МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет

«Харківський авіаційний інститут»

Факультет систем управління літальних апаратів

Кафедра комп’ютерних наук та інформаційних технологій

**Пояснювальна записка**

**до кваліфікаційної роботи**

(тип кваліфікаційної роботи)

бакалавра

(освітній ступінь)

Розробка веб-застосунку для генерації сертифікатів   
на підтвердження навчальних досягнень

на тему

ХАІ.302.336ст.25В.122.123018 ПЗ

Виконав: здобувач 4 курсу групи №336ст

Спеціальність 122 «Комп’ютерні науки»

(код та найменування)

Освітня програма Комп’ютеризація обробки інформації та управління .

Бабич А.О.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище й ініціали студента)

доц. каф. 302, к.т.н.

Керівник: *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Малєєва Ю.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище й ініціали)

Рецензент: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище й ініціали)

Харків – 2025



УДК 004.8

**РЕФЕРАТ**

Розробка веб-застосунку для генерації сертифікатів на підтвердження навчальних досягнень – Бабич Ангеліна Олександрівна, кваліфікаційна робота бакалавра, Харків, Національний аерокосмічний університет «Харківський авіаційний інститут», кількість сторінок, враховуючи додатки 95, кількість таблиць 15, кількість рисунків 30, кількість джерел літератури 25.

КЛЮЧОВІ СЛОВА: сертифікат, автоматизація процесів, генерація документів, веб-застосунок, Node.js, Vue.js, MongoDB.

МЕТА: розробка інструменту для автоматизації процесу створення сертифікатів для різноманітних подій, семінарів, курсів чи конференцій.

ОБ’ЄКТ ДОСЛІДЖЕННЯ: процес формування сертифікатів учасників семінарів, курсів, конференцій та інших освітніх подій.

ПРЕДМЕТ ДОСЛІДЖЕННЯ: методи та засоби веб-розробки, що використовуються для створення веб-застосунку для генерації сертифікатів.

ЗАВДАННЯ:

* проаналізувати проблеми та засоби автоматизації формування сертифікатів;
* виконати порівняльний аналіз програмного забезпечення аналогічного за обраними критеріями;
* провести аналіз вимог, визначити й візуалізувати на UML-діаграмі основні прецеденти;
* виконати проєктування інтерфейсу веб-застосунку та побудувати необхідні для успішної реалізації UML-діаграми;
* розробити програмний продукт та провести його тестування;
* описати проєктні рішення, інструменти, використані технології та підходи до розробки, режими роботи створеного застосунку;
* розрахувати економічні показники створеного продукту.

МЕТОДИ ДОСЛІДЖЕННЯ: Node.js, Vue.js, MongoDB.

ОТРИМАНІ РЕЗУЛЬТАТИ: розроблено веб-застосунок для генерації сертифікатів на підтвердження навчальних досягнень, що дозволяє автоматизувати та спростити процес створення персоналізованих сертифікатів учасників семінарів, курсів, конференцій та інших освітніх подій.

UDC 004.8

**ABSTRACT**

Development of a web application for generating certificates to confirming educational achievements – Babych Anhelina Oleksandrivna, Bachelor Thesis, Kharkiv, National Aerospace University "Kharkiv Aviation Institute," total number of pages including appendices: 95, number of tables: 15, number of figures: 30, number of references: 25.

KEYWORDS: certificate, process automation, document generation, web application, Node.js, Vue.js, MongoDB.

OBJECTIVE: development of a tool to automate the process of creating certificates for various events, seminars, courses, or conferences.

RESEARCH OBJECT: the process of generating certificates for participants of seminars, courses, conferences, and other educational events.

RESEARCH SUBJECT: methods and tools of web development used to create a web application for certificate generation.

TASKS:

* analyze the problems and tools for automating certificate generation;
* conduct a comparative analysis of software based on selected criteria;
* perform a requirements analysis, define and visualize key use cases using UML diagrams;
* design the web application interface and create the necessary UML diagrams for successful implementation;
* develop the software product and conduct its testing;
* describe project decisions, tools, technologies, and approaches used in development, as well as the operating modes of the created application;
* calculate the economic indicators of the developed product.

RESEARCH METHODS: Node.js, Vue.js, MongoDB.

RESULTS: a web application has been developed for generating certificates confirming educational achievements, enabling the automation and simplification of the process of creating personalized certificates for participants of seminars, courses, conferences, and other educational events.

