшити обсяг необхідних обчислень при обробці транзакцій;
- покращити масштабованість системи;
- підвищити ефективність роботи з великим обсягом дипломів.

Правильно спроектована структура даних є важливою умовою ефективної роботи децентралізованих додатків, особливо у випадку масового використання системи.

5.7 Розробка мобільного додатку

Ще одним перспективним напрямком розвитку платформи є створення мобільного додатку. Більшість користувачів сучасних інформаційних систем активно використовують мобільні пристрої, тому наявність мобільного клієнта може значно підвищити доступність платформи.

Мобільний додаток може забезпечити такі можливості:

- перегляд власних дипломів та сертифікатів;
- отримання сповіщень про нові документи;
- швидку перевірку дипломів за допомогою QR-коду;
- взаємодію з блокчейном через криптографічний гаманець.

Використання мобільних технологій дозволить значно спростити процес доступу до освітніх документів та підвищить зручність використання системи.

5.8 Інтеграція з державними інформаційними системами

Одним із найбільш перспективних напрямків розвитку платформи є інтеграція з державними реєстрами освітніх документів. Така інтеграція дозволить створити єдину цифрову екосистему для управління дипломами та сертифікатами [2].

Можливі варіанти інтеграції включають:

- автоматичну синхронізацію даних між блокчейн-платформою та державними реєстрами;
- використання блокчейну як додаткового рівня перевірки документів;
- обмін даними між освітніми установами та державними органами.

Використання блокчейн-технологій у державних інформаційних системах може підвищити рівень прозорості, зменшити ризик підробки документів та спростити міжнародне визнання дипломів.

5.9 Загальні перспективи розвитку

Розроблена платформа демонструє потенціал використання блокчейн-технологій у сфері управління освітніми документами. Подальший розвиток системи може перетворити її на універсальну платформу для зберігання та перевірки різних типів сертифікатів, дипломів та інших документів.

Інтеграція з освітніми установами, державними системами та міжнародними організаціями може створити єдину глобальну інфраструктуру перевірки освітніх досягнень. Такий підхід сприятиме підвищенню довіри до цифрових документів, спростить процес міжнародного працевлаштування та забезпечить довгострокове зберігання інформації про освіту користувачів.

Таким чином, розроблена платформа має значний потенціал для подальшого розвитку та може стати основою для створення сучасної цифрової екосистеми управління освітніми документами.

ВИСНОВКИ

У результаті виконання роботи поставлена задача з розробки захищеної децентралізованої платформи була успішно виконана. Створена система реалізує основні механізми видачі та перевірки освітніх документів на основі блокчейн-технологій, підтверджує технічну доцільність обраного підходу та відповідає меті кваліфікаційної роботи – розробці програмного забезпечення захищеної розподіленої платформи для управління дипломами та сертифікатами. Реалізована система на рівні Proof of Concept демонструє повний функціональний цикл роботи платформи та підтверджує можливість подальшого розвитку до рівня виробничого рішення.

У ході виконання кваліфікаційної роботи було досліджено існуючі підходи до видачі та перевірки дипломів і сертифікатів, а також проаналізовано їхні основні переваги та недоліки. Проведений аналіз показав, що більшість сучасних систем базуються на централізованій архітектурі, яка має суттєві вразливості: єдину точку відмови, залежність від одного оператора інфраструктури та складність автоматизованої міжнародної верифікації документів.

З метою вирішення зазначених проблем у роботі було поставлено задачу розробки захищеної децентралізованої платформи для управління дипломами та сертифікатами із використанням блокчейн-технологій. Для реалізації системи було обрано блокчейн Solana та фреймворк Anchor.

Вибір Solana обґрунтовано результатами порівняльного аналізу провідних блокчейн-платформ. Solana забезпечує найнижчу комісію за транзакцію (менше 0,001 USD) та найвищу пропускну здатність (до 65 000 транзакцій на секунду) серед розглянутих рішень, що безпосередньо впливає на операційну вартість та швидкодію платформи. Фреймворк Anchor обрано завдяки декларативному опису облікових записів, вбудованій валідації обмежень та зручному TypeScript-клієнту для взаємодії з програмою.

У рамках роботи було спроєктовано архітектуру системи, що включає блокчейн-частину з реалізацією бізнес-логіки у вигляді смарт-контрактів, механізми зберігання метаданих дипломів, а також клієнтський веб-інтерфейс для взаємодії користувачів із платформою.

Смарт-контракт платформи реалізовано мовою Rust і визначає сім типів облікових записів, що зберігаються безпосередньо у блокчейні: адміністратор, емітент, користувач, шаблон диплому, лічильник шаблонів, диплом та лічильник дипломів. Адресація облікових записів побудована на основі Program Derived Address (PDA), що забезпечує детерміновану та відтворювану адресацію без зберігання покажчиків. Смарт-контракт реалізує чотирнадцять інструкцій, що охоплюють повний бізнес-цикл: реєстрацію та підтвердження емітентів, шаблонів і користувачів, видачу дипломів та їх підтвердження або відхилення отримувачем.

Особливу увагу в роботі було приділено забезпеченню надійності та коректності функціонування системи. Розроблена блокчейн-частина платформи покрита автоматизованими тестами, що дозволяють перевірити правильність реалізації основних бізнес-сценаріїв та стійкість до некоректних вхідних даних.

