**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ**
**Кафедра комп'ютерних наук**

# КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА СИСТЕМИ ОБЛІКУ ТА АВТОМАТИЗАЦІЇ ВИДАЧІ ДОВІДОК СТУДЕНТАМ ІЗ QR-ВЕРИФІКАЦІЄЮ ТА ПЕРЕВІРКОЮ СПРАВЖНОСТІ ОНЛАЙН»**

Виконав: студент 4 курсу, групи 6.1222
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми Комп'ютерні науки
(назва освітньої програми)
Д. О. Кісіль
(ініціали та прізвище)

Керівник доцент кафедри комп'ютерних наук, PhD Лебедєва-Дичко А.С.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри фундаментальної та прикладної математики, професор, д.т.н. Гребенюк С.М.
(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 бакалавр

Спеціальність 122 Комп'ютерні науки
(шифр і назва)

Освітня програма Комп'ютерні науки

**ЗАТВЕРДЖУЮ**
В.о. завідувача кафедри комп'ютерних наук, к.пед.н., доцент

_____ Пшенична О.С.
(підпис)

" 08 " грудня 2025 р.

# З А В Д А Н Н Я

## НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Кісіль Данилу Олеговичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Розробка системи обліку та автоматизації видачі довідок студентам із QR-верифікацією та перевіркою справжності онлайн

керівник роботи Лебедєва-Дичко Анастасія Сергіївна, доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 02 » грудня 2025 року № 1741-с

2. Строк подання студентом роботи 04.06.2026

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Теоретичні основи розробки систем електронного документообігу та верифікації даних студентів.
2. Проєктування системи обліку та автоматизації видачі студентських довідок.
3. Розробка та тестування системи автоматизації видачі довідок студентам.

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 05.12.2025

## 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. | 17.02.2026 | |
| | | | |
| 2. | Збір вихідних даних. | 20.03.2026 | |
| | | | |
| 3. | Обробка методичних та теоретичних | 29.03.2026 | |
| | джерел. | | |
| | | | |
| 4. | Розробка першого та другого розділу. | 19.04.2026 | |
| | | | |
| 5. | Розробка третього розділу. | 20.05.2026 | |
| | | | |
| 6. | Оформлення та нормоконтроль | 04.06.2026 | |
| | кваліфікаційної роботи бакалавра. | | |
| | | | |
| 7. | Захист кваліфікаційної роботи. | 19.06.2026 | |

Студент _____ Д. О. Кісіль
(підпис) (ініціали та прізвище)

Керівник роботи _____ А. С. Лебедєва-Дичко
(підпис) (ініціали та прізвище)

**Нормоконтроль пройдено**

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

# РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка системи обліку та автоматизації видачі довідок студентам із QR-верифікацією та перевіркою справжності онлайн»: 86 с., 17 рис., 8 табл., 24 джерел, 2 додатки.

АВТОМАТИЗАЦІЯ, БАЗА ДАНИХ, ВЕБЗАСТОСУНОК, ЕЛЕКТРОННИЙ ДОКУМЕНТООБІГ, ОНЛАЙН-ПЕРЕВІРКА, СТУДЕНТСЬКІ ДОВІДКИ, ЦИФРОВИЙ ПІДПИС, QR-ВЕРИФІКАЦІЯ.

Об’єктом дослідження є процеси обліку студентів та видачі довідок у закладах вищої освіти.

Предметом дослідження є методи та програмні засоби автоматизації видачі студентських довідок із використанням QR-верифікації та онлайн перевірки справжності документів.

Метою роботи є розробка інформаційної системи обліку та автоматизації видачі студентських довідок із впровадженням механізму QR-верифікації та онлайн перевірки їхньої справжності.

Методи дослідження. Використано аналіз літератури та ІС, системний аналіз, проєктування БД, об’єктно-орієнтоване програмування, тестування ПЗ та методи порівняння ефективності рішень.

У роботі проаналізовано сучасні підходи до електронного документообігу у закладах освіти та систем автоматизації видачі довідок. Спроєктовано архітектуру системи та структуру бази даних для обліку студентів, заявок і виданих документів. Реалізовано основні модулі системи, зокрема авторизацію користувачів, подання та обробку заявок, формування електронних довідок із унікальними ідентифікаторами, цифровим підписом та QR-кодом. Реалізовано механізм онлайн-верифікації документів для перевірки їх автентичності в реальному часі. Тестування підтвердило коректність, стабільність роботи системи та ефективність реалізованих рішень.

# SUMMARY

Bachelor’s Qualifying Thesis «Development of a System for Student Certificate Management and Automated Issuance with QR Verification and Online Authenticity Checking»: 86 pages, 17 figures, 8 tables, 24 sources, 2 supplements.

AUTOMATION, DATABASE, WEB APPLICATION, ELECTRONIC DOCUMENT MANAGEMENT, ONLINE VERIFICATION, STUDENT CERTIFICATES, DIGITAL SIGNATURE, QR VERIFICATION.

The object of the study is the processes of student record management and certificate issuance in higher education institutions.

The subject of the study is methods and software tools for automating the issuance of student certificates using QR verification and online document authenticity checking.

The aim of the work is to develop an information system for accounting and automation of student certificate issuance with the implementation of QR verification and online authenticity checking mechanisms.

Research methods include analysis of scientific literature and information systems, systems analysis, database design methods, object-oriented programming, software testing, and comparative evaluation of solution effectiveness.

The work analyzes modern approaches to electronic document management in educational institutions and systems for automating certificate issuance. The system architecture and database structure for managing students, requests, and issued documents were designed. The main system modules were implemented, including user authentication, request processing, generation of electronic certificates with unique identifiers, digital signatures, and QR codes. An online verification mechanism was implemented to ensure real-time document authenticity checking. Testing confirmed the correctness, stability, and efficiency of the implemented solution.

**ЗМІСТ**

Завдання на кваліфікаційну роботу……..2
Реферат.....4
Summary.....5
Вступ.....8
1 Теоретичні основи розробки систем електронного документообігу та верифікації даних студентів.....11
1.1 Поняття та особливості систем електронного документообігу у закладах освіти.....11
1.2 Аналіз сучасних систем автоматизації видачі довідок та цифрової верифікації документів.....14
1.3 Вимоги до системи обліку та автоматизації видачі студентських довідок із QR-верифікацією.....19
2 Проєктування системи обліку та автоматизації видачі студентських довідок.....24
2.1 Аналіз предметної області та постановка задачі системи.....24
2.2 Проєктування архітектури системи та бази даних.....27
2.3 Проєктування механізму QR-верифікації та онлайн перевірки справжності документів.....33
2.4 Проєктування користувацьких інтерфейсів та вибір технологій реалізації.....35
3 Розробка та тестування системи автоматизації видачі довідок студентам...40
3.1 Реалізація програмного забезпечення системи.....40
3.2 Розробка функціоналу формування, зберігання та видачі електронних довідок….....44
3.3 Реалізація QR-верифікації та онлайн перевірки автентичності документів.....48
3.4 Тестування системи та аналіз результатів роботи.....51

3.5 Інструкція користувача.....53
Висновки.....55
Перелік посилань.....57
Додаток А Лістинг функціоналу формування, зберігання та видачі електронних довідок.....60
Додаток Б Інтерфейс прозробленого вебзастосунку.....82

# ВСТУП

У сучасних закладах вищої освіти значно зростає обсяг адміністративних процесів, пов’язаних з обліком студентів та наданням їм різноманітних довідок. Традиційні підходи до оформлення довідок, які передбачають ручну обробку запитів, паперовий документообіг та особисте звернення до адміністративних підрозділів, є малоефективними в умовах цифрової трансформації освіти. Вони призводять до значних витрат часу, перевантаження працівників деканатів, а також підвищують ризики виникнення помилок або підробки документів.

Однією з ключових проблем сучасного освітнього документообігу є недостатній рівень автоматизації процесів видачі та перевірки студентських довідок. У багатьох закладах освіти довідки досі формуються у напівавтоматичному режимі, а їх перевірка здійснюється вручну, що створює можливості для фальсифікацій. Саме тому актуальним є впровадження цифрових систем, які дозволяють не лише автоматизувати процес формування документів, але й забезпечити їхню автентифікацію та швидку перевірку через сучасні технології, зокрема QR-коди та онлайн-сервіси верифікації.

Актуальність теми кваліфікаційної роботи зумовлена необхідністю підвищення ефективності управління студентськими даними, скорочення часу обробки запитів на довідки, а також забезпечення надійного механізму захисту від підробки документів. Використання інформаційних систем із підтримкою QR-верифікації дозволяє значно підвищити рівень довіри до електронних документів і спростити процес їх перевірки як всередині навчального закладу, так і зовнішніми установами.

Метою роботи є розробка інформаційної системи обліку та автоматизації видачі студентських довідок із впровадженням механізму QR-верифікації та онлайн перевірки їхньої справжності.

Для досягнення поставленої мети визначено такі завдання:

- проаналізувати сучасні системи електронного документообігу у закладах освіти;
- дослідити існуючі підходи до автоматизації видачі довідок та цифрової верифікації документів;
- сформувати вимоги до розроблюваної системи;
- спроєктувати архітектуру системи та базу даних;
- розробити механізм генерації та перевірки QR-кодів;
- реалізувати функціонал формування, збереження та видачі електронних довідок;
- провести тестування програмного забезпечення та аналіз отриманих результатів.

Об’єктом дослідження є процеси обліку студентів та видачі довідок у закладах вищої освіти.

Предметом дослідження є методи та програмні засоби автоматизації видачі студентських довідок із використанням QR-верифікації та онлайн перевірки справжності документів.

Методи дослідження. Використано аналіз літератури та ІС, системний аналіз, проєктування БД, об’єктно-орієнтоване програмування, тестування ПЗ та методи порівняння ефективності рішень.

Короткий історичний розвиток електронного документообігу показує перехід від паперових архівів до повністю цифрових систем управління даними. Спочатку автоматизація охоплювала лише зберігання інформації, однак із розвитком вебтехнологій та мобільних рішень з’явилися системи електронного документообігу, що забезпечують повний цикл роботи з документами – від створення до перевірки автентичності. В останні роки активно впроваджуються технології QR-кодування та криптографічного підпису, що значно підвищує рівень безпеки цифрових документів.

Практичне значення отриманих результатів полягає у можливості впровадження розробленої системи в роботу деканатів та адміністративних підрозділів закладів вищої освіти. Система дозволяє автоматизувати процес

видачі довідок, зменшити навантаження на персонал, мінімізувати людський фактор та забезпечити швидку перевірку справжності документів через QR-код.

Теоретичне значення роботи полягає у систематизації підходів до побудови інформаційних систем електронного документообігу та дослідженні методів цифрової верифікації документів у освітньому середовищі.

Наукова новизна полягає у спробі поєднанні автоматизованої системи видачі студентських довідок із механізмом онлайн-верифікації та QR-перевірки, що забезпечує комплексний підхід до захисту та перевірки автентичності освітніх документів у межах єдиної інформаційної системи.

# 1 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ СИСТЕМ ЕЛЕКТРОННОГО ДОКУМЕНТООБІГУ ТА ВЕРИФІКАЦІЇ ДАНИХ СТУДЕНТІВ

## 1.1 Поняття та особливості систем електронного документообігу у закладах освіти

Системи електронного документообігу (СЕДО) у закладах освіти є складними інформаційно-організаційними системами, що забезпечують автоматизоване управління документами протягом усього їх життєвого циклу. До такого циклу належать створення, реєстрація, обробка, погодження, підписання, передавання, зберігання, архівування та контроль виконання документів. У контексті освітніх установ СЕДО охоплюють як адміністративні документи (накази, розпорядження, протоколи), так і навчально-організаційні матеріали (журнали, відомості, звіти, довідки студентів) [1].

У сучасних умовах цифровізації освіти електронний документообіг розглядається як один із ключових інструментів підвищення ефективності управління закладом освіти. Його впровадження дозволяє зменшити обсяг паперової документації, скоротити час обробки інформації, мінімізувати вплив людського фактора та забезпечити більш високий рівень точності та достовірності даних. Крім того, СЕДО сприяє формуванню єдиного інформаційного простору закладу освіти, де всі документи доступні в електронному вигляді та можуть оперативно використовуватися різними підрозділами.

Нормативно-правову основу функціонування електронного документообігу в Україні становить Закон України «Про електронні документи та електронний документообіг» [2]. Відповідно до нього електронний документ визначається як інформація в електронній формі, яка містить обов’язкові реквізити та може використовуватися як аналог паперового документа за умови дотримання встановлених вимог щодо його формування та підписання. Закон

також регламентує повний процес обігу електронних документів, включаючи їх створення, передачу, зберігання та доступ до них.

У закладах освіти порядок впровадження СЕДО визначається також внутрішніми нормативними актами та інструкціями з діловодства, які розробляються відповідно до вимог Міністерства освіти і науки України. Це означає, що кожен заклад освіти може самостійно адаптувати систему електронного документообігу відповідно до власної структури, потреб та рівня цифрової інфраструктури.

З наукової точки зору СЕДО у закладах освіти визначається як інтегрована організаційно-технічна система, що забезпечує автоматизацію процесів роботи з документами та підвищує ефективність управління освітнім процесом. Вона є складовою частиною загального інформаційного середовища закладу освіти та взаємодіє з іншими цифровими системами, такими як електронні журнали, системи управління навчанням (LMS), студентські портали та кадрові системи [3; 4].

Однією з ключових особливостей СЕДО є забезпечення повного життєвого циклу документа в електронній формі. На етапі створення використовуються стандартизовані шаблони документів, що забезпечує уніфікацію документообігу в межах закладу. Далі документ проходить етапи реєстрації, маршрутизації, погодження та підписання, у тому числі із застосуванням кваліфікованого електронного підпису. Після цього документ передається виконавцям, контролюється його виконання, а завершальний етап включає зберігання або архівування відповідно до встановлених термінів [5].

Важливою функціональною особливістю СЕДО є автоматизація управлінських процесів. Це дозволяє значно зменшити навантаження на адміністративний персонал, прискорити формування звітності та забезпечити оперативний доступ до актуальної інформації для прийняття управлінських рішень. Крім того, електронний формат роботи з документами знижує ймовірність помилок, дублювання інформації та втрати даних.

Окремо слід виділити підтримку дистанційної взаємодії між учасниками освітнього процесу. СЕДО дозволяє студентам, викладачам та адміністрації отримувати необхідні документи (довідки, витяги, накази) у цифровому форматі без необхідності особистого звернення до деканату або адміністративних підрозділів. Це особливо актуально в умовах розвитку дистанційного та змішаного навчання, коли швидкість доступу до інформації є критично важливою [1].

Ще однією важливою особливістю є забезпечення прозорості та контрольованості діяльності закладу освіти. Електронний документообіг дозволяє фіксувати всі дії з документами, що підвищує рівень відповідальності виконавців і забезпечує можливість аудиту. Це також сприяє підвищенню довіри до управлінських процесів з боку студентів, викладачів та зовнішніх зацікавлених сторін [6].

Важливим напрямом розвитку СЕДО є її інтеграція з іншими інформаційними системами закладу освіти. Завдяки цьому формується єдина цифрова екосистема, у якій дані вводяться один раз і використовуються у різних підсистемах без дублювання. Такий підхід підвищує ефективність роботи адміністрації та викладачів і забезпечує узгодженість інформаційних потоків [5].

Особливості впровадження систем електронного документообігу залежать від типу закладу освіти. У школах основна увага приділяється електронним журналам, обліку навчальних досягнень та кадровому діловодству. У закладах вищої освіти СЕДО має більш складну структуру та охоплює також наукову діяльність, студентське діловодство, фінансово-адміністративні процеси та сервіси для здобувачів освіти. У спеціалізованих закладах система адаптується під специфіку освітнього процесу та внутрішні організаційні потреби.

Таким чином, системи електронного документообігу у закладах освіти є невід’ємною складовою сучасної цифрової трансформації освіти. Вони забезпечують ефективне управління інформаційними потоками, підвищують якість адміністративних процесів, сприяють розвитку електронних сервісів та створюють основу для подальшої автоматизації освітнього середовища.

## 1.2 Аналіз сучасних систем автоматизації видачі довідок та цифрової верифікації документів

Сучасний стан цифровізації освіти висуває нові вимоги до систем управління університетами, зокрема в частині автоматизації адміністративних процесів, пов’язаних з оформленням та видачею документів студентам. Проблема автоматизації видачі довідок є багатоаспектною і охоплює питання електронного документообігу, верифікації особистості, захисту від підроблення та забезпечення зручного онлайн-доступу до послуг. Для визначення концепції розроблюваної системи необхідно провести аналіз наявних рішень як на рівні окремих університетів, так і на державному рівні.

Протягом останніх років в Україні сформувалась розгалужена інфраструктура цифрового обліку в освіті. Центральним елементом цієї інфраструктури є Єдина державна електронна база з питань освіти (ЄДЕБО), створена постановою Кабінету Міністрів України ще у 2011 році та суттєво розширена в наступні роки. Станом на 2024 рік ЄДЕБО охоплює 881 заклад вищої та фахової передвищої освіти, а також понад 1968 органів управління освітою [7]. До обов’язкових складових системи належать: Реєстр суб’єктів освітньої діяльності, Реєстр документів про освіту, Реєстр сертифікатів зовнішнього незалежного оцінювання та Реєстр студентських квитків [7]. Проте ЄДЕБО є передусім державним реєстром, а не системою обслуговування студентів: вона забезпечує облік і перевірку документів про освіту, але не автоматизує повний цикл видачі довідок на рівні окремого закладу.