**ЗМІСТ**

[ВСТУП 8](#_Toc199091317)

[1 АНАЛІЗ ПРОБЛЕМ ТА ЗАСОБІВ АВТОМАТИЗАЦІЇ ФОРМУВАННЯ СЕРТИФІКАТІВ ПРИ ПРОВЕДЕННІ СЕМІНАРІВ, КУРСІВ ТА КОНФЕРЕНЦІЙ 10](#_Toc199091318)

[1.1 Види подій та сертифікатів 10](#_Toc199091319)

[1.2 Проблема керування сертифікатами 12](#_Toc199091320)

[1.3 Інформаційні системи для генерації сертифікатів 12](#_Toc199091321)

[1.4 Висновки за розділом 19](#_Toc199091322)

[2 КОНЦЕПТУАЛЬНЕ ПРОЄКТУВАННЯ ЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ФОРМУВАННЯ СЕРТИФІКАТІВ НА ПІДТВЕРДЖЕННЯ НАВЧАЛЬНИХ ДОСЯГНЕНЬ 20](#_Toc199091323)

[2.1 Функціональні можливості системи 20](#_Toc199091324)

[2.2 Опис прецедентів системи 23](#_Toc199091325)

[2.3 Проєктування інтерфейсу користувача 25](#_Toc199091326)

[2.4 Технологічні аспекти проєктування інтерфейсу 30](#_Toc199091327)

[2.5 Висновки за розділом 31](#_Toc199091328)

[3 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА БАЗИ ДАНИХ ПРОДУКТУ 32](#_Toc199091329)

[3.1 Опис архітектури створеного продукту. Опис класів 32](#_Toc199091330)

[3.2 Опис бази даних 33](#_Toc199091331)

[3.3 Особливості розгортання програмного продукту 36](#_Toc199091332)

[3.4 Висновки за розділом 38](#_Toc199091333)

[4 РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ ТА ОПИС РЕЖИМІВ ЙОГО РОБОТИ 39](#_Toc199091334)

[4.1 Опис мови програмування 39](#_Toc199091335)

[4.2 Використані технології та бібліотеки 40](#_Toc199091336)

[4.3 Середовище розробки 46](#_Toc199091337)

[4.4 Опис основних режимів роботи веб-застосунку 48](#_Toc199091338)

[4.5 Тестування створеного продукту та його результати 51](#_Toc199091339)

[4.6 Висновки за розділом 54](#_Toc199091340)

[5 РОЗРАХУНОК ЕКОНОМІЧНИХ ПОКАЗНИКІВ 55](#_Toc199091341)

[5.1 Мета розділу 55](#_Toc199091342)

[5.2 Визначення тривалості та розрахунок собівартості виконання проєкту (перший варіант) 55](#_Toc199091343)

[5.3 Визначення тривалості та розрахунок собівартості виконання проєкту (другий варіант) 61](#_Toc199091344)

[5.4 Висновки за розділом 65](#_Toc199091345)

[ВИСНОВКИ 66](#_Toc199091346)

[СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 67](#_Toc199091347)

[ДОДАТОК А. ВИХІДНІ КОДИ (ФРАГМЕНТ) 70](#_Toc199091348)

[ДОДАТОК Б. ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ 74](#_Toc199091349)

# 

# ВСТУП

**Актуальність теми дослідження:** в сучасному світі, коли онлайн-освіта, вебінари та онлайн-курси стають все більш популярними, потреба в зручному та швидкому способі генерування сертифікатів для учасників стає надзвичайно актуальною. Ручне створення сертифікатів для кожного учасника забирає значну кількість часу та може призвести до помилок. Веб-застосунок для генерації сертифікатів в наш час є необхідним інструментом для освітніх установ, конференцій та інших подій, що надають учасникам сертифікати про участь.

Розробка такого веб-застосунку не тільки спростила б процес генерації сертифікатів, але й дозволила б автоматизувати цей процес, зменшити ймовірність помилок та зберегти час організаторів подій. Крім того, веб-застосунок забезпечить зручний доступ до сертифікатів для кожного учасника, що є додатковою перевагою для користувачів.

Такий веб-застосунок також може забезпечити збереження даних про учасників та їхні сертифікати в електронному форматі, що дозволить легко відстежувати участь у різних подіях та швидко пред'являти сертифікати на запит. Це сприятиме покращенню процесів адміністрування та архівування даних, що є важливим аспектом в організації будь-яких освітніх заходів. Такий веб-застосунок може також враховувати різні дизайни сертифікатів, щоб задовольнити потреби різних організацій та подій.

З огляду на швидкий розвиток онлайн-освіти та необхідність ефективного керування даними про учасників подій, розробка подібного веб-застосунку є дійсно актуальною і перспективною темою. Вона не тільки відповідає потребам сучасного освітнього середовища, але й сприяє оптимізації процесів управління та забезпечує зручний доступ до важливої інформації для всіх зацікавлених сторін.

**Мета дослідження –** розробка інструменту для автоматизації процесу створення сертифікатів для різноманітних подій, семінарів, курсів чи конференцій.