Тести реалізовано мовою Rust у вигляді інтеграційних тестів, що взаємодіють із локальним валідатором Solana через бібліотеку anchor-client. Такий підхід забезпечує перевірку фактичної поведінки смарт-контракту в умовах, максимально наближених до production, без використання заглушок або імітованого середовища. Тестове покриття охоплює тринадцять сценаріїв – ініціалізацію системи, повний цикл реєстрації кожної сутності та видачу й підтвердження дипломів.

Для взаємодії користувачів із системою було реалізовано простий веб-інтерфейс, який забезпечує доступ до основних функцій платформи. Інтерфейс дозволяє виконувати операції створення, перегляду та перевірки дипломів і сертифікатів, а також здійснювати адміністративне управління обліковими записами.

Отримані результати підтверджують перспективність використання блокчейн-технологій у сфері зберігання та перевірки освітніх документів [8]. Подальший розвиток платформи може бути спрямований на розширення функціоналу, покращення інтерфейсу, інтеграцію з державними інформаційними системами та представлення дипломів у форматі NFT.

ПЕРЕЛІК ПОСИЛАНЬ

1. ДСТУ ISO/IEC 25010:2016. Системна та програмна інженерія. Вимоги до якості систем і програмного забезпечення та їхнє оцінювання (SQuaRE). Моделі якості систем і програмних продуктів. Київ : ДП «УкрНДНЦ», 2017.

2. Опірський І. Р., Баласька В. П., Побережник В. М. Сучасні можливості використання технології блокчейн у системі освіти. *Український науковий журнал інформаційної безпеки*. 2023. Т. 29, № 3. С. 138–146.

3. Про вищу освіту: Закон України від 01.07.2014 № 1556-VII. URL : https://zakon.rada.gov.ua/laws/show/1556-18 (дата звернення : 28.02.2026).

4. Про електронні довірчі послуги: Закон України від 05.10.2017 № 2155-VIII. URL : https://zakon.rada.gov.ua/laws/show/2155-19 (дата звернення : 12.01.2026).

5. Про освіту: Закон України від 05.09.2017 № 2145-VIII. URL : https://zakon.rada.gov.ua/laws/show/2145-19 (дата звернення : 25.02.2026).

6. Про Порядок ведення Єдиної державної електронної бази з питань освіти: Постанова Кабінету Міністрів України від 13.07.2011 № 751. URL : https://zakon.rada.gov.ua/laws/show/872-2025-%D0%BF (дата звернення : 25.04.2026).

7. Таненбаум Е., ван Стін М. Розподілені системи: принципи та парадигми. Київ : Видавнича група BHV, 2003. 877 с.

8. Шваб К. Четверта промислова революція. Київ : Форс Україна, 2019. 232 с.

9. Alammary A., Alhazmi S., Alshammari M., Al-Raddaoui S. Blockchain Technology in Higher Education: A Systematic Review. *IEEE Access*. 2019. Vol. 7. P. 147097–147117.

10. Anchor Framework Documentation [Electronic resource]. URL : https://www.anchor-lang.com/ (дата звернення : 15.03.2026).

11. Andreas M. Antonopoulos, David A. Harding. Mastering Bitcoin: Programming the Open Blockchain. 2nd ed. O'Reilly Media, 2017. 416 p.

12. Benet J. IPFS - Content Addressed, Versioned, P2P File System. *Protocol Labs*, 2014. URL : https://arxiv.org/abs/1407.3561 (дата звернення : 25.04.2026).

13. Chen G., Xu B., Lu M., Chen N. S. Exploring blockchain technology and its potential applications for education. *Smart Learning Environments*. 2018. Vol. 5, No 1. URL : https://doi.org/10.1186/s40561-017-0050-x (дата звернення : 25.04.2026).

14. Grech A., Camilleri A. F. Blockchain in Education. *JRC Science for Policy Report*. Luxembourg: Publications Office of the European Union, 2017. 132 p.

15. Klabnik S., Nichols C. The Rust Programming Language. San Francisco : No Starch Press, 2018. 560 p.

16. Martin R. C. Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall, 2008. 464 p.

17. Myers G. J., Sandler C., Badgett T. The Art of Software Testing. 3rd ed. Wiley, 2011. 256 p.

18. Nakamoto S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. URL : https://bitcoin.org/bitcoin.pdf (дата звернення : 05.04.2026).

19. React Documentation. URL : https://react.dev/ (дата звернення : 25.02.2026).

20. Solana Docs: Accounts. URL : https://docs.solana.com/developing/ programming-model/accounts (дата звернення : 11.04.2026).

21. Swan M. Blockchain: Blueprint for a New Economy. O'Reilly Media, 2015. 152 p.

22. TypeScript Documentation. URL : https://www.typescriptlang.org/docs/ (дата звернення : 20.03.2026).

23. Vite Documentation. URL : https://vite.dev/ (дата звернення : 26.04.2026).

24. Wood G. Ethereum: A Secure Decentralised Generalised Transaction Ledger. 2014. URL : https://ethereum.github.io/yellowpaper/paper.pdf (дата звернення : 25.04.2026).

25. Yakovenko A. Solana: A new architecture for a high performance blockchain v0.8.13. Solana Labs, 2017. URL : https://solana.com/solana-whitepaper.pdf (дата звернення : 25.04.2026).