На рівні окремих університетів в Україні поширені програмні комплекси класу АСУ ВНЗ (автоматизована система управління вищим навчальним закладом). Типова така система складається з модулів «Деканат», «Навчальна частина», «Абітурієнт», «Кафедра», «Відділ кадрів» та веб-порталу для студентів. Веб-портал дозволяє переглядати розклад, успішність, навчальні плани, контролювати оплату за навчання [8]. Проте функціонал

автоматизованого формування та видачі довідок у більшості таких систем або відсутній, або реалізований у спрощеному вигляді. Студент, як правило, змушений звертатись до деканату особисто або надсилати запит електронною поштою, вказуючи своє ПІБ, факультет, спеціальність, курс і мету отримання документа [9].

Аналіз вітчизняних рішень дозволяє виділити декілька підходів до часткової автоматизації видачі документів студентам:

– особистий кабінет студента з можливістю подання онлайн-заявки та завантаження готової довідки (реалізований в окремих провідних університетах);
– електронний документообіг (ЕДО) через спеціалізовані платформи, що передбачає маршрутизацію документів між підрозділами із застосуванням кваліфікованого електронного підпису (КЕП);
– інтеграція із державним порталом «Дія» для підтвердження особи та доступу до цифрових документів.

Важливу роль у загальній екосистемі цифрових документів відіграє застосунок «Дія», яким станом на 2026 рік користуються понад 24 мільйони українців, а кількість доступних електронних послуг перевищила 130 [10]. Платформа забезпечує верифікацію документів через QR-код і реалізує принцип «держава у смартфоні» – можливість пред’явити цифровий документ без паперового оригіналу. Сервіс Дія.Підпис дозволяє здійснювати кваліфікований електронний підпис документів у мобільному додатку, що відкриває можливості для інтеграції університетських систем із загальнодержавною інфраструктурою [10]. Втім, підключення закладу освіти до сервісів Дії вимагає значних організаційно-технічних ресурсів і наразі не є масовою практикою для видачі конкретних довідок студентам.

На міжнародному рівні верифікація документів за допомогою QR-кодів набула широкого поширення у сфері вищої освіти. Дослідники зазначають, що традиційні методи захисту документів – водяні знаки, голограми, підписи – вже не є достатнім засобом захисту від підробок: кількість виявлених підроблених

шаблонів документів зросла на 50% у період з 2023 по 2024 рік [11]. QR-код, вбудований в офіційний документ, посилається на захищений цифровий запис у базі видавця, що дозволяє миттєво перевірити автентичність документа за допомогою смартфона.

Провідні університети світу вже впровадили такі рішення. Університет Джонса Гопкінса (США) з 2016 року видає всім випускникам криптографічно підписані цифрові дипломи з унікальним ідентифікатором, а з жовтня 2024 року на зворотному боці паперових дипломів розміщується QR-код SURETYLOCK™, сканування якого надає в режимі реального часу автентифікаційну інформацію: ім’я власника, дату підтвердження та назву установи [12].

Комісія університетських грантів Індії зобов’язала всі університети розміщувати QR-коди на дипломах для запобігання фальсифікаціям та спрощення перевірки [11].

Університет штату Крос-Рівер (Нігерія) додав QR-коди до сертифікатів для боротьби з фальшивими документами [11].

З технологічного погляду QR-верифікація може реалізовуватись у трьох основних архітектурних варіантах:

- централізована база даних – QR-код містить унікальний ідентифікатор, за яким система звертається до центрального сервера для отримання записів про документ. Перевагою є простота реалізації, недоліком – залежність від доступності серверу та ризик єдиної точки відмови;
- Blockchain-верифікація – дані про документ записуються в незмінний блокчейн-реєстр. Це забезпечує децентралізоване зберігання, незмінність записів і прозорість. Академічні дослідження 2024 року демонструють перспективність застосування смарт-контрактів Ethereum та Hyperledger Fabric для реєстрації академічних сертифікатів [13]. Виявлено, що платформа Polygon (zkEVM) забезпечує до 94% економії витрат порівняно з Ethereum Mainnet при операціях видачі та відкликання сертифікатів [14];

– криптографічний підпис у самому QR-коді – документ містить криптографічно підписані дані безпосередньо в QR-коді, що дозволяє перевіряти достовірність навіть без доступу до інтернету. Такий підхід застосовує, зокрема, компанія Qryptal у своїх рішеннях для університетів [15; 16].

Порівняльний аналіз зазначених підходів наведено в таблиці 1.1.

Таблиця 1.1 – Порівняння підходів до QR-верифікації документів

| **Критерій** | **Централізована БД** | **Blockchain** | **Крипто-підпис у QR** |
|---|---|---|---|
| Складність впровадження | Низька | Висока | Середня |
| Захист від підробок | Середній | Дуже високий | Високий |
| Офлайн-верифікація | ✗ Ні | ✗ Ні | ✓ Так |
| Відкликання / оновлення документа | ✓ Легко | Складно | Складно |
| Залежність від сервера | Висока | Низька (децентраліз.) | Відсутня |
| Вартість експлуатації | Низька | Висока (gas fees) | Низька |
| Придатність для видачі довідок ЗВО | 4 | 3 | 5 |

Централізований підхід характеризується простотою впровадження та зручністю адміністрування, однак вразливий щодо доступності сервера та ризику атак на єдиний вузол. Блокчейн-підхід забезпечує найвищий рівень стійкості до підробок та незмінність записів, але потребує значних технічних компетенцій і операційних витрат. Криптографічний підпис у QR-коді є оптимальним для офлайн-верифікації, однак ускладнює відкликання або оновлення документів.

Варто окремо розглянути специфіку задачі видачі саме довідок (а не дипломів): на відміну від фінального освітнього документа, довідка є оперативним підтвердженням поточного статусу студента і може видаватись багаторазово з різних підстав (підтвердження навчання, місце проходження практики, відстрочка від призову тощо). Це ставить вимогу швидкого

автоматизованого формування документа, гнучкого шаблонування та надійного онлайн-каналу верифікації.

Документообіг при видачі довідок є значно інтенсивнішим, ніж при видачі дипломів, що робить автоматизацію цього процесу особливо актуальною.

Проведений аналіз виявив низку недоліків наявних рішень, які обумовлюють доцільність розробки спеціалізованої системи:

- більшість університетських АСУ не мають повного циклу автоматизованої видачі довідок із верифікацією – студент змушений особисто звертатись до деканату або надсилати запит електронною поштою [9];
- ЄДЕБО виконує функцію державного реєстру, а не сервісу для студентів, і не охоплює оперативну видачу внутрішніх довідок закладу;
- QR-верифікація документів практично не застосовується у вітчизняних університетах для поточних довідок, тоді як міжнародний досвід демонструє її ефективність та зростаючу необхідність [11];
- відсутні комплексні рішення, що поєднують: онлайн-подання заявки, автоматичне формування документа, накладення КЕП, генерацію QR-коду та публічний сервіс онлайн-перевірки автентичності.

Таким чином, аналіз сучасних систем підтверджує актуальність та практичну значимість розробки інтегрованої системи обліку та автоматизації видачі довідок студентам із QR-верифікацією.

Запропонована система має усунути виявлені недоліки, забезпечивши повний цикл – від подання заявки студентом у особистому кабінеті до миттєвої онлайн-перевірки виданого документа третьою стороною – і тим самим підвищити ефективність адміністративних процесів університету та рівень захисту документів від підробок.

## 1.3 Вимоги до системи обліку та автоматизації видачі студентських довідок із QR-верифікацією

Проєктована система обліку та автоматизації видачі студентських довідок із QR-верифікацією та онлайн перевіркою справжності документів належить до класу інформаційних систем електронного документообігу, орієнтованих на автоматизацію адміністративних процесів у закладах освіти. Основною метою такої системи є забезпечення швидкого, безпечного та контрольованого формування студентських довідок, їх зберігання, а також можливості перевірки автентичності через цифрові механізми.

Формування вимог до системи є ключовим етапом проєктування, оскільки саме вони визначають функціональність, архітектуру, рівень безпеки та якість майбутнього програмного продукту. Вимоги поділяються на функціональні, які описують поведінку системи, та нефункціональні, які визначають її якісні характеристики.

Функціональні вимоги (табл. 1.2) визначають перелік основних можливостей системи, які забезпечують повний цикл роботи з довідками студентів – від створення до перевірки їхньої справжності.

Таблиця 1.2 містить функціональні вимоги до системи обліку та автоматизації видачі студентських довідок із QR-верифікацією. У ній визначено основні можливості системи, які охоплюють реєстрацію та автентифікацію користувачів, подання студентами заявок на отримання довідок, перегляд їхнього статусу та отримання вже сформованих документів. Окремо в таблиці наведено адміністративні функції, що забезпечують роботу працівників деканату: створення та редагування облікових записів студентів, опрацювання заявок, підтвердження або відхилення запитів, а також можливість ручної видачі та анулювання довідок. Крім того, таблиця включає системні функції, такі як автоматична генерація довідок після підтвердження заявки, створення QR-кодів для кожного документа, ведення журналу перевірок та забезпечення онлайн-перевірки справжності довідок за унікальним ідентифікатором. Також

передбачено функцію відображення статистичних даних для адміністратора, що дозволяє аналізувати роботу системи.

Таблиця 1.2 – Функціональні вимоги до системи

| Код | Вимога | Роль | Пріоритет |
|---|---|---|---|
| FR-01 | Реєстрація нового облікового запису студента (email, пароль, ПІБ, номер студентського квитка) | Студент | Високий |
| FR-02 | Автентифікація користувача за email та паролем | Студент, Адміністратор | Високий |
| FR-03 | Завершення сесії користувача (вихід із системи) | Студент, Адміністратор | Високий |
| FR-04 | Подання заявки на отримання довідки з вибором типу та призначення | Студент | Високий |
| FR-05 | Перегляд статусу поданих заявок (очікує, підтверджено, відхилено) | Студент | Високий |
| FR-06 | Перегляд виданих довідок та їх QR-кодів | Студент | Високий |
| FR-07 | Створення та редагування облікових записів студентів | Адміністратор | Середній |
| FR-08 | Розгляд та обробка заявок (підтвердження або відхилення) | Адміністратор | Високий |
| FR-09 | Автоматичне формування довідки з унікальним ідентифікатором після підтвердження | Система | Високий |
| FR-10 | Можливість ручної видачі довідки без поданої заявки | Адміністратор | Середній |
| FR-11 | Генерація QR-коду для кожної створеної довідки | Система | Високий |
| FR-12 | Анулювання або відкликання виданої довідки | Адміністратор | Високий |
| FR-13 | Онлайн-перевірка справжності довідки за QR-кодом або унікальним ID | Будь-який користувач | Високий |
| FR-14 | Відображення статистики роботи системи (довідки, заявки, користувачі) | Адміністратор | Низький |
| FR-15 | Журналювання перевірок довідок у системі | Система | Середній |

Нефункціональні вимоги (табл. 1.3) визначають якісні характеристики системи, які впливають на її стабільність, зручність використання, продуктивність та можливість подальшого розвитку.

Таблиця 1.3 містить нефункціональні вимоги до системи обліку та автоматизації видачі студентських довідок із QR-верифікацією. У ній визначено якісні характеристики системи, які не описують її конкретні функції, але впливають на загальну ефективність, зручність використання та стабільність роботи.

Таблиця 1.3 – Нефункціональні вимоги системи

| Код | Категорія | Вимога |
|---|---|---|
| NFR-01 | Продуктивність | Час відповіді сторінки перевірки довідки не перевищує 2 секунд |
| NFR-02 | Зручність | Адаптивний інтерфейс для мобільних пристроїв і ПК |
| NFR-03 | Доступність | Відповідність базовим принципам WCAG (доступність інтерфейсу) |
| NFR-04 | Локалізація | Повна підтримка української мови |
| NFR-05 | Надійність | Забезпечення цілісності даних через транзакції та ключі БД |
| NFR-06 | Розширюваність | Можливість додавання нових типів довідок без зміни структури бази даних |
| NFR-07 | Портативність | Використання SQLite для спрощеного розгортання системи |
| NFR-08 | Підтримуваність | Модульна архітектура системи з розділенням логіки та інтерфейсу |

Зокрема, вимоги до продуктивності передбачають мінімальний час відгуку системи під час перевірки довідок, що є важливим для забезпечення швидкої онлайн-верифікації. Вимоги до зручності визначають необхідність адаптивного інтерфейсу, який коректно працює як на мобільних пристроях, так і на персональних комп’ютерах.

Окремо в таблиці наведено вимоги до доступності, які передбачають дотримання базових принципів WCAG для забезпечення зручності використання системи різними категоріями користувачів. Вимога локалізації встановлює повну підтримку української мови інтерфейсу.

Також визначено вимоги до надійності, які забезпечують цілісність даних завдяки використанню транзакцій та механізмів бази даних. Вимога розширюваності передбачає можливість додавання нових типів довідок без зміни структури бази даних, що дозволяє легко масштабувати систему.

Вимоги до портативності визначають використання SQLite як простої у розгортанні бази даних, а вимоги до підтримуваності передбачають модульну архітектуру системи з чітким розділенням бізнес-логіки, інтерфейсу та роботи з даними.

Оскільки система працює з персональними даними студентів та електронними документами, вимоги до безпеки є критично важливими і визначають механізми захисту інформації та перевірки автентичності довідок (табл. 1.4).

Таблиця 1.4 – Вимоги до безпеки та верифікації

| Код | Вимога |
|---|---|
| SR-01 | Паролі користувачів зберігаються у вигляді криптографічного хешу (scrypt із сіллю) |
| SR-02 | Доступ до адміністративних функцій дозволено лише користувачам із роллю адміністратора |
| SR-03 | Сесії користувачів реалізуються через захищені HTTP-only токени з обмеженим часом дії |
| SR-04 | Кожна довідка підписується криптографічним підписом HMAC-SHA256 |
| SR-05 | Будь-яка зміна даних довідки автоматично робить її недійсною при перевірці |
| SR-06 | Захист від SQL-ін’єкцій реалізовано через параметризовані запити до бази даних |
| SR-07 | Публічна сторінка перевірки доступна без авторизації, але не дозволяє змінювати дані |

Таблиця 1.4 містить вимоги до безпеки та верифікації системи обліку та автоматизації видачі студентських довідок із QR-верифікацією. У ній визначено основні механізми захисту даних, автентифікації користувачів та забезпечення цілісності електронних документів.

Зокрема, вимоги SR-01–SR-03 стосуються захисту облікових даних користувачів і доступу до системи. Вони передбачають зберігання паролів у вигляді криптографічних хешів, обмеження доступу до адміністративних функцій лише для користувачів із відповідною роллю, а також використання захищених сесій із HTTP-only токенами для запобігання несанкціонованому доступу.

Вимоги SR-04–SR-05 визначають механізм захисту електронних довідок. Кожен документ підписується криптографічним підписом HMAC-SHA256, що дозволяє перевіряти його автентичність. У разі будь-якої зміни даних довідка

автоматично вважається недійсною під час перевірки, що запобігає підробці документів.

Вимога SR-06 описує захист системи від SQL-ін’єкцій шляхом використання параметризованих запитів до бази даних, що підвищує загальний рівень безпеки інформаційної системи.

Остання вимога SR-07 регламентує доступ до публічної сторінки перевірки довідок, яка є відкритою для всіх користувачів, але не надає можливості змінювати дані або отримувати доступ до конфіденційної інформації.

Отже, визначено основні вимоги до системи обліку та автоматизації видачі студентських довідок із QR-верифікацією. Встановлено функціональні вимоги, які описують ключові можливості системи для студентів, адміністратора та системних процесів, зокрема реєстрацію, подання заявок, формування довідок і їх перевірку.

Нефункціональні вимоги визначають якісні характеристики системи, такі як продуктивність, зручність використання, доступність, надійність, розширюваність та підтримуваність. Окрему увагу приділено забезпеченню стабільної роботи системи та її адаптивності до різних пристроїв.

Вимоги до безпеки та верифікації регламентують захист даних користувачів, контроль доступу, захищене зберігання паролів, а також механізми криптографічного підпису та перевірки справжності довідок. Загалом сформовані вимоги створюють основу для подальшого проєктування та реалізації системи.

# 2 ПРОЄКТУВАННЯ СИСТЕМИ ОБЛІКУ ТА АВТОМАТИЗАЦІЇ ВИДАЧІ СТУДЕНТСЬКИХ ДОВІДОК

## 2.1 Аналіз предметної області та постановка задачі системи

Предметною областю даного дослідження є процес обліку та видачі студентських довідок у закладах вищої освіти, який традиційно реалізується через деканати або інші адміністративні підрозділи. Студентська довідка виступає офіційним документом, що підтверджує факт навчання особи у закладі освіти та використовується для подання до різних установ, зокрема державних органів, банківських установ, роботодавців або інших організацій, які потребують підтвердження статусу студента.

У класичному підході процес видачі довідок є переважно паперовим і передбачає значну кількість ручних операцій, починаючи від подання запиту студентом і завершуючи видачею готового документа. Така організація роботи призводить до суттєвих витрат часу як з боку адміністративного персоналу, так і з боку студентів, які змушені особисто звертатися до деканату, очікувати в чергах та контролювати статус виконання запиту. Крім того, відсутність централізованої цифрової системи ускладнює облік уже виданих довідок, а також створює ризики втрати або дублювання інформації.

Особливої актуальності набуває проблема перевірки справжності довідок, оскільки традиційні паперові документи, навіть із печатками та підписами, можуть бути підробленими або зміненими. У таких умовах виникає потреба у створенні цифрового механізму верифікації, який дозволив би будь-якій сторонній установі швидко та надійно перевірити достовірність документа без необхідності звернення до навчального закладу.

Для більш детального розуміння предметної області доцільно розглянути інформаційні потоки (рис. 2.1), що виникають у процесі видачі студентської довідки.