Для досягнення мети, необхідно вирішити наступні **завдання**:

* проаналізувати проблеми та засоби автоматизації формування сертифікатів;
* виконати аналіз програмного забезпечення аналогічного призначення та провести порівняння за обраними критеріями;
* провести аналіз вимог та визначити й візуалізувати на UML-діаграмі основні прецеденти;
* використовуючи спеціалізоване програмне забезпечення, виконати проєктування інтерфейсу веб-застосунку та побудувати необхідні для успішної реалізації UML-діаграми;
* розробити програмний продукт та провести його тестування;
* описати проєктні рішення, інструменти, використані технології та підходи до розробки;
* описати режими роботи створеного застосунку;
* розрахувати економічні показники створеного продукту.

**Об’єкт дослідження**– процес формування сертифікатів учасників семінарів, курсів, конференцій та інших освітніх подій.

**Предмет дослідження** – методи та засоби веб-розробки, що використовуються для створення веб-застосунку для генерації сертифікатів.

В процесі виконання кваліфікаційної роботи були використані наступні **методи дослідження**: Node.js, Vue.js, MongoDB.

**Практичне значення отриманих результатів.** Розроблено веб-застосунок для генерації сертифікатів на підтвердження навчальних досягнень, що дозволяє автоматизувати та спростити процес створення персоналізованих сертифікатів.

**Структура та обсяг кваліфікаційної роботи бакалавра.** Кваліфікаційна робота бакалавра складається зі вступу, 5 розділів, висновків, 2 додатків та переліку посилань. Повний обсяг кваліфікаційної роботи бакалавра становить 94 сторінки, в тому числі: 29 рисунки; 15 таблиць; 2 додатки на 26 сторінках; перелік посилань з 25 найменувань на 3 сторінках.

# АНАЛІЗ ПРОБЛЕМ ТА ЗАСОБІВ АВТОМАТИЗАЦІЇ ФОРМУВАННЯ СЕРТИФІКАТІВ ПРИ ПРОВЕДЕННІ СЕМІНАРІВ, КУРСІВ ТА КОНФЕРЕНЦІЙ

## 1.1 Види подій та сертифікатів

Подія – це те, що відбувалося або відбулося, сталося; явище, факт суспільного або особистого життя. Освітня подія — це організований захід, спрямований на навчання, підвищення кваліфікації або обмін знаннями. Вона може бути у формі лекції, семінару, тренінгу, конференції чи курсу, де учасники здобувають нові навички та знання. Після успішного проходження такої події учасники отримують сертифікат, який підтверджує їхні освітні досягнення [1].

Розглянемо деякі з видів освітніх подій окремо. Семінар – це одна з форм навчання, при якій учасники дискутують, обговорюють представлену проблему. Семінар прийнято відносити до навчальної форми у вищих навчальних закладах. Хоча, на даний момент, семінари стали поширеними також для особистісного навчання, навчання бізнесу та багатьох інших сфер життя людини.

Семінар як форма навчання з’явилася в Стародавній Греції і Стародавньому Римі. На семінарах студенти виступали зі своєю доповіддю, після чого обмінювалися думками, проводили дискусії і приходили до спільного консенсусу. Саме знаходження єдиного правильного рішення було основною метою і функцією семінарів. Форми проведення семінарів.

Всі семінари прийнято ділити на три категорії: навчальні, науково-практичні і бізнес-семінари.

Перший вид семінарів відноситься до навчальних занять в університеті.

Науково-практичні семінари збирають аудиторію науковців, які вже мають досвід і знання в обговорюваній темі і знайомляться з роботами своїх колег.

Бізнес-семінари – це обмін досвідом серед представників однієї спеціальності, удосконалення знань щодо нових методик роботи. Саме бізнес-семінари вважаються найкращим методом підвищення кваліфікації.

Семінари, які можуть організовуватися в рамках однієї компанії називаються корпоративними. Вони можуть бути відкритими – вільними для відвідування усіма зацікавленими учасниками. У будь-якому випадку, корпоративний або відкритий, семінар допомагає ознайомитися з новими тенденціями, методиками роботи і бути на крок попереду своїх конкурентів. В окрему категорію можна винести також вебінари.

Сучасні технології покликані полегшувати життя своїм користувачам. Тепер зовсім не обов’язково долати величезні відстані, щоб бути присутнім на тому чи іншому семінарі. Багато компаній і окремих спікерів організовують семінари в режимі «онлайн». Структура таких вебінарів не відрізняється від звичайних офлайн зустрічей: спікер представляє свою тему, в ході доповіді учасники вебінару можуть писати свої питання в коментарях. Виступаючий читає питання і відразу ж дає на них відповіді [2].

Онлайн-курс – це програма навчання, яка організована відповідно до навчального плану (зазвичай за модулями) та проходить у віртуальному просторі. Онлайн-курси можуть бути неформальними та зосередженими на одній навичці або формальними, що ведуть до отримання сертифікату чи ступеня.