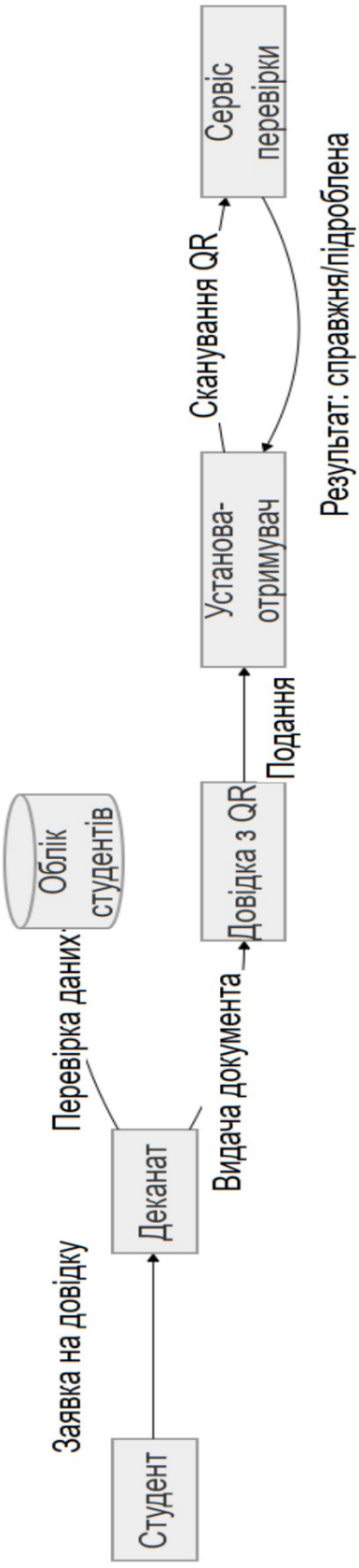



Рисунок 2.1 – Діаграма інформаційних потоків процесу видачі студентських довідок

Як видно з наведеної діаграми, процес починається з ініціації студентом запиту на отримання довідки, який надходить до працівників деканату для подальшої обробки. Після перевірки актуальності даних формується електронна або паперова довідка, якій автоматично присвоюється унікальний ідентифікатор. Наступним етапом є генерація QR-коду, що містить зашифровану інформацію або посилання на онлайн-сервіс перевірки. Завершальним етапом є можливість зовнішньої перевірки документа, яка дозволяє підтвердити або спростувати його справжність у режимі реального часу.

Додатково для формалізації взаємодії користувачів із системою було побудовано діаграму варіантів використання (рис. 2.2), яка дозволяє визначити основні функціональні можливості системи та ролі її учасників.

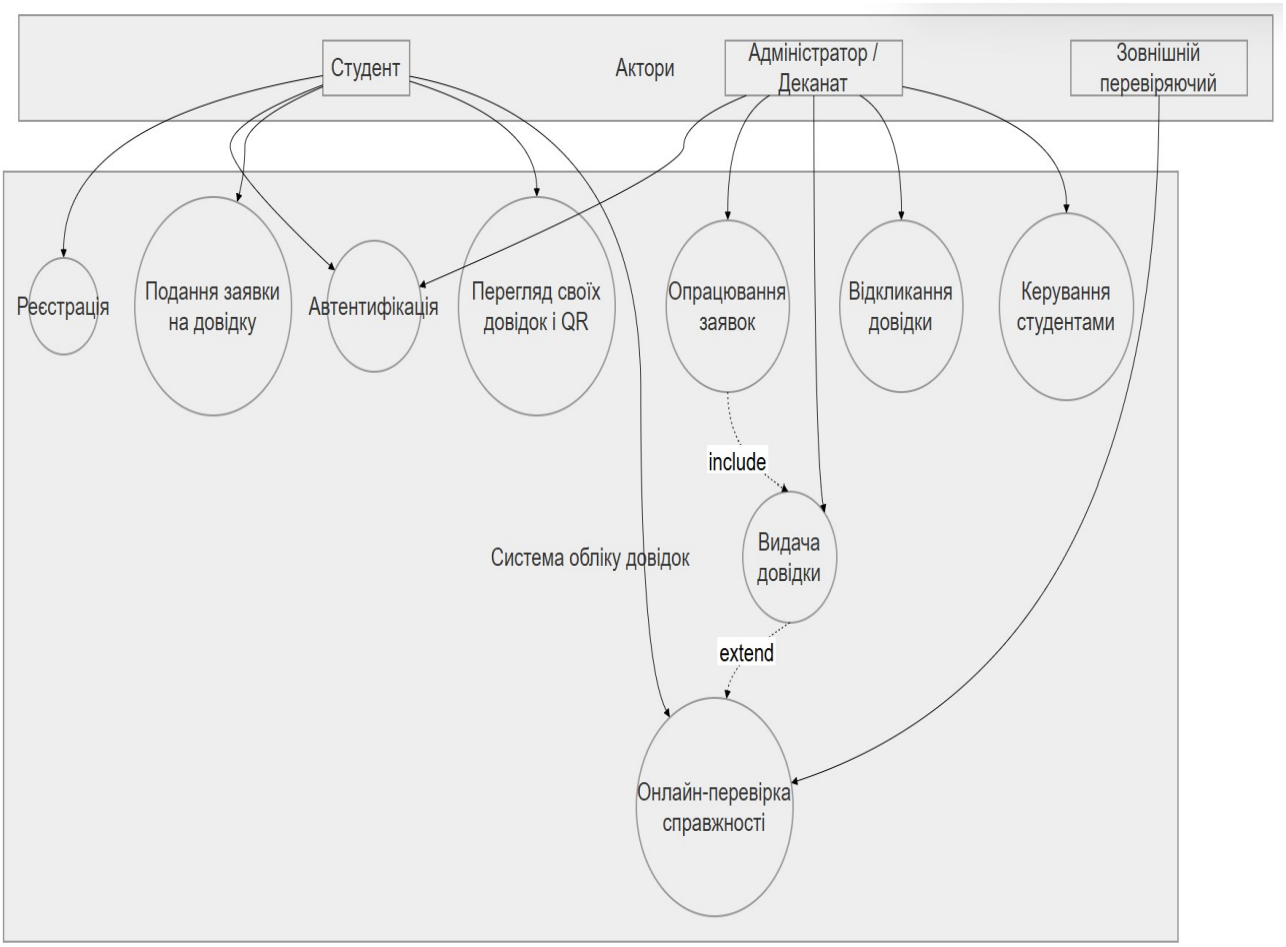



Рисунок 2.2 – Діаграма варіантів використання системи обліку та видачі студентських довідок

Аналіз діаграми варіантів використання дозволяє зробити висновок, що система передбачає три основні категорії користувачів: студента, адміністратора та зовнішнього перевіряючого. Кожна з ролей має чітко визначені права доступу та функціональні можливості. Студент взаємодіє із системою через реєстрацію, авторизацію, подання запитів та перегляд власних довідок. Адміністратор забезпечує повний цикл обробки заявок, включаючи їх перевірку, формування та відкликання документів, а також управління студентськими обліковими записами. Зовнішній користувач, у свою чергу, має обмежений доступ і може лише здійснювати перевірку справжності довідки за допомогою QR-коду або унікального ідентифікатора.

Підсумовуючи, можна зазначити, що постановка задачі полягає у створенні вебзастосунку, який забезпечує повну автоматизацію процесу обліку та видачі студентських довідок. Така система повинна охоплювати всі етапи життєвого циклу документа – від подання заявки до його верифікації третіми сторонами. Особлива увага приділяється впровадженню механізму QR-верифікації, який дозволяє забезпечити прозорість, достовірність та захищеність електронних документів.

У результаті визначено, що вхідними даними системи є персональні дані студента, тип та призначення довідки, а також облікові дані для автентифікації. Вихідними даними виступають сформовані електронні довідки з унікальними ідентифікаторами, QR-коди, результати перевірки справжності та статистична інформація щодо роботи системи.

## 2.2 Проєктування архітектури системи та бази даних

Архітектура системи обліку та автоматизації видачі студентських довідок із QR-верифікацією спроєктована за багаторівневим (layered) принципом, який передбачає логічне розділення системи на незалежні функціональні рівні. Такий

підхід дозволяє ізолювати бізнес-логіку від інтерфейсу користувача та процесів зберігання даних, що підвищує структурованість системи, спрощує її супровід і забезпечує можливість подальшого масштабування.

У межах запропонованої архітектури виділяється три основні рівні. Перший – рівень представлення, який відповідає за взаємодію користувача із системою. На цьому рівні реалізуються інтерфейси для студентів, адміністраторів та зовнішніх користувачів, які здійснюють перевірку довідок. Саме тут відбувається введення даних, перегляд інформації та отримання результатів обробки запитів.

Другий рівень – рівень бізнес-логіки, який забезпечує основну функціональність системи. Він відповідає за обробку заявок на довідки, перевірку коректності введених даних, формування електронних документів, керування їхнім життєвим циклом, а також реалізацію механізму генерації унікальних ідентифікаторів та QR-кодів. Крім того, на цьому рівні виконується контроль доступу користувачів відповідно до їхніх ролей.

Третій рівень – рівень доступу до даних, який забезпечує збереження, отримання та оновлення інформації в базі даних. Він відповідає за взаємодію з таблицями системи, підтримку цілісності даних, виконання транзакційних операцій та забезпечення коректного зв’язку між сутностями.

Загальна структура архітектури системи представлена на рисунку 2.3.

Запропонована архітектура забезпечує чіткий поділ відповідальностей між компонентами системи, що дозволяє уникнути змішування інтерфейсної логіки з логікою обробки даних. Це підвищує зрозумілість системи та полегшує її подальший розвиток, зокрема у частині розширення функціоналу або зміни способів взаємодії з користувачем.

Проєктування бази даних здійснюється на основі аналізу предметної області та визначення основних сутностей, які беруть участь у процесі формування, видачі та перевірки студентських довідок. Модель даних повинна забезпечувати зберігання інформації про студентів, користувачів системи, заявки на довідки, самі довідки, а також результати їх перевірки.

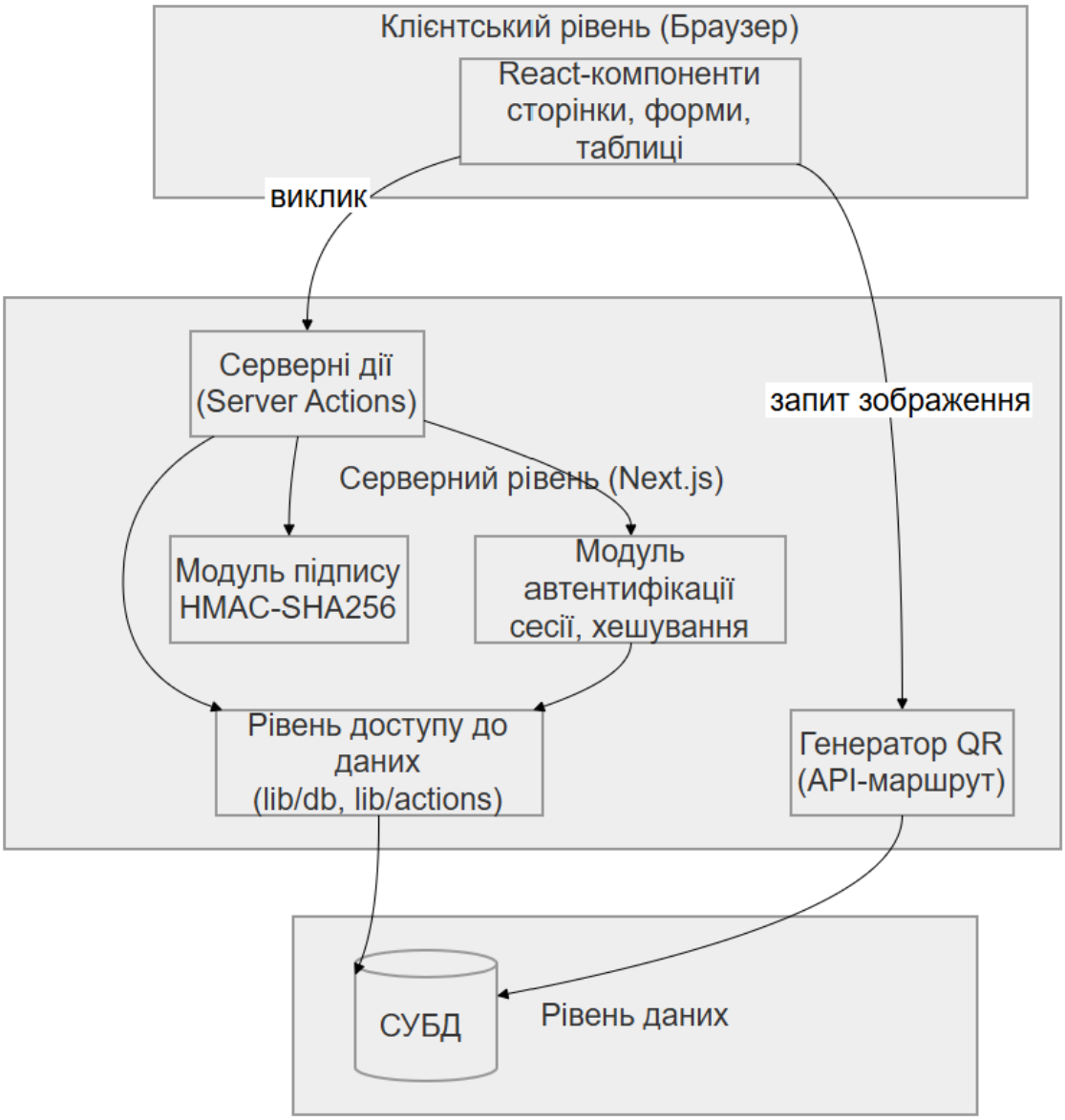



Рисунок 2.3 – Узагальнена багаторівнева архітектура системи обліку та автоматизації видачі студентських довідок

Важливою вимогою до бази даних є забезпечення цілісності інформації, уникнення надлишковості даних та підтримка зв’язків між сутностями. Усі сутності повинні бути логічно пов’язані між собою таким чином, щоб забезпечувався повний життєвий цикл довідки – від моменту подання заявки до її перевірки.

Основними сутностями системи є: Student (Студент), User (Користувач), Request (Заявка), Certificate (Довідка) та Verification (Перевірка). Кожна з цих

сутностей виконує окрему функцію в межах системи, але разом вони формують цілісну інформаційну модель.

Концептуальна модель даних (рис. 2.4) відображає структуру системи у вигляді сутностей та зв’язків між ними. Вона дозволяє формалізувати предметну область і є основою для подальшого логічного та фізичного проєктування бази даних.

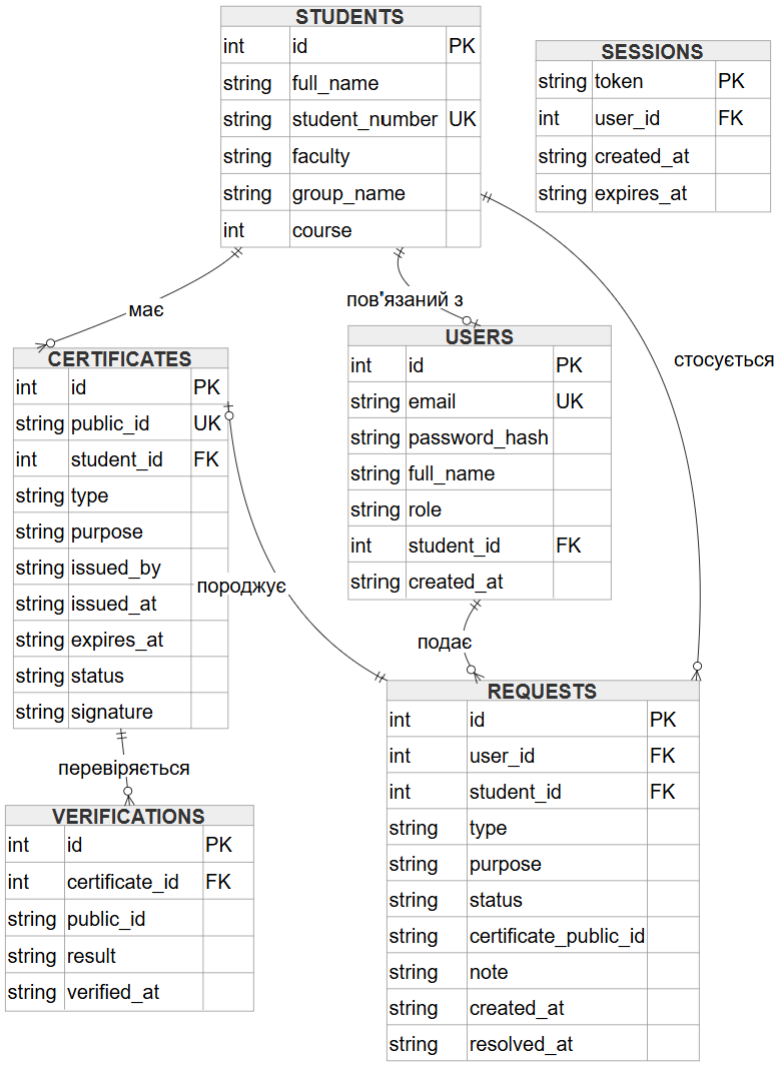



Рисунок 2.4 – Концептуальна ER-модель системи обліку та автоматизації видачі студентських довідок

Концептуальна ER-діаграма відображає структуру бази даних системи обліку та автоматизації видачі студентських довідок із QR-верифікацією, а також логічні зв’язки між основними сутностями предметної області. Вона дозволяє

простежити повний життєвий цикл довідки – від моменту подання заявки студентом до її перевірки зовнішнім користувачем.

Центральною сутністю моделі є таблиця STUDENTS, яка містить основні дані про студентів, включаючи ПІБ, унікальний номер студентського квитка, факультет, навчальну групу та курс. Саме ця сутність виступає базовою точкою зв’язку для більшості інших елементів системи, оскільки всі операції з довідками та заявками прив’язані до конкретного студента.

Зв’язок між STUDENTS та CERTIFICATES реалізує відношення «один до багатьох», що означає, що один студент може отримати кілька довідок у різний час. Це відображає реальну ситуацію в освітніх закладах, де довідки видаються багаторазово залежно від потреб студента. Кожна довідка зберігає унікальний ідентифікатор, тип, призначення, дату видачі, строк дії, статус та криптографічний підпис, що забезпечує її автентичність.

Сутність USERS відповідає за облікові записи користувачів системи та містить дані для автентифікації, такі як email, хеш пароля та роль користувача (студент або адміністратор). Між STUDENTS і USERS встановлено зв’язок «один до одного або один до нуля», що означає, що не кожен студент обов’язково має обліковий запис, але кожен обліковий запис, що належить студенту, пов’язаний з конкретною студентською сутністю.

Зв’язок між USERS та REQUESTS відображає процес подання заявок: один користувач може створювати багато заявок на отримання довідок. Водночас таблиця REQUESTS також пов’язана зі STUDENTS, що дозволяє відстежувати, до якого саме студента відноситься заявка, навіть якщо вона подана через адміністратора.

Сутність REQUESTS є проміжною ланкою між студентом і довідкою. Вона містить інформацію про тип довідки, її призначення, статус обробки, дату створення та дату завершення розгляду. Саме через цей механізм реалізується процес узгодження та прийняття рішень щодо видачі документа. У разі схвалення заявки вона «породжує» відповідну запис у таблиці CERTIFICATES, що відображає логічний перехід від запиту до офіційного документа.

Таблиця CERTIFICATES містить усю інформацію про видані довідки, включаючи публічний ідентифікатор, дані студента, тип довідки, статус, дату видачі та криптографічний підпис. Саме ця сутність є основною для подальшої перевірки документів.

Зв’язок між CERTIFICATES та VERIFICATIONS реалізує процес контролю справжності довідок. Одна довідка може проходити багато перевірок, наприклад при повторному скануванні QR-коду різними установами. Кожна перевірка фіксується окремим записом із результатом та часом виконання, що дозволяє вести журнал верифікацій.

Додатково в моделі присутня сутність SESSIONS, яка забезпечує управління активними сесіями користувачів. Вона зберігає токен сесії, прив’язаний до конкретного користувача, а також час створення та завершення дії сесії. Це дозволяє реалізувати контроль доступу та безпечну роботу із системою.

У цілому представлена ER-діаграма демонструє логічно завершену структуру даних, у якій кожна сутність виконує чітко визначену роль, а зв’язки між ними забезпечують цілісність процесу обліку, видачі та перевірки студентських довідок.

Логічна структура бази даних передбачає виділення таблиць відповідно до сутностей предметної області та встановлення між ними зв’язків через ключові поля. Така структура забезпечує нормалізацію даних, зменшення дублювання інформації та підвищення цілісності збережених даних.

Отже, здійснено проєктування архітектури системи обліку та автоматизації видачі студентських довідок із QR-верифікацією, а також розроблено концептуальну модель бази даних. Обґрунтовано використання багаторівневої архітектури, що забезпечує чітке розділення клієнтської частини, бізнес-логіки та доступу до даних, підвищуючи модульність і зручність супроводу системи.

Розроблена ER-діаграма відображає основні сутності предметної області та їх взаємозв’язки, що дозволяє реалізувати повний життєвий цикл студентської довідки – від створення заявки до її перевірки через QR-код. Запропонована

структура бази даних забезпечує цілісність даних і підтримку ключових функцій системи.

## 2.3 Проєктування механізму QR-верифікації та онлайн перевірки справжності документів

Основою механізму захисту студентських довідок від підробки є використання криптографічного цифрового підпису, що повністю виключає можливість несанкціонованого редагування даних без виявлення змін під час перевірки. На відміну від традиційних методів захисту, таких як мокрі печатки або візуальні елементи, запропонований підхід базується на математичній гарантії цілісності даних.

Під час створення довідки система формує криптографічний підпис із використанням алгоритму HMAC-SHA256 та секретного ключа сервера. Такий підпис зберігається разом із записом довідки у базі даних і використовується як еталон автентичності документа.

Формально процес підписування можна подати у вигляді виразу: $S=HMAC_{SHA256}(K,M)$, де K – секретний ключ сервера, який недоступний зовнішнім користувачам, а M – повідомлення, сформоване шляхом конкатенації ключових параметрів довідки:

Таким чином, будь-яка навіть мінімальна зміна одного з параметрів довідки призводить до повної зміни підпису, що дозволяє однозначно визначити факт підробки під час верифікації.

QR-код у системі виконує функцію швидкого доступу до сторінки перевірки документа. Він не містить конфіденційних даних, а лише унікальний ідентифікатор довідки, закодований у вигляді URL-адреси.

Типова структура посилання має вигляд: https://<домен>/verify/CERT-XXXX-XXXX-XXXX.

Такий підхід забезпечує максимальну простоту використання: будь-який стандартний QR-сканер автоматично перенаправляє користувача на вебсторінку перевірки, де відбувається повний аналіз документа. Водночас відсутність критичних даних у самому QR-коді підвищує безпеку системи, оскільки виключає можливість їх підробки або витоку.

Процес формування довідки є послідовністю взаємопов'язаних етапів, які починаються з підтвердження заявки адміністратором і завершуються генерацією QR-коду для студента. Після затвердження заявки система створює унікальний ідентифікатор довідки public_id, який використовується як основний ключ для подальших операцій.

Далі формується структуроване повідомлення, що включає ключові атрибути довідки. На його основі обчислюється криптографічний підпис за алгоритмом HMAC-SHA256. Отримане значення разом із даними довідки зберігається у базі даних, що забезпечує неможливість прихованої модифікації запису без втрати валідності підпису.

Після цього генерується QR-код, який містить посилання на сторінку перевірки. Сам QR-код передається студенту разом із довідкою і використовується як інструмент миттєвої верифікації документа будь-якою третьою стороною.

Процес онлайн-перевірки побудований як послідовність логічних перевірок, які виконуються після сканування QR-коду або введення унікального ідентифікатора довідки.

Спочатку система здійснює пошук довідки у базі даних за public_id. У разі відсутності запису користувач отримує результат «документ не знайдено». Якщо запис існує, система переходить до наступного етапу – перевірки криптографічного підпису.

На цьому етапі підпис обчислюється повторно на основі збережених даних і порівнюється з еталонним значенням. У випадку невідповідності система однозначно класифікує документ як підроблений. Якщо підпис збігається,

виконується додаткова перевірка статусу довідки (active, revoked) та терміну її дії.

Лише за умови проходження всіх перевірок документ вважається справжнім і валідним. Результат перевірки додатково записується у журнал верифікацій для подальшого аудиту та аналітики.

Результати верифікації формуються залежно від комбінації перевірених умов і відображають кінцевий стан документа. Якщо підпис збігається, а статус довідки є активним і термін дії не завершено, система визначає документ як справжній і підтверджений.

У випадку, коли довідку було відкликано адміністрацією, вона отримує статус «відкликана» незалежно від правильності підпису. Якщо термін дії документа завершився, система позначає його як прострочений. Найбільш критичним випадком є невідповідність підпису, що однозначно свідчить про зміну даних і класифікується як підробка. Якщо ж ідентифікатор відсутній у базі даних, документ вважається незареєстрованим.

Таким чином, запропонований механізм QR-верифікації забезпечує багаторівневу перевірку автентичності документів, поєднуючи криптографічний захист, контроль життєвого циклу довідки та можливість миттєвої онлайн-перевірки.

## 2.4 Проєктування користувацьких інтерфейсів та вибір технологій реалізації

Проєктування користувацького інтерфейсу системи обліку довідок базується на принципі мінімалізму, логічної структурованості та розподілу функціональності відповідно до ролей користувачів. Основною метою є забезпечення швидкого доступу до ключових операцій – подання заявок, перегляду довідок, адміністрування та перевірки справжності документів – без перевантаження інтерфейсу зайвими елементами. У межах цього підрозділу

визначено структуру сторінок, модель доступу та технологічні рішення для реалізації інтерфейсної частини системи.

На першому етапі проєктування було сформовано загальну карту сторінок застосунку (рис. 2.5), яка відображає логіку навігації між основними модулями системи. Вона дозволяє зрозуміти, як користувач переходить від публічної частини до персонального кабінету або адміністративної панелі, а також як реалізується сценарій перевірки довідки за QR-кодом.

Побудована карта сторінок демонструє поділ системи на три логічні зони: публічну частину, призначену для перевірки довідок; користувацький кабінет студента; та адміністративну панель. Така структура забезпечує логічну ізоляцію функціоналу та зменшує ризик перевантаження інтерфейсу.

Зокрема, публічна сторінка перевірки не потребує авторизації, що є критично важливим для швидкої верифікації документа сторонніми організаціями. Натомість кабінет студента містить інструменти для подання заявок та перегляду власних довідок, тоді як адміністративний модуль реалізує повний цикл управління документами.

Наступним етапом проєктування інтерфейсу є визначення прав доступу для різних категорій користувачів (табл. 2.1). Це дозволяє забезпечити безпеку системи та запобігти несанкціонованому доступу до адміністративних функцій.

Таблиця 2.1 – Матриця доступу до функціональних можливостей системи

| **Функціональна можливість** | **Гість** | **Студент** | **Адміністратор** |
|---|---|---|---|
| Перегляд головної сторінки | + | + | + |
| Перевірка довідки | + | + | + |
| Реєстрація та авторизація | + | – | – |
| Подання заявки на довідку | – | + | – |
| Перегляд власних довідок | – | + | – |
| Адміністративна панель | – | – | + |
| Обробка заявок | – | – | + |
| Видача та відкликання довідок | – | – | + |
| Керування студентами | – | – | + |

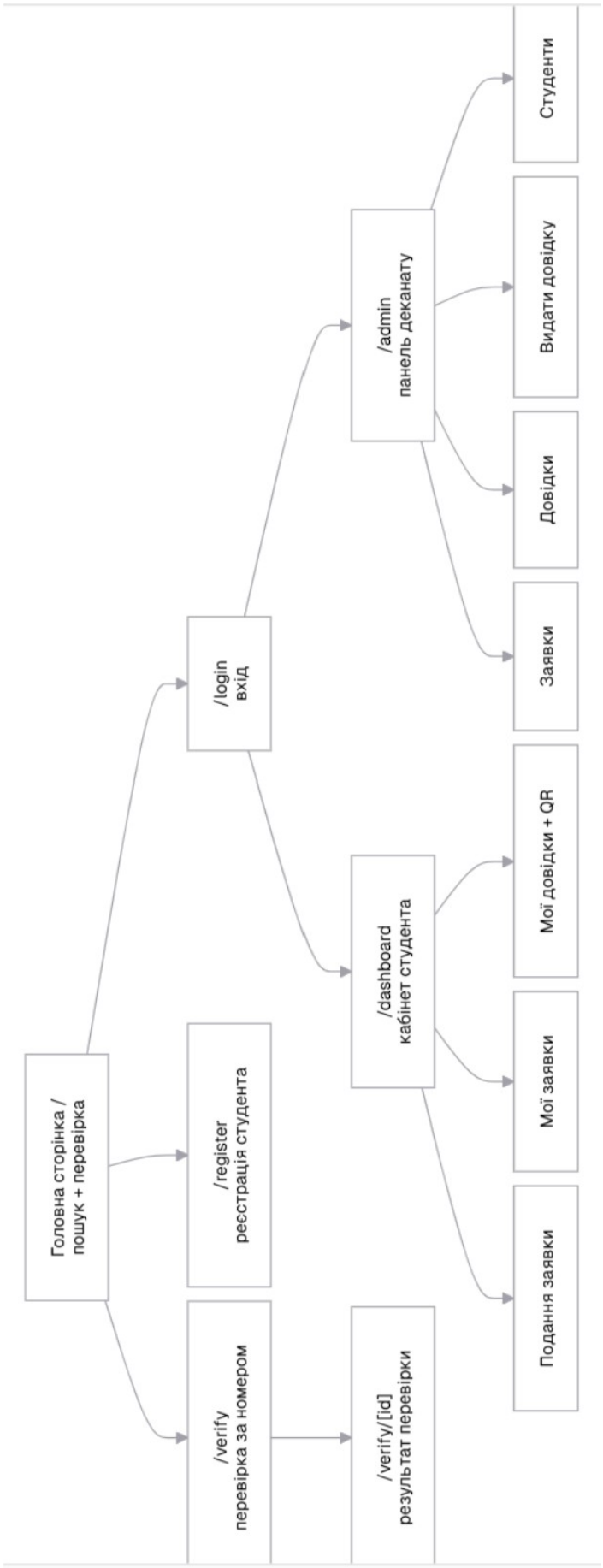



Рисунок 2.5 – Карта сторінок користувацького інтерфейсу системи

Проведений аналіз показує, що система реалізує класичну модель рольового доступу (Role-Based Access Control), де кожна роль має чітко визначений набір дозволених дій. Це дозволяє уникнути дублювання функцій та підвищує рівень безпеки.

Окремо слід відзначити механізм перенаправлення користувачів після авторизації: студенти автоматично потрапляють до особистого кабінету, тоді як адміністратори – до панелі деканату. У випадку спроби доступу до заборонених сторінок система виконує редирект на сторінку входу.

Після визначення структури інтерфейсу важливим етапом є вибір технологічного стеку, який забезпечить реалізацію всіх функціональних вимог системи, включно з продуктивністю, масштабованістю та зручністю розробки (табл. 2.2).

Таблиця 2.2 – Обґрунтування вибору технологій реалізації

| **Рівень системи** | **Технологія** | **Обґрунтування вибору** |
|---|---|---|
| Фронтенд і бекенд | Next.js (App Router) [17] | Забезпечує гібридний підхід SSR/CSR, вбудовані API та серверні дії |
| Мова програмування | TypeScript [18] | Підвищує надійність коду завдяки статичній типізації |
| UI-фреймворк | React [19] | Компонентна архітектура та повторне використання UI-елементів |
| Стилізація | Tailwind CSS [20] + shadcn/ui [21] | Швидке створення адаптивного інтерфейсу з єдиним стилістичним стандартом |
| База даних | SQLite (node:sqlite) [22] | Легка інтеграція без окремого серверу бази даних |
| Безпека | crypto (HMAC-SHA256, scrypt) [23] | Захист паролів та забезпечення цілісності документів |
| QR-генерація | Qrcode [24] | Автоматичне створення QR-кодів для перевірки документів |

Використаний стек технологій дозволяє реалізувати систему як єдиний монолітний вебзастосунок із мінімальною кількістю зовнішніх залежностей. Це значно спрощує розгортання та супровід системи, що є важливим для освітнього середовища.

Проєктування користувацького інтерфейсу базується на принципах простоти, мінімалізму та функціональної зрозумілості. Інтерфейс не

перевантажується зайвими елементами, а всі дії користувача логічно групуються за сценаріями використання.

Особлива увага приділяється адаптивності інтерфейсу, оскільки значна частина взаємодії з системою відбувається через мобільні пристрої під час сканування QR-кодів. Також враховано вимоги доступності, зокрема використання семантичної HTML-структури та достатнього контрасту елементів.

Таким чином, запропонований підхід до проєктування інтерфейсу забезпечує баланс між зручністю використання, безпекою та технічною ефективністю реалізації системи.

# 3 РОЗРОБКА ТА ТЕСТУВАННЯ СИСТЕМИ АВТОМАТИЗАЦІЇ ВИДАЧІ ДОВІДОК СТУДЕНТАМ

## 3.1 Реалізація програмного забезпечення системи

Реалізація програмного забезпечення розроблюваної системи здійснюється як завершальний етап переходу від концептуального та логічного проєктування до безпосереднього створення функціонального вебзастосунку. Основна увага на цьому етапі приділяється практичному втіленню визначеної архітектури, забезпеченню взаємодії між модулями системи та реалізації ключових бізнес-процесів, пов’язаних з обліком студентських довідок, їх видачею та перевіркою автентичності.

Система реалізована з урахуванням принципів модульності, розділення відповідальності та повторного використання компонентів, що дозволяє підтримувати її масштабованість та спрощує подальший супровід.

Структура програмного проєкту побудована відповідно до можливостей фреймворку Next.js із використанням App Router, який дозволяє поєднувати серверну та клієнтську логіку в межах єдиної архітектури. Такий підхід забезпечує логічне розділення відповідальності між різними рівнями системи: користувацьким інтерфейсом, бізнес-логікою та доступом до даних.

Загальна структура проєкту відображає принцип шаруватої архітектури, де кожен каталог виконує чітко визначену роль у функціонуванні системи (рис. 3.1).

Побудована структура демонструє чітке розділення рівнів системи. Каталог app/ відповідає за маршрутизацію та сторінки застосунку, реалізуючи користувацькі сценарії взаємодії. Каталог components/ містить повторно використовувані інтерфейсні елементи, які формують візуальну частину системи та забезпечують єдину стилістичну узгодженість.