Різні онлайн-курси мають різні функції: деякі курси можуть бути суворо прив’язані до навчальної програми, інші можуть дозволити вам взяти перерву та зробити перерву у класі, коли дозволяє ваш розклад. Деякі можуть мати «живі» дискусійні групи, заплановані для відеоконференцій у певний час, інші можуть просто записувати лекції та надавати вам доступ до їх відео в будь-який час.

Онлайн-курси для самостійного навчання – це онлайн-курси, на яких ви можете вивчати розділи, проходити тести, відвідувати лекції та виконувати будь-яку курсову роботу у зручний для вас час – немає фіксованих термінів для проходження програми або її частин. Це дає більше автономії щодо того, де і коли виконувати курсову роботу, але декому важко залишатися в курсі через відсутність структури.

У онлайн-курсів багато переваг. Однією з основних переваг є гнучкість: онлайн-курси доступні у будь-який час та в будь-якому місці. Онлайн-курси з самостійним вивченням, де навчання здійснюється самостійно і немає жодних тимчасових зобов’язань, крім того, що студент мотивований витратити, часто ідеально підходять для тих, хто має щільний графік або хто намагається поєднувати сім’ю або кілька робіт [3].

Тренінг – це перш за все, навчання, яке базується на досвіді людини, розвиває здібності особистості до пошуку нових знань, творчості, допомагає здійснити інтелектуальний прорив, формує новий світогляд.

Важливою особливістю цієї форми підвищення кваліфікації є те, що його учасники не лише отримують нову інформацію, але відпрацьовують одночасно нові навички, використовуючи їх на практиці.

Тренінги умовно поділяються на дві групи: одні мають на меті формування професійних, ділових навичок і умінь працівника та/або групи працівників – це бізнес-тренінги, а інші торкаються психологічних аспектів спілкування в колективі та у виробничій діяльності і формують навички поведінки – це психологічні або соціально-психологічні тренінги. Окремо слід відзначити тренінги вендорів – великих компаній-постачальників програмного забезпечення. Кожна така компанія зазвичай має свій власний підхід до навчання, включаючи специфічні навчальні інструменти, методичні та психологічні аспекти.

Під час кожного тренінгу пропонується виконати ті чи інші вправи, орієнтовані на розвиток чи демонстрацію певних психологічних якостей або навичок. Для цього застосовуються рольові, ділові ігри та інші методи [4].

## 1.2 Проблема керування сертифікатами

Сертифікати відіграють критичну роль у перевірці автентичності, конфіденційності та цілісності даних під час їх передачі через мережу. Однак існує низка проблем, пов'язаних з керуванням сертифікатами, які можуть впливати на їх ефективність та надійність.

Перш за все, проблема керування сертифікатами пов'язана з технічними аспектами їх використання та розповсюдження. Це включає в себе:

* як сертифікати створюються та підписуються;
* яким чином сертифікати поширюються серед користувачів та систем;
* як швидко та ефективно можна відкликати сертифікати у випадку компрометації або зміни статусу.

Не менш важливим аспектом є безпека. Вона включає в себе стійкість до атак і достатній рівень захисту конфіденційних даних.

Одна з ключових проблем полягає в керуванні життєвим циклом сертифікатів, включаючи їхнє вчасне оновлення, відкликання та видалення після закінчення терміну дії.

Проблема керування сертифікатами є складною та багатогранною. Вона охоплює технічні, безпекові та організаційні аспекти, які потребують уваги для забезпечення ефективності та безпеки систем, що використовують цифрові сертифікати. Для подолання цих проблем необхідні комплексні підходи та постійне вдосконалення процесів керування сертифікатами.

## 1.3 Інформаційні системи для генерації сертифікатів

Розглянемо кілька доступних на ринку програмних продуктів, які можуть конкурувати з нашим, проаналізуємо їхні переваги та недоліки, щоб визначити найкращий вибір для реалізації подібних сценаріїв. Також треба звернути увагу на те, яка функціональність існуючих рішень впливатиме на проєктні рішення під час розробки генератора сертифікатів.

1. Free Certificate Maker for Custom Certificates [5] є потужним онлайн-інструментом для створення індивідуальних сертифікатів, що надає користувачам можливість легко та швидко створювати професійні та привабливі сертифікати для різних потреб.

Сайт пропонує широкий вибір попередньо розроблених шаблонів сертифікатів, які користувачі можуть налаштовувати відповідно до своїх вимог. Ці шаблони містять усі необхідні елементи для створення сертифікатів, що економить час і зусилля. Інструмент підтримує додавання брендових кольорів і шрифтів, що дозволяє створювати сертифікати, які відповідають фірмовому стилю компанії чи установи.

Користувачі можуть почати створення сертифіката, обравши один з доступних шаблонів. Після цього вони можуть налаштувати текст, кольори, графіку та інші елементи за допомогою