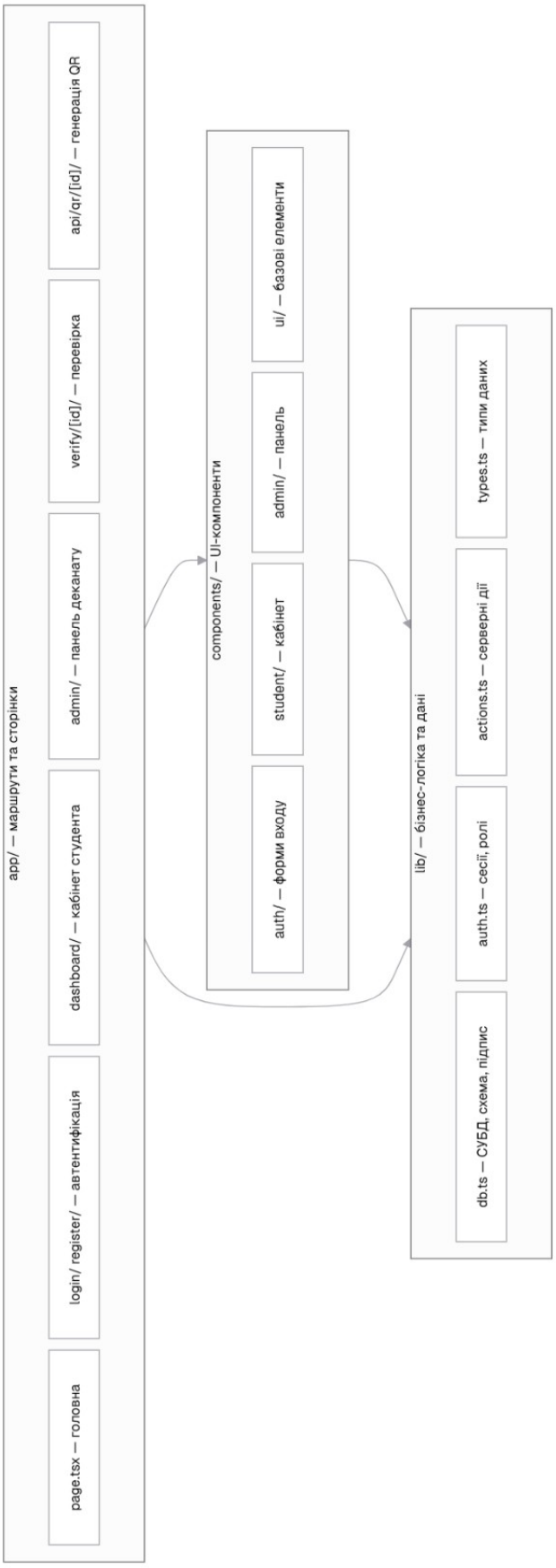



Рисунок 3.1 – Структура програмного проєкту системи

Окремо виділений каталог lib/ виконує роль ядра бізнес-логіки. Саме тут реалізуються механізми роботи з базою даних, автентифікації користувачів, а також серверні операції, які забезпечують основний функціонал системи. Така організація коду дозволяє уникнути змішування логіки інтерфейсу та обробки даних, що підвищує читабельність та підтримуваність проєкту.

Подальша реалізація системи базується на наборі ключових модулів, кожен з яких відповідає за окрему функціональну підсистему (табл. 3.1). Такий підхід дозволяє ізолювати логіку та спростити тестування й масштабування.

Таблиця 3.1 – Основні програмні модулі системи

| **Модуль** | **Файл** | **Призначення** |
|---|---|---|
| Доступ до даних | lib/db.ts | Ініціалізація SQLite, створення таблиць, робота з підписами |
| Автентифікація | lib/auth.ts | Реалізація входу, виходу, перевірки сесій та ролей |
| Серверні дії | lib/actions.ts | Обробка заявок, видача та відкликання довідок |
| Типи даних | lib/types.ts | Опис структур Student, User, Certificate, Request |
| UI-компоненти | components/* | Форми, таблиці, сторінки кабінетів користувачів |

Аналіз наведених модулів показує, що система побудована за принципом чіткого розмежування відповідальності. Це означає, що кожен модуль виконує лише одну групу пов’язаних задач, що значно спрощує подальший розвиток системи та зменшує ризик виникнення помилок при масштабуванні.

Окрему увагу в процесі реалізації приділено організації доступу до бази даних. Система використовує вбудовану СУБД SQLite, що дозволяє уникнути необхідності підняття окремого серверного середовища та значно спрощує розгортання застосунку.

З’єднання з базою даних реалізовано у вигляді єдиного екземпляра (singleton), що гарантує повторне використання підключення протягом усього життєвого циклу застосунку.

```
import { DatabaseSync } from "node:sqlite"
let db: DatabaseSync | null = null
export function getDb(): DatabaseSync {
```

```
  if (db) return db
  const database = new DatabaseSync(DB_PATH)
  database.exec("PRAGMA foreign_keys = ON;")
  createSchema(database)
  seedIfEmpty(database)
  db = database
  return db
}
```

Увімкнення режиму foreign_keys забезпечує контроль цілісності даних між таблицями, що є критично важливим для системи обліку довідок, оскільки дозволяє уникати порушення зв’язків між студентами, заявками та виданими документами.

Крім того, автоматичне створення схеми бази даних при першому запуску дозволяє значно спростити процес розгортання системи, оскільки відсутня необхідність у ручному налаштуванні структури БД.

У процесі реалізації програмного забезпечення особлива увага приділялася безпеці та цілісності даних. Зокрема, використання серверних дій Next.js дозволило мінімізувати кількість клієнтських запитів до бази даних, а також приховати бізнес-логіку від користувача.

Додатково застосовано механізми хешування паролів та цифрового підпису документів, що забезпечує захист як користувацьких даних, так і сформованих довідок. Це дозволяє гарантувати неможливість несанкціонованої модифікації документів та підвищує загальний рівень довіри до системи.

Таким чином, реалізована програмна система є цілісним вебзастосунком, у якому поєднано сучасні підходи до архітектури, безпеки та організації коду, що забезпечує її ефективність та практичну придатність для використання в закладах вищої освіти.

## 3.2 Розробка функціоналу формування, зберігання та видачі електронних довідок

Розробка даного функціонального модуля є центральною складовою програмної системи, оскільки саме він забезпечує повний цикл роботи з електронними довідками: від ініціації запиту студентом до генерації, підписання, зберігання та подальшої перевірки документа.

Загальна логіка побудована відповідно до принципів серверно-орієнтованої архітектури, де всі критичні операції (створення, зміна статусів, підписання) виконуються на стороні сервера. Це дозволяє забезпечити цілісність даних, захист від підробки та контрольований доступ до інформації.

Життєвий цикл електронної довідки відображає логічну модель переходів між станами документа в межах системи. Дана модель реалізована у вигляді скінченного автомата станів, що забезпечує однозначність обробки кожного документа (рис. 3.2).

Початковим етапом є створення заявки студентом. Після цього заявка передається на розгляд адміністратора, який приймає рішення щодо її підтвердження або відхилення. У випадку підтвердження система автоматично формує електронну довідку зі статусом «Активна». Далі документ може бути або відкликаний уповноваженим користувачем, або автоматично переведений у стан «Прострочена» після завершення терміну дії.

Таким чином, запропонована модель забезпечує формалізовану та контрольовану логіку життєвого циклу документа, що виключає некоректні переходи між станами.

Однією з ключових вимог до електронної довідки є її унікальність та можливість перевірки автентичності. Для цього у системі реалізовано механізм генерації публічного ідентифікатора та криптографічного підпису. Відповідний програмний код наведено у Додатку А.1.




Рисунок 3.2 – Діаграма життєвого циклу електронної довідки

У межах реалізації використовується функція генерації ідентифікатора формату CERT-XXXX-XXXX-XXXX, що формує людинозрозумілий, але водночас статистично унікальний код за допомогою криптографічно стійкого генератора випадкових чисел. Це забезпечує неможливість передбачення або підбору ідентифікаторів сторонніми користувачами.

Паралельно з цим формується підпис документа. Для цього створюється структурований набір ключових параметрів довідки, який включає публічний

ідентифікатор, ПІБ студента, номер студентського квитка, тип довідки та дату її видачі. Ці дані об’єднуються в єдину структуру, після чого серіалізуються у формат JSON і передаються до криптографічної функції HMAC-SHA256 із використанням секретного ключа сервера. У результаті формується цифровий підпис, який гарантує цілісність документа та дозволяє виявити будь-які зміни у його вмісті.

Таким чином, описаний механізм забезпечує одразу дві функції: генерацію унікального ідентифікатора для пошуку довідки та створення захищеного цифрового підпису для перевірки її достовірності.

Безпосередня видача електронної довідки реалізована у вигляді серверної дії, яка виконується виключно користувачем із роллю адміністратора. Відповідний програмний код наведено у Додатку А.2.

Після ініціації операції система спочатку здійснює перевірку прав доступу, що виключає можливість несанкціонованого створення документів. Далі відбувається отримання даних студента з бази даних, на основі яких формується новий запис довідки. На цьому етапі генерується унікальний ідентифікатор документа, а також визначається точний час його створення.

Наступним кроком є формування криптографічного підпису, який створюється на основі ключових атрибутів довідки. Це дозволяє забезпечити автентичність документа ще до моменту його збереження в базі даних. Після генерації підпису запис про довідку додається до таблиці certificates, де фіксується її статус як активної.

Важливою особливістю реалізації є використання серверного рендерінгу та функції оновлення маршруту після завершення операції, що забезпечує синхронізацію стану інтерфейсу адміністратора з фактичним станом бази даних.

Функціональність системи також передбачає механізм обробки заявок студентів, що є проміжним етапом між ініціацією запиту та видачею довідки. Відповідний код реалізації наведено у Додатку А.2.

Студент формує заявку, яка зберігається у базі даних зі статусом «pending». Далі адміністратор переглядає список заявок і приймає рішення щодо кожної з

них. У разі схвалення система автоматично ініціює процес генерації електронної довідки, після чого заявка отримує статус «approved» та зв’язується з відповідним документом через публічний ідентифікатор довідки.

Якщо ж заявка відхиляється, її статус змінюється на «rejected», а в системі може фіксуватися додатковий коментар адміністратора. Така модель забезпечує прозорість процесу та дозволяє відстежувати повний шлях обробки кожного запиту.

З метою забезпечення безпеки та конфіденційності реалізовано механізм розмежованого доступу до даних. Кожен користувач системи отримує доступ лише до тієї інформації, яка відповідає його ролі та обліковому запису.

Зокрема, студенти мають можливість переглядати лише власні довідки та заявки, що реалізується через фільтрацію SQL-запитів за ідентифікатором користувача поточної сесії. Адміністратори, у свою чергу, мають повний доступ до всіх записів системи, включаючи можливість створення, редагування та відкликання довідок.

Окремо слід відзначити, що система використовує централізований механізм автентифікації, який базується на сесіях та токенах доступу. Це дозволяє забезпечити захищену взаємодію між клієнтською та серверною частинами застосунку, а також унеможливлює доступ до даних без проходження процедури авторизації.

У підсумку слід зазначити, що реалізований функціонал формування, зберігання та видачі електронних довідок забезпечує повний життєвий цикл документа, включаючи його створення, автентифікацію, обробку запитів, контроль доступу та перевірку достовірності, що відповідає сучасним вимогам до електронних систем документообігу у закладах вищої освіти.

## 3.3 Реалізація QR-верифікації та онлайн перевірки автентичності документів

Реалізація підсистеми QR-верифікації та онлайн перевірки автентичності електронних довідок є важливим етапом забезпечення довіри до цифрових документів у межах розробленої інформаційної системи. Даний функціональний модуль забезпечує можливість швидкої ідентифікації довідки через QR-код, а також її подальшої перевірки в режимі реального часу з використанням криптографічних механізмів захисту. Загальна архітектура підсистеми побудована за принципом клієнт-серверної взаємодії, де серверна частина відповідає за генерацію коду та перевірку даних, а клієнтська – за відображення результатів верифікації.

Формування QR-коду у системі реалізовано як серверний API-маршрут /api/qr/[id], що забезпечує динамічну генерацію зображення на основі унікального ідентифікатора довідки. Відповідний програмний код наведено у Додатку А.3.

У межах реалізації спочатку визначається базова URL-адреса системи, після чого до неї додається маршрут сторінки перевірки довідки у форматі /verify/{id}. Таким чином формується повний унікальний лінк, який містить ідентифікатор конкретного документа.

Далі цей URL передається до бібліотеки генерації QR-кодів, яка перетворює текстове представлення адреси у графічне зображення у форматі PNG. При цьому додатково задаються параметри візуалізації, зокрема розмір зображення, відступи та кольорова схема, що забезпечує коректне сканування QR-коду будь-якими стандартними мобільними пристроями.

Результатом виконання API-запиту є бінарне зображення, яке повертається клієнту із заголовком Content-Type: image/png та параметрами кешування. Це дозволяє зменшити навантаження на сервер і прискорити повторне завантаження QR-кодів у межах інтерфейсу користувача.

Процес онлайн-верифікації електронної довідки реалізовано як послідовність взаємодій між користувачем, сервером застосунку та базою даних. Узагальнена схема цього процесу наведена у вигляді діаграми послідовності (рис. 3.3).

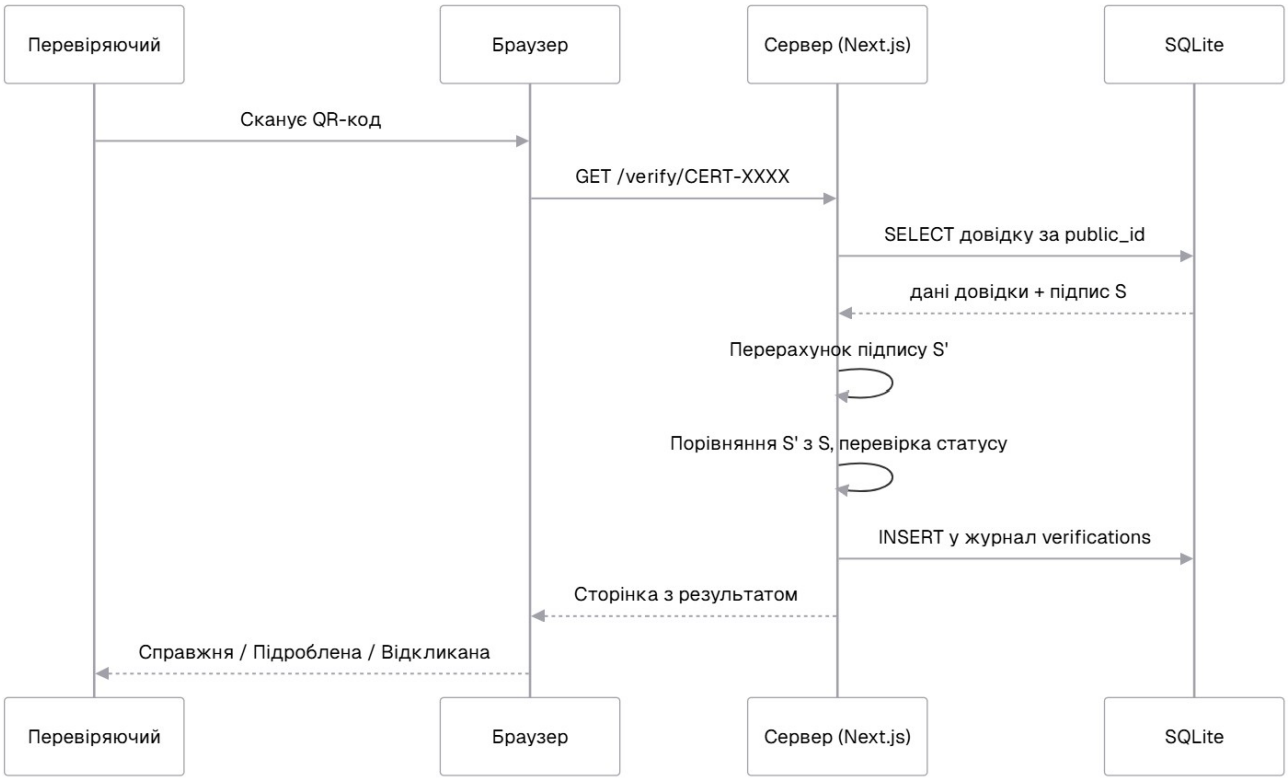



Рисунок 3.3 – Послідовність взаємодії під час перевірки

У типовому сценарії перевіряюча особа сканує QR-код за допомогою мобільного пристрою, після чого автоматично відкривається сторінка перевірки, що містить унікальний ідентифікатор довідки. Цей ідентифікатор передається на сервер, де виконується запит до бази даних для отримання повного запису про відповідну довідку.

Після отримання даних система здійснює перевірку цифрового підпису, порівнюючи збережене значення з перерахованим на основі актуальних параметрів документа. У разі успішного проходження перевірки додатково виконується контроль статусу довідки (активна, відкликана або прострочена).

На завершальному етапі результат перевірки записується до журналу верифікацій, після чого користувач отримує відповідний статус документа:

«чинна», «відкликана», «прострочена» або «не знайдена». Таким чином забезпечується повна прозорість процесу перевірки та можливість аудиту кожного запиту.

Центральним елементом підсистеми є алгоритм криптографічної перевірки автентичності документа. Його реалізація наведена у Додатку А.2.

Алгоритм базується на принципі повторного обчислення цифрового підпису за допомогою HMAC-SHA256 із використанням секретного ключа сервера. Після отримання запису довідки з бази даних формується структура даних, яка включає ключові атрибути документа: унікальний ідентифікатор, ПІБ студента, номер студентського квитка, тип довідки та дату її видачі. Ці дані формують вхідне повідомлення для генерації контрольного підпису.

Отримане значення порівнюється з підписом, який збережений у базі даних на момент створення довідки. У випадку повної відповідності документ вважається автентичним. Якщо ж значення не збігаються, система фіксує факт модифікації даних і позначає довідку як потенційно підроблену.

Додатково алгоритм виконує перевірку статусу документа. У разі якщо довідка була відкликана адміністрацією або втратила чинність через закінчення терміну дії, система повертає відповідний негативний результат навіть за умови коректності підпису.

Рівень безпеки реалізованої підсистеми базується на поєднанні криптографічного захисту, унікальної ідентифікації документів та контролю статусів у базі даних. Узагальнення основних загроз і відповідних механізмів захисту наведено в таблиці 3.2.

Завдяки такій багаторівневій системі захисту забезпечується висока надійність перевірки електронних довідок, що унеможливлює їх несанкціоноване копіювання або підробку без зміни криптографічного сліду.

Також інтерфейс розробленого вебзастосунку наведено в додатку Б.

Таблиця 3.2 – Механізми забезпечення безпеки QR-верифікації

| **Потенційна загроза** | **Реалізований механізм захисту** | **Результат обробки** |
|---|---|---|
| Підміна даних довідки | Перерахунок HMAC-SHA256 підпису | Виявлення стану «підроблено» |
| Використання неіснуючого ідентифікатора | Перевірка наявності запису в БД | Статус «не знайдено» |
| Використання анульованої довідки | Контроль поля status | Статус «відкликана» |
| Використання простроченого документа | Перевірка expires_at | Статус «прострочена» |
| Спроби криптопідбору ключа | Використання HMAC-SHA256 (256-bit) | Практична неможливість злому |

У результаті реалізації підсистеми QR-верифікації було створено механізм швидкої та надійної перевірки електронних довідок, який поєднує зручність використання для кінцевого користувача та високий рівень криптографічного захисту. Використання QR-кодів забезпечує миттєвий доступ до сторінки перевірки, а алгоритм цифрового підпису гарантує цілісність і достовірність даних. Сукупність цих підходів дозволяє ефективно протидіяти підробці документів та забезпечує довіру до електронної системи документообігу у цілому.

## 3.4 Тестування системи та аналіз результатів роботи

У процесі розробки вебзастосунку було проведено комплексне тестування з метою перевірки коректності функціонування всіх модулів системи, відповідності реалізованого функціоналу вимогам технічного завдання, а також оцінювання стабільності та безпеки роботи програмного забезпечення. Тестування виконувалося за принципом методу «чорної скриньки», що дозволило зосередитися на перевірці поведінки системи без аналізу внутрішньої реалізації коду.

Основний акцент було зроблено на моделюванні реальних сценаріїв використання системи різними категоріями користувачів, зокрема студентами та

адміністраторами. Для автоматизації частини перевірок використовувався браузерний агент, що дозволило відтворювати типові дії користувача у вебінтерфейсі, тоді як критичні безпекові сценарії додатково перевірялися вручну для виключення можливих вразливостей.

Функціональне тестування охоплювало перевірку процесів реєстрації та авторизації користувачів, створення та обробки заявок на видачу довідок, формування електронних документів, а також коректність роботи механізму їх перевірки. Особливу увагу було приділено сценаріям доступу з різними ролями, що дозволило підтвердити правильність реалізації системи розмежування прав доступу.

У результаті виконання тестових сценаріїв було підтверджено коректну роботу основних бізнес-процесів системи. Зокрема, користувачі успішно проходили процес реєстрації та входу в систему з подальшим перенаправленням до відповідних інтерфейсів залежно від ролі. Студентська частина системи коректно забезпечувала створення заявок на отримання довідок та відображення їх актуального статусу, тоді як адміністративна частина дозволяла виконувати перевірку, підтвердження або відхилення заявок, а також здійснювати пряму видачу документів.

Окремо було протестовано сценарії обробки помилкових дій користувача, зокрема спроби входу з неправильними обліковими даними, доступу до адміністративної панелі без відповідних прав, а також перевірки неіснуючих або некоректних даних довідок. У всіх випадках система коректно реагувала шляхом відхилення запиту та відображення відповідного повідомлення або перенаправлення користувача.

Додатково проведено тестування механізму верифікації електронних довідок, що включало перевірку справжніх документів, відкликаних записів, а також випадків підробки даних. Результати підтвердили, що система коректно ідентифікує зміну даних завдяки перевірці криптографічного підпису, що дозволяє виявляти будь-які спроби модифікації інформації.

Аналіз результатів тестування засвідчив повну відповідність реалізованої системи функціональним вимогам. Усі заплановані сценарії були успішно виконані, а виявлені на етапі розробки технічні недоліки були усунуті. Система продемонструвала стабільну роботу, коректне розмежування доступу між користувачами та адміністраторами, а також надійність механізму перевірки автентичності документів, що є критично важливим для даного типу програмного забезпечення.

### 3.5 Інструкція користувача

Розроблена система є веборієнтованим застосунком, доступ до якого здійснюється через стандартний браузер без необхідності встановлення додаткового програмного забезпечення. Архітектура системи передбачає три основні режими роботи: гостьовий режим, призначений для перевірки автентичності документів; режим студента, який забезпечує подання та відстеження заявок; а також режим адміністратора, який використовується для управління довідками та студентськими запитами.

Для студентів передбачено послідовний сценарій роботи, що починається з реєстрації облікового запису із зазначенням персональних даних, таких як електронна пошта, пароль, прізвище та ім’я, а також номер студентського квитка. Після успішної авторизації користувач отримує доступ до особистого кабінету, де реалізовано функціонал подання заявки на отримання довідки. У процесі створення заявки студент обирає тип необхідного документа та вказує його призначення, після чого заявка надсилається на розгляд адміністратора.

Подальша взаємодія користувача полягає у відстеженні статусу поданих заявок, який може змінюватися залежно від етапу обробки. Після підтвердження заявки система автоматично формує електронну довідку, яка стає доступною у відповідному розділі особистого кабінету. Кожен документ супроводжується

унікальним QR-кодом, що дозволяє здійснювати швидку перевірку його автентичності.

Для адміністратора передбачено розширений функціонал управління системою, доступний через адміністративну панель. Основними завданнями адміністратора є перегляд та обробка студентських заявок, їх підтвердження або відхилення, а також можливість прямої видачі довідок без попередньої заявки. Після підтвердження заявки система автоматично формує електронний документ із цифровим підписом і QR-кодом, що забезпечує подальшу можливість перевірки.

Окремий режим передбачено для перевірки документів. Користувач може відсканувати QR-код або ввести унікальний ідентифікатор довідки на сторінці перевірки. У результаті система здійснює запит до бази даних, перевіряє наявність документа, його статус, а також цілісність цифрового підпису. Результат перевірки відображається у вигляді зрозумілого статусу, що інформує користувача про справжність, відкликання або відсутність документа в системі.

Таким чином, реалізована інструкція користувача забезпечує інтуїтивно зрозумілу послідовність дій для кожної категорії користувачів, що сприяє ефективному використанню системи без необхідності додаткового навчання.

# ВИСНОВКИ

У ході виконання кваліфікаційної роботи було здійснено комплексне дослідження, проєктування, реалізацію та тестування вебсистеми автоматизації видачі студентських довідок із використанням механізму QR-верифікації та онлайн перевірки автентичності документів.

На першому етапі роботи було проаналізовано теоретичні основи електронного документообігу в закладах освіти. Розглянуто сутність, призначення та особливості функціонування систем електронного документообігу, визначено їх роль у підвищенні ефективності управлінських процесів у навчальних закладах. Окрему увагу приділено аналізу сучасних рішень автоматизації видачі довідок та цифрової верифікації документів, що дозволило виявити їхні переваги та недоліки, зокрема обмежену гнучкість, відсутність уніфікованих механізмів перевірки та недостатній рівень захисту від підробки. На основі проведеного аналізу сформульовано вимоги до розроблюваної системи, що охоплюють функціональні, безпекові та користувацькі аспекти.

На другому етапі було виконано проєктування системи обліку та автоматизації видачі студентських довідок. Здійснено аналіз предметної області та сформульовано постановку задачі, що передбачає створення вебзастосунку для централізованого управління процесами подання заявок, видачі електронних документів і їх подальшої перевірки. Розроблено архітектуру системи та структуру бази даних, яка забезпечує зберігання інформації про студентів, користувачів, заявки, довідки та результати верифікації. Окремо спроєктовано механізм QR-верифікації, який дозволяє здійснювати швидку перевірку автентичності документа через унікальне посилання на сторінку перевірки, а також визначено підхід до формування цифрового підпису як засобу захисту цілісності даних. Також було розроблено інтерфейси користувача та обґрунтовано вибір сучасного технологічного стеку для реалізації системи.

На третьому етапі здійснено програмну реалізацію системи та її основних функціональних модулів. Реалізовано механізми автентифікації користувачів, управління ролями, подання та обробки заявок, а також формування електронних довідок. Особливу увагу приділено реалізації механізмів генерації унікального ідентифікатора документа, формування цифрового підпису та збереження даних у базі. Додатково реалізовано QR-верифікацію та онлайн перевірку автентичності документів, що забезпечує можливість миттєвого контролю справжності довідки через вебінтерфейс.

У межах тестування системи було перевірено коректність роботи всіх основних функціональних модулів, включаючи сценарії реєстрації, авторизації, створення та обробки заявок, видачі довідок і їх верифікації. Результати тестування підтвердили стабільну роботу системи та відповідність усіх реалізованих функцій поставленим вимогам. Окремо підтверджено ефективність механізму криптографічного захисту, який забезпечує виявлення будь-яких спроб модифікації даних у довідках. Також було розроблено інструкцію користувача, що забезпечує зрозумілий алгоритм дій для всіх категорій користувачів системи.

Отже, у результаті виконання кваліфікаційної роботи було створено повноцінну вебсистему автоматизації видачі студентських довідок із підтримкою QR-верифікації та криптографічного захисту даних. Реалізоване програмне забезпечення забезпечує ефективну автоматизацію документообігу, підвищує рівень безпеки зберігання та перевірки документів, а також спрощує взаємодію між студентами та адміністрацією закладу освіти.

# ПЕРЕЛІК ПОСИЛАНЬ

1. Електронний документообіг у закладах освіти. URL : https://arts.gov.ua/elektronnyj-dokumentoobig-u-zakladah-osvity/ (дата звернення : 29.03.2026).

2. Про електронні документи та електронний документообіг : Закон України від 22.05.2003 № 851-IV : станом на 01.01.2024. URL : https://zakon.rada.gov.ua/laws/show/851-15#Text (дата звернення : 29.03.2026).

3. Величкевич М. Б., Мітрофан Н. В., Кунанець Н. Е. Електронний документообіг : тенденції та перспективи. *Academic Journals and Conferences SISN*. 2010. Т. 689, № 2. С. 44–53.

4. Гарбич-Мошора О. Електронний документообіг у закладі вищої освіти: тенденції та перспективи. *Молодь і ринок*. 2018. № 9 (164). DOI : https://doi.org/10.24919/2308-4634.2018.144290.

5. Про затвердження Порядку роботи з електронними документами у діловодстві та їх підготовки до передавання на архівне зберігання : наказ Міністерства юстиції України від 11.11.2014 № 1886/5 : станом на 01.04.2026. URL : https://zakon.rada.gov.ua/laws/show/z1421-14#Text (дата звернення : 29.03.2026).

6. Білик О. Сучасні підходи до впровадження електронного документообігу у систему державного управління. *Науковий вісник Вінницької академії безперервної освіти. Серія «Екологія, публічне управління та адміністрування»*. 2023. № 4. С. 38–45. DOI : https://doi.org/10.32782/2786-5681-2023-4.06.

7. ЄДЕБО – Єдина державна електронна база з питань освіти : офіційний сайт. URL : https://info.edbo.gov.ua/about (дата звернення : 29.03.2026).

8. Програмний комплекс «Автоматизована система управління навчальним закладом» : ЗНУ. URL : https://sites.znu.edu.ua/navchalnyj_viddil/

normatyvna_basa/timchasovij_poryadok_roboti_z_asu_mkr.pdf (дата звернення : 29.03.2026).

9. Академічна довідка : зразок, порядок оформлення та видачі. Блог Магістр. URL : https://blog.magistr.ua/akademdovidka/ (дата звернення : 29.03.2026).

10. Дія.Підпис : офіційна сторінка інтеграції. URL : https://integration.diia.gov.ua/signature.html (дата звернення : 29.03.2026).

11. Your 2025 Guide to Document Authentication QR Code. QR Mark. URL : https://qrmark.com/blog/document-authentication-qr-code/ (дата звернення : 29.03.2026).

12. Diplomas. University Registrar, Johns Hopkins University. URL : https://registrar.jhu.edu/academic-records/diplomas/ (дата звернення : 29.03.2026).

13. Gangwar S., Chaurasia A. Blockchain-based Authentication and Verification System for Academic Certificate using QR Code and Decentralized Applications. *International Journal of Computer Applications*. 2024. Vol. 186, No. 26. URL : https://www.ijcaonline.org/archives/volume186/number26/gangwar-2024-ijca-923722.pdf (дата звернення : 29.03.2026).

14. ZKBAR-V : A Zero-Knowledge Proof-Enabled Blockchain-Based Academic Record Verification System. *Sensors*. 2025. Vol. 25, No. 11. DOI : https://doi.org/10.3390/s25113450.

15. Educational Certificate Authentication Using QR Code. Qryptal. URL : https://www.qryptal.com/industries/education/ (дата звернення : 29.03.2026).

16. Smart Credentialing and Verification System for National Certificates using Blockchain Technology. *Proceedings of ICDTE 2024. ACM Digital Library*. DOI : https://doi.org/10.1145/3696230.3696266.

17. Next.js – the React framework. URL : https://nextjs.org/ (дата звернення : 29.03.2026).

18. TypeScript : JavaScript with Syntax for Types. URL : https://www.typescriptlang.org/ (дата звернення : 29.03.2026).

19. React. URL : https://react.dev/ (дата звернення : 29.03.2026).

20. Tailwind CSS Documentation. URL : https://tailwindcss.com/ (дата звернення : 29.03.2026).

21. shadcn/ui Documentation. URL : https://ui.shadcn.com/ (дата звернення : 29.03.2026).

22. SQLite Documentation. URL : https://sqlite.org/docs.html (дата звернення : 29.03.2026).

23. HMAC-SHA256 vs scrypt | Compare Top Cryptographic Hashing Algorithms. MojoAuth. URL : https://mojoauth.com/compare-hashing-algorithms/hmac-sha256-vs-scrypt (дата звернення : 29.03.2026).

24. qrcode. npm. URL : https://www.npmjs.com/package/qrcode (дата звернення : 29.03.2026).

# ДОДАТОК А

## Лістинг функціоналу формування, зберігання та видачі електронних довідок

**Лістинг А.1 Код бази даних**

```
import { DatabaseSync } from "node:sqlite"
import path from "node:path"
import fs from "node:fs"
import crypto from "node:crypto"

// Persist the SQLite database file in a writable data directory.
const DATA_DIR = path.join(process.cwd(), ".data")
if (!fs.existsSync(DATA_DIR)) {
  fs.mkdirSync(DATA_DIR, { recursive: true })
}
const DB_PATH = path.join(DATA_DIR, "certificates.db")

// Secret used to sign certificate payloads (HMAC) for authenticity checks.
const SIGNING_SECRET = process.env.CERT_SIGNING_SECRET || "v0-dev-cert-signing-secret-change-me"

// Use a singleton across hot reloads in development.
const globalForDb = globalThis as unknown as { __certDb?: DatabaseSync }

function createDb(): DatabaseSync {
  const db = new DatabaseSync(DB_PATH)
```

```
db.exec("PRAGMA journal_mode = WAL;")
db.exec("PRAGMA foreign_keys = ON;")

db.exec(`
  CREATE TABLE IF NOT EXISTS students (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    full_name TEXT NOT NULL,
    student_number TEXT NOT NULL UNIQUE,
    faculty TEXT NOT NULL,
    group_name TEXT NOT NULL,
    course INTEGER NOT NULL,
    created_at TEXT NOT NULL DEFAULT (datetime('now'))
  );
`)

db.exec(`
  CREATE TABLE IF NOT EXISTS certificates (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    public_id TEXT NOT NULL UNIQUE,
    student_id INTEGER NOT NULL,
    type TEXT NOT NULL,
    purpose TEXT NOT NULL,
    issued_by TEXT NOT NULL,
    issued_at TEXT NOT NULL DEFAULT (datetime('now')),
    expires_at TEXT,
    status TEXT NOT NULL DEFAULT 'active',
    signature TEXT NOT NULL,
    FOREIGN KEY (student_id) REFERENCES students(id) ON DELETE
CASCADE
  );
```

```
`)

db.exec(`
  CREATE TABLE IF NOT EXISTS verifications (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    certificate_id INTEGER,
    public_id TEXT NOT NULL,
    result TEXT NOT NULL,
    verified_at TEXT NOT NULL DEFAULT (datetime('now'))
  );
`)

db.exec(`
  CREATE TABLE IF NOT EXISTS users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    email TEXT NOT NULL UNIQUE,
    password_hash TEXT NOT NULL,
    full_name TEXT NOT NULL,
    role TEXT NOT NULL DEFAULT 'student',
    student_id INTEGER,
    created_at TEXT NOT NULL DEFAULT (datetime('now')),
    FOREIGN KEY (student_id) REFERENCES students(id) ON DELETE
SET NULL
  );
`)

db.exec(`
  CREATE TABLE IF NOT EXISTS sessions (
    token TEXT PRIMARY KEY,
    user_id INTEGER NOT NULL,
```

```
            created_at TEXT NOT NULL DEFAULT (datetime('now')),
            expires_at TEXT NOT NULL,
            FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE
CASCADE
          );
        `)

        db.exec(`
          CREATE TABLE IF NOT EXISTS requests (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            user_id INTEGER NOT NULL,
            student_id INTEGER,
            type TEXT NOT NULL,
            purpose TEXT NOT NULL,
            status TEXT NOT NULL DEFAULT 'pending',
            certificate_public_id TEXT,
            note TEXT,
            created_at TEXT NOT NULL DEFAULT (datetime('now')),
            resolved_at TEXT,
            FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE
CASCADE,
            FOREIGN KEY (student_id) REFERENCES students(id) ON DELETE
SET NULL
          );
        `)

        return db
      }

      // ---- Password hashing (scrypt) ----
```

```
export function hashPassword(password: string): string {
  const salt = crypto.randomBytes(16).toString("hex")
  const derived = crypto.scryptSync(password, salt, 64).toString("hex")
  return `${salt}:${derived}`
}

export function verifyPassword(password: string, stored: string): boolean {
  const [salt, hash] = stored.split(":")
  if (!salt || !hash) return false
  const derived = crypto.scryptSync(password, salt, 64)
  const hashBuf = Buffer.from(hash, "hex")
  if (hashBuf.length !== derived.length) return false
  return crypto.timingSafeEqual(hashBuf, derived)
}

export function generateSessionToken(): string {
  return crypto.randomBytes(32).toString("hex")
}

export function getDb(): DatabaseSync {
  if (!globalForDb.__certDb) {
    globalForDb.__certDb = createDb()
    seedIfEmpty(globalForDb.__certDb)
  }
  return globalForDb.__certDb
}

// ---- Signature helpers (authenticity) ----
```

```
export function signPayload(payload: Record<string, unknown>): string {
  const data = JSON.stringify(payload)
  return crypto.createHmac("sha256",
SIGNING_SECRET).update(data).digest("hex")
}

export function buildSignaturePayload(args: {
  public_id: string
  full_name: string
  student_number: string
  type: string
  issued_at: string
}) {
  return {
    public_id: args.public_id,
    full_name: args.full_name,
    student_number: args.student_number,
    type: args.type,
    issued_at: args.issued_at,
  }
}

export function generatePublicId(): string {
  // Human-readable, hard-to-guess id, e.g. CERT-7F3A-9KQ2-LM84
  const part = () =>
    crypto.randomBytes(2).toString("hex").toUpperCase().replace(/[^0-9A-Z]/g,
"")
  return `CERT-${part()}-${part()}-${part()}`
}
```

```
function seedIfEmpty(db: DatabaseSync) {
  const userRow = db.prepare("SELECT COUNT(*) AS c FROM users").get() as { c: number }
  if (userRow.c === 0) {
    // Default admin / dean account.
    db.prepare(
      "INSERT INTO users (email, password_hash, full_name, role) VALUES (?, ?, ?, 'admin')",
    ).run("admin@univer.edu", hashPassword("admin1234"), "Адміністратор деканату")
  }

  const row = db.prepare("SELECT COUNT(*) AS c FROM students").get() as { c: number }
  if (row.c > 0) return

  const students = [
    { full_name: "Іваненко Олена Петрівна", student_number: "ST-2023-001", faculty: "Факультет інформаційних технологій", group_name: "КН-21", course: 3 },
    { full_name: "Коваленко Андрій Сергійович", student_number: "ST-2022-114", faculty: "Економічний факультет", group_name: "ЕК-19", course: 4 },
    { full_name: "Шевченко Марія Олегівна", student_number: "ST-2024-058", faculty: "Юридичний факультет", group_name: "ПР-12", course: 2 },
  ]

  const insertStudent = db.prepare(
    "INSERT INTO students (full_name, student_number, faculty, group_name, course) VALUES (?, ?, ?, ?, ?)",
  )
```

```
for (const s of students) {
  insertStudent.run(s.full_name, s.student_number, s.faculty, s.group_name,
s.course)
}

// Issue one sample active certificate for the first student.
const student = db
  .prepare("SELECT * FROM students WHERE student_number = ?")
  .get("ST-2023-001") as {
  id: number
  full_name: string
  student_number: string
}

const public_id = "CERT-DEMO-0001-2026"
const issued_at = new Date().toISOString()
const type = "Довідка про навчання"
const signature = signPayload(
  buildSignaturePayload({
    public_id,
    full_name: student.full_name,
    student_number: student.student_number,
    type,
    issued_at,
  }),
)

db.prepare(
  `INSERT INTO certificates
```

```
            (public_id, student_id, type, purpose, issued_by, issued_at, expires_at,
status, signature)
            VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)`,
          ).run(
            public_id,
            student.id,
            type,
            "Для надання за місцем вимоги",
            "Деканат ФІТ",
            issued_at,
            null,
            "active",
            signature,
          )

          // Demo student account linked to the first student.
          db.prepare(
            "INSERT INTO users (email, password_hash, full_name, role, student_id)
VALUES (?, ?, ?, 'student', ?)",
          ).run(
            "student@univer.edu",
            hashPassword("student1234"),
            student.full_name,
            student.id,
          )
        }
```

**Лістинг А.2 Код функціоналу / дій**

```
"use server"

import { revalidatePath } from "next/cache"
import {
  getDb,
  signPayload,
  buildSignaturePayload,
  generatePublicId,
} from "./db"
import type {
  CertificateWithStudent,
  RequestWithUser,
  Student,
  VerificationResult,
} from "./types"
import { getCurrentUser, requireAdmin, requireUser } from "./auth"

const SELECT_CERT_WITH_STUDENT = `
  SELECT
    c.*,
    s.full_name,
    s.student_number,
    s.faculty,
    s.group_name,
    s.course
  FROM certificates c
  JOIN students s ON s.id = c.student_id
`
```

```
// node:sqlite returns rows with a null prototype, which cannot cross the
// Server -> Client Component boundary. Convert each row to a plain object.
function toPlain<T>(rows: unknown[]): T[] {
  return rows.map((r) => ({ ...(r as object) })) as T[]
}

export async function getStudents(): Promise<Student[]> {
  const db = getDb()
  return toPlain<Student>(db.prepare("SELECT * FROM students ORDER BY full_name").all())
}

export async function getCertificates(): Promise<CertificateWithStudent[]> {
  const db = getDb()
  return toPlain<CertificateWithStudent>(
    db.prepare(`${SELECT_CERT_WITH_STUDENT} ORDER BY c.issued_at DESC`).all(),
  )
}

export async function getStats() {
  const db = getDb()
  const total = (db.prepare("SELECT COUNT(*) c FROM certificates").get() as { c: number }).c
  const active = (
    db.prepare("SELECT COUNT(*) c FROM certificates WHERE status = 'active'").get() as {
      c: number
    }
```

```
    ).c
    const revoked = (
      db.prepare("SELECT COUNT(*) c FROM certificates WHERE status =
'revoked'").get() as {
        c: number
      }
    ).c
    const students = (db.prepare("SELECT COUNT(*) c FROM students").get()
as { c: number }).c
    const verifications = (
      db.prepare("SELECT COUNT(*) c FROM verifications").get() as { c:
number }
    ).c
    return { total, active, revoked, students, verifications }
  }

  export async function createStudent(formData: FormData) {
    await requireAdmin()
    const db = getDb()
    const full_name = String(formData.get("full_name") || "").trim()
    const student_number = String(formData.get("student_number") || "").trim()
    const faculty = String(formData.get("faculty") || "").trim()
    const group_name = String(formData.get("group_name") || "").trim()
    const course = Number(formData.get("course") || 1)

    if (!full_name || !student_number || !faculty || !group_name) {
      return { ok: false, error: "Заповніть усі поля" }
    }

    try {
```

```
        db.prepare(
          "INSERT INTO students (full_name, student_number, faculty,
group_name, course) VALUES (?, ?, ?, ?, ?)",
        ).run(full_name, student_number, faculty, group_name, course)
      } catch {
        return { ok: false, error: "Студент із таким номером уже існує" }
      }

      revalidatePath("/admin")
      return { ok: true }
    }

    export async function issueCertificate(formData: FormData) {
      await requireAdmin()
      const db = getDb()
      const student_id = Number(formData.get("student_id"))
      const type = String(formData.get("type") || "").trim()
      const purpose = String(formData.get("purpose") || "").trim()
      const issued_by = String(formData.get("issued_by") || "").trim()
      const expires_at = String(formData.get("expires_at") || "").trim() || null

      if (!student_id || !type || !issued_by) {
        return { ok: false, error: "Оберіть студента, тип та хто видав" }
      }

      const student = db.prepare("SELECT * FROM students WHERE id
= ?").get(student_id) as
        | Student
        | undefined
      if (!student) return { ok: false, error: "Студента не знайдено" }
```

```
    const public_id = generatePublicId()
    const issued_at = new Date().toISOString()
    const signature = signPayload(
      buildSignaturePayload({
        public_id,
        full_name: student.full_name,
        student_number: student.student_number,
        type,
        issued_at,
      }),
    )

    db.prepare(
      `INSERT INTO certificates
        (public_id, student_id, type, purpose, issued_by, issued_at, expires_at,
status, signature)
       VALUES (?, ?, ?, ?, ?, ?, ?, 'active', ?)`,
    ).run(public_id, student_id, type, purpose, issued_by, issued_at, expires_at,
signature)

    revalidatePath("/admin")
    return { ok: true, public_id }
  }

  export async function revokeCertificate(publicId: string) {
    await requireAdmin()
    const db = getDb()
    db.prepare("UPDATE certificates SET status = 'revoked' WHERE public_id
= ?").run(publicId)
```

```
  revalidatePath("/admin")
  return { ok: true }
 }

 export async function restoreCertificate(publicId: string) {
  await requireAdmin()
  const db = getDb()
  db.prepare("UPDATE certificates SET status = 'active' WHERE public_id
= ?").run(publicId)
  revalidatePath("/admin")
  return { ok: true }
 }

 // Core authenticity check used by the public verification page.
 export async function verifyCertificate(publicId: string):
Promise<VerificationResult> {
  const db = getDb()
  const id = publicId.trim().toUpperCase()

  const certRow = db
   .prepare(`${SELECT_CERT_WITH_STUDENT} WHERE c.public_id = ?
`)
   .get(id) as CertificateWithStudent | undefined
  const cert = certRow ? ({ ...certRow } as CertificateWithStudent) : undefined

  const logResult = (result: string, certificate_id: number | null) => {
   db.prepare(
    "INSERT INTO verifications (certificate_id, public_id, result) VALUES
(?, ?, ?)",
   ).run(certificate_id, id, result)
```

```
}

if (!cert) {
  logResult("not_found", null)
  return { valid: false, reason: "not_found" }
}

// 1. Verify cryptographic signature (detect tampering).
const expected = signPayload(
  buildSignaturePayload({
    public_id: cert.public_id,
    full_name: cert.full_name,
    student_number: cert.student_number,
    type: cert.type,
    issued_at: cert.issued_at,
  }),
)
if (expected !== cert.signature) {
  logResult("tampered", cert.id)
  return { valid: false, reason: "tampered", certificate: cert }
}

// 2. Check revocation.
if (cert.status === "revoked") {
  logResult("revoked", cert.id)
  return { valid: false, reason: "revoked", certificate: cert }
}

// 3. Check expiration.
if (cert.expires_at && new Date(cert.expires_at) < new Date()) {
```

```
    logResult("expired", cert.id)
    return { valid: false, reason: "expired", certificate: cert }
  }

  logResult("valid", cert.id)
  return { valid: true, certificate: cert }
}

// ---------------- Certificate requests ----------------

const SELECT_REQUEST_WITH_USER = `
  SELECT
    r.*,
    u.email AS user_email,
    u.full_name AS user_name,
    s.student_number AS student_number,
    s.group_name AS group_name
  FROM requests r
  JOIN users u ON u.id = r.user_id
  LEFT JOIN students s ON s.id = r.student_id
`

// Student creates a new certificate request.
export async function createRequest(formData: FormData) {
  const user = await requireUser()
  const db = getDb()
  const type = String(formData.get("type") || "").trim()
  const purpose = String(formData.get("purpose") || "").trim()

  if (!type) return { ok: false, error: "Оберіть тип довідки" }
```

```
    db.prepare(
      "INSERT INTO requests (user_id, student_id, type, purpose) VALUES
(?, ?, ?, ?)",
    ).run(user.id, user.student_id, type, purpose || "Для надання за місцем
вимоги")

    revalidatePath("/dashboard")
    revalidatePath("/admin")
    return { ok: true }
  }

  // Student's own requests.
  export async function getMyRequests(): Promise<RequestWithUser[]> {
    const user = await getCurrentUser()
    if (!user) return []
    const db = getDb()
    return toPlain<RequestWithUser>(
      db
        .prepare(`${SELECT_REQUEST_WITH_USER} WHERE r.user_id = ?
ORDER BY r.created_at DESC`)
        .all(user.id),
    )
  }

  // All requests (admin).
  export async function getRequests(): Promise<RequestWithUser[]> {
    await requireAdmin()
    const db = getDb()
    return toPlain<RequestWithUser>(
```

```
      db.prepare(`${SELECT_REQUEST_WITH_USER} ORDER BY r.created_at DESC`).all(),
    )
  }

  // Admin approves a request: issues a certificate and links it.
  export async function approveRequest(requestId: number) {
    await requireAdmin()
    const db = getDb()
    const req = db.prepare("SELECT * FROM requests WHERE id = ?").get(requestId) as
      | { id: number; student_id: number | null; type: string; purpose: string; status: string }
      | undefined
    if (!req) return { ok: false, error: "Заявку не знайдено" }
    if (req.status !== "pending") return { ok: false, error: "Заявку вже опрацьовано" }
    if (!req.student_id) {
      return {
        ok: false,
        error: "Користувач не прив’язаний до студента. Вкажіть номер студента в профілі.",
      }
    }

    const student = db.prepare("SELECT * FROM students WHERE id = ?").get(req.student_id) as
      | Student
      | undefined
    if (!student) return { ok: false, error: "Студента не знайдено" }
```

```
    const public_id = generatePublicId()
    const issued_at = new Date().toISOString()
    const signature = signPayload(
      buildSignaturePayload({
        public_id,
        full_name: student.full_name,
        student_number: student.student_number,
        type: req.type,
        issued_at,
      }),
    )

    db.prepare(
      `INSERT INTO certificates
        (public_id, student_id, type, purpose, issued_by, issued_at, expires_at,
status, signature)
       VALUES (?, ?, ?, ?, ?, ?, ?, 'active', ?)`,
    ).run(public_id, student.id, req.type, req.purpose, "Деканат", issued_at, null,
signature)

    db.prepare(
      "UPDATE requests SET status = 'approved', certificate_public_id = ?,
resolved_at = ? WHERE id = ?",
    ).run(public_id, issued_at, requestId)

    revalidatePath("/admin")
    revalidatePath("/dashboard")
    return { ok: true, public_id }
  }
```

```
export async function rejectRequest(requestId: number, note: string) {
  await requireAdmin()
  const db = getDb()
  db.prepare(
    "UPDATE requests SET status = 'rejected', note = ?, resolved_at = ?
WHERE id = ? AND status = 'pending'",
  ).run(note || "Відхилено деканатом", new Date().toISOString(), requestId)
  revalidatePath("/admin")
  revalidatePath("/dashboard")
  return { ok: true }
}

// Certificates belonging to the current student user.
export async function getMyCertificates(): Promise<CertificateWithStudent[]>
{
  const user = await getCurrentUser()
  if (!user || !user.student_id) return []
  const db = getDb()
  return toPlain<CertificateWithStudent>(
    db
      .prepare(`${SELECT_CERT_WITH_STUDENT} WHERE c.student_id
= ? ORDER BY c.issued_at DESC`)
      .all(user.student_id),
  )
}
```

**Лістинг А.3 Генерація QR-коду**

```
import { NextResponse } from "next/server"
import QRCode from "qrcode"

// Generates a PNG QR code that points to the public verification URL.
export async function GET(
  request: Request,
  { params }: { params: Promise<{ id: string }> },
) {
  const { id } = await params
  const origin = new URL(request.url).origin
  const verifyUrl = `${origin}/verify/${encodeURIComponent(id)}`

  const png = await QRCode.toBuffer(verifyUrl, {
    type: "png",
    width: 512,
    margin: 2,
    color: {
      dark: "#27272a",
      light: "#ffffff",
    },
  })

  return new NextResponse(new Uint8Array(png), {
    headers: {
      "Content-Type": "image/png",
      "Cache-Control": "public, max-age=86400",
    },
  })
}
```

# ДОДАТОК Б

## Інтерфейс розробленого вебзастосунку

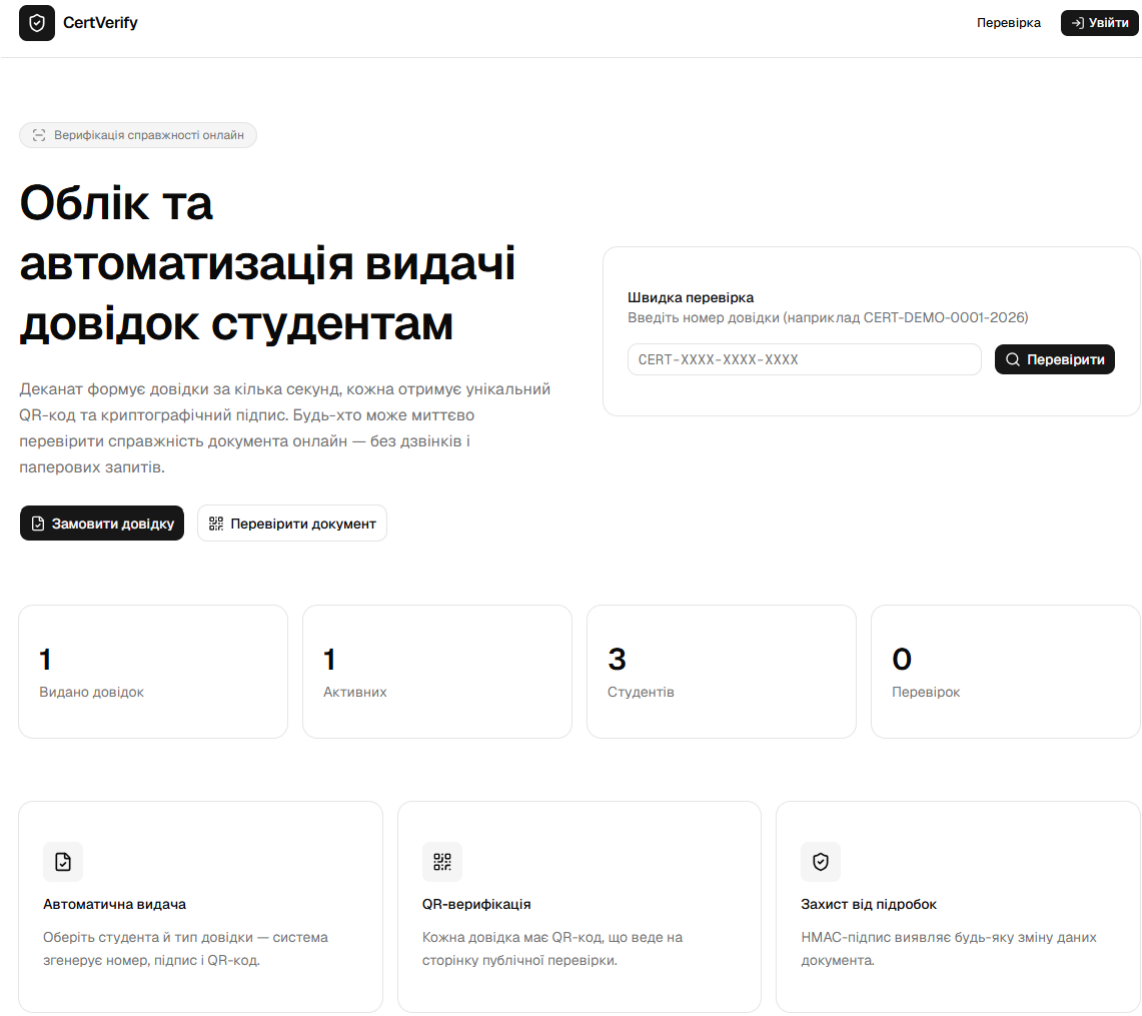



Рисунок Б.1 – Головна сторінка

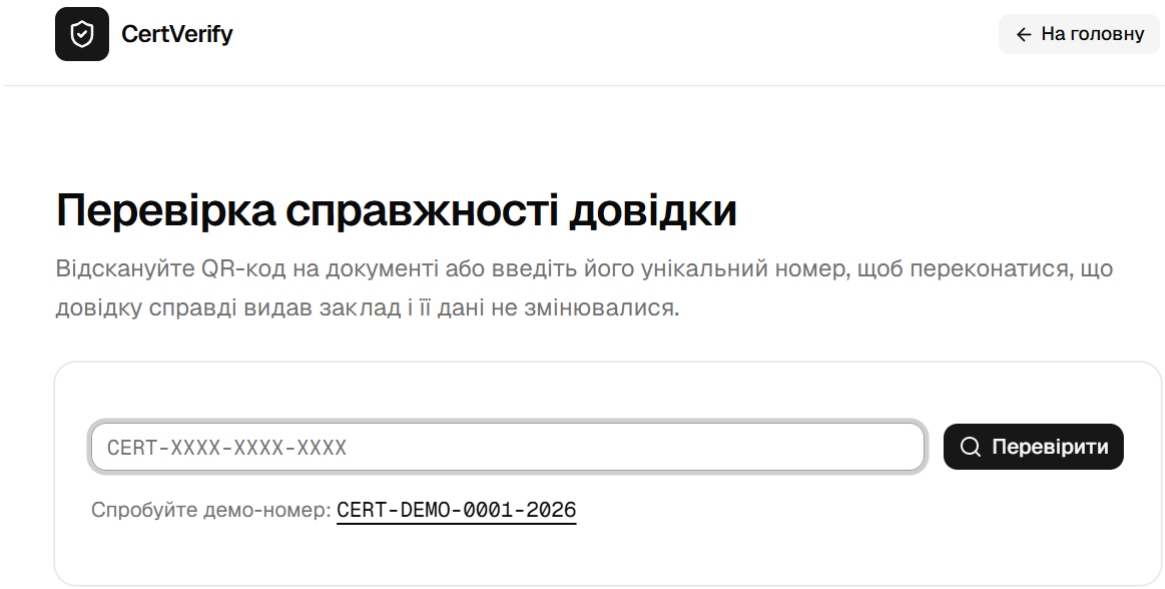



Рисунок Б.2 – Перевірка довідки

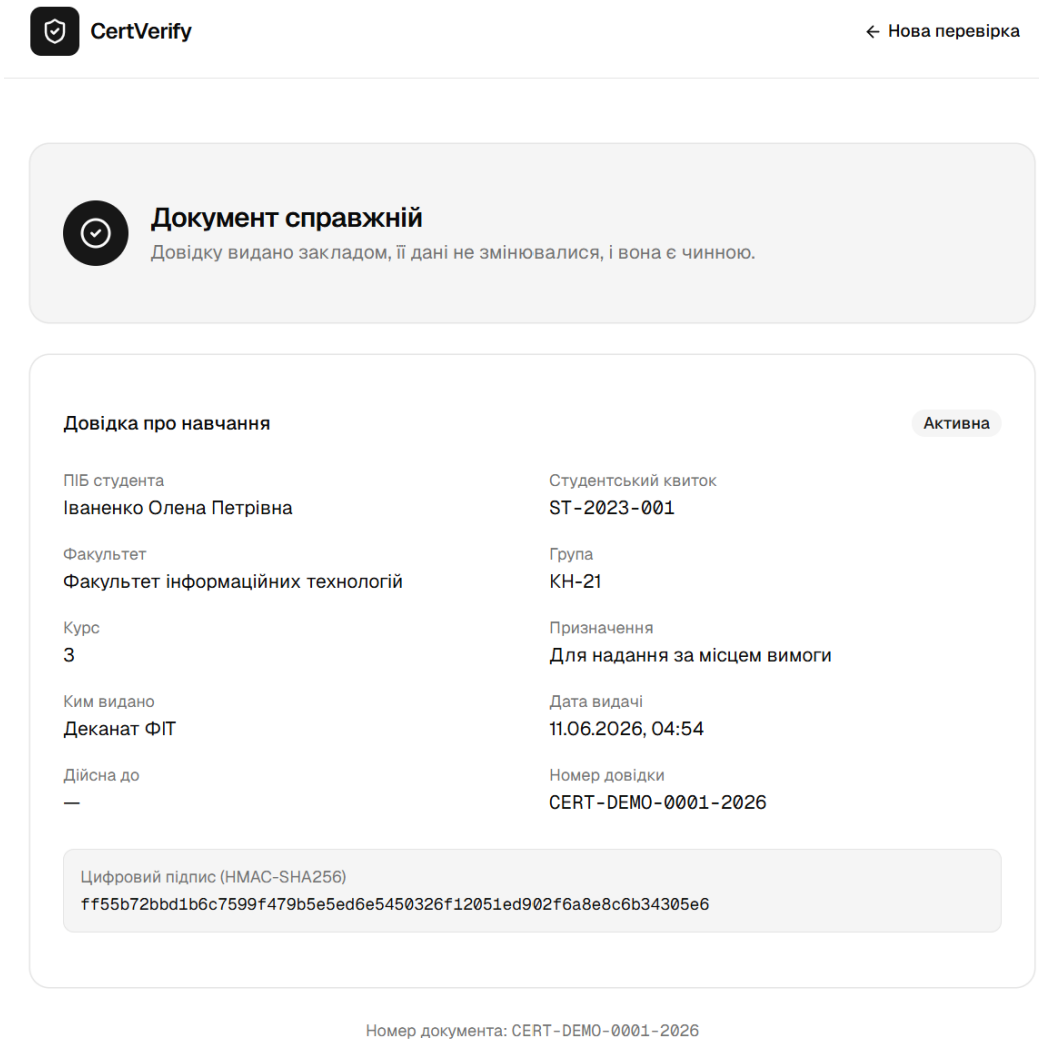



Рисунок Б.3 – Інформація про довідку

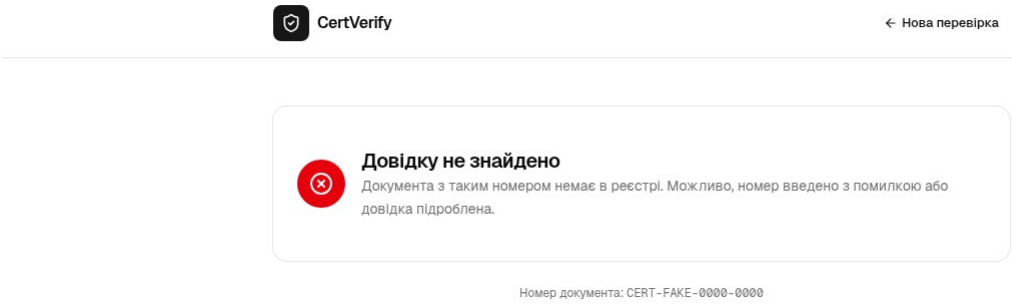



Рисунок Б.4 – Інформація про відсутність довідки

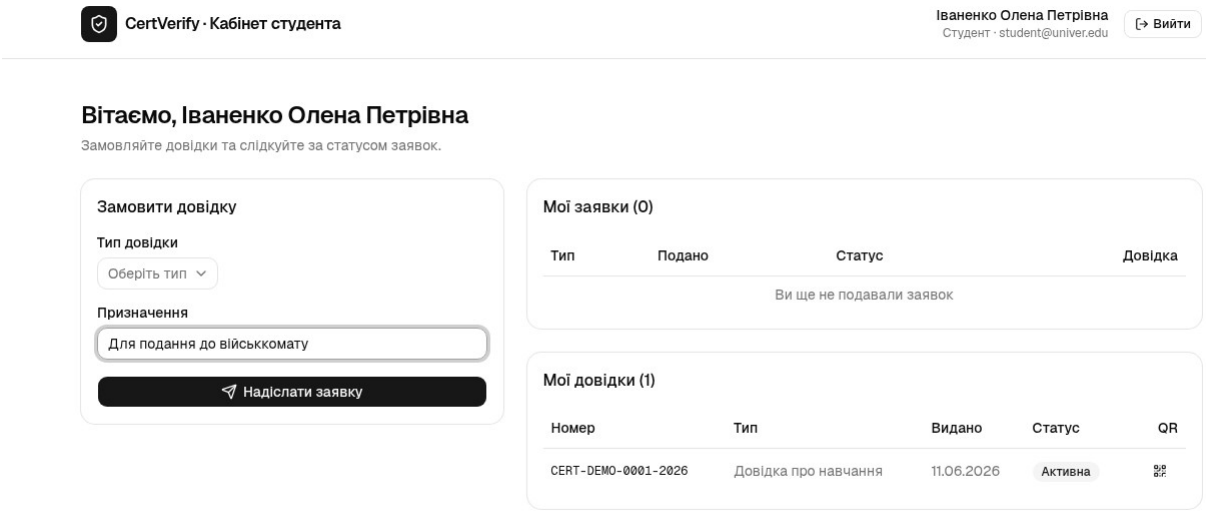



Рисунок Б.5 – Сторінка студента

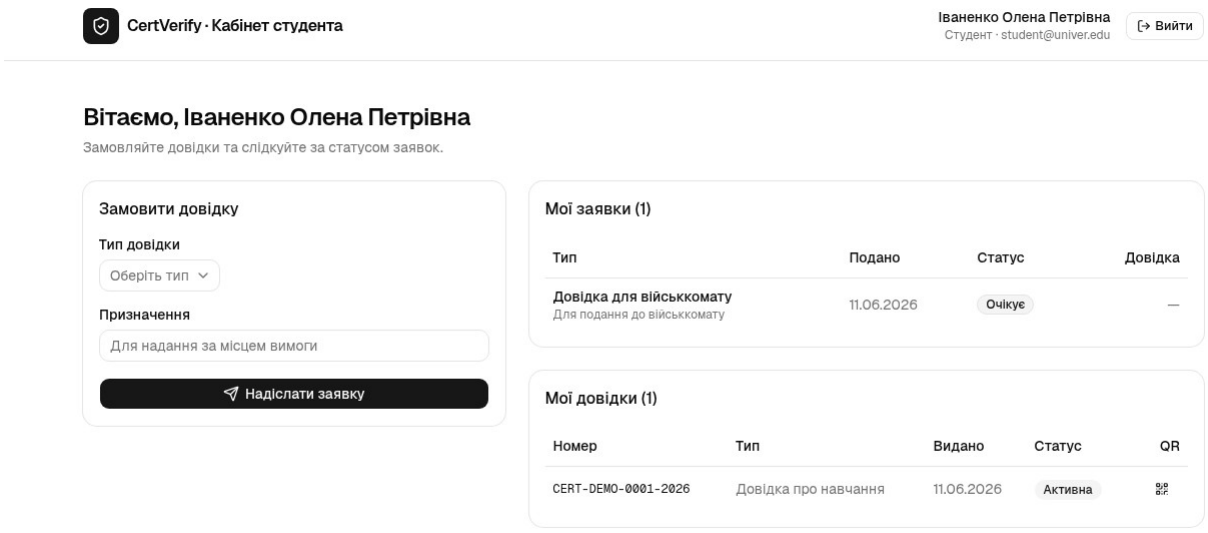



Рисунок Б.6 – Подання заявки на отримання довідки

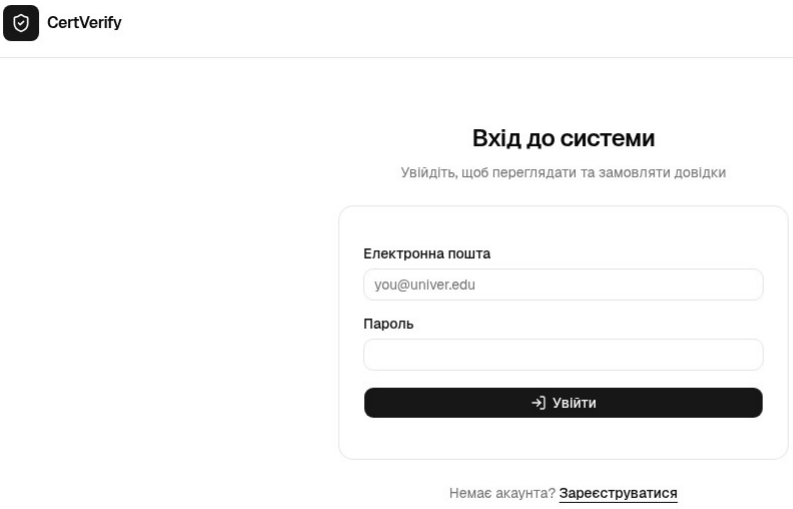



Рисунок Б.7 – Сторінка входу

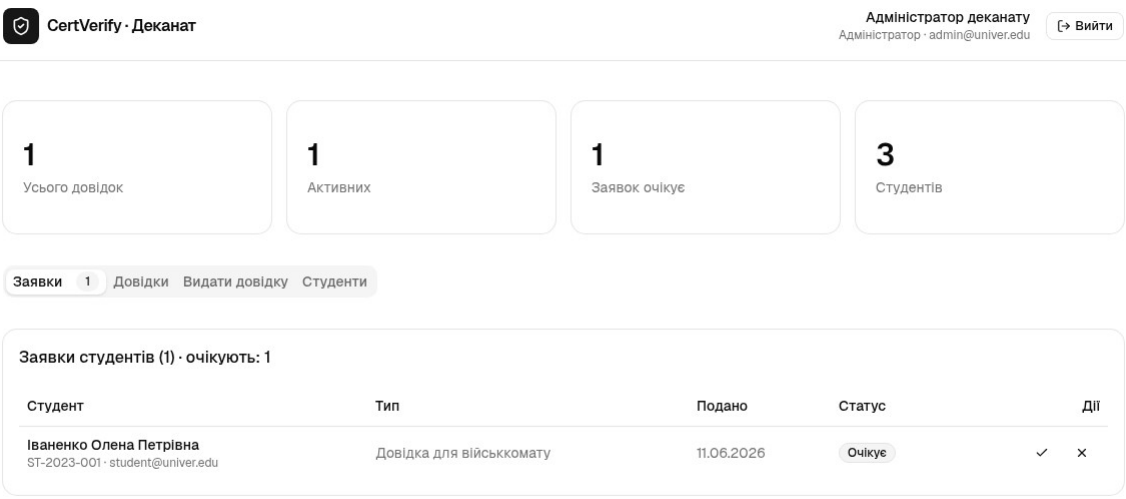



Рисунок Б.8 – Дашборд деканату

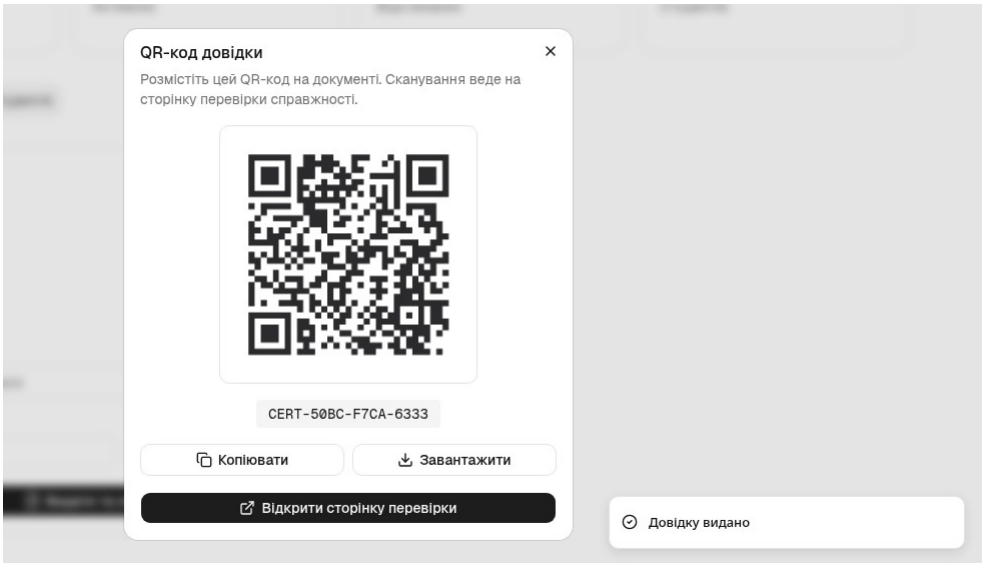



Рисунок Б.9 – Генерація QR-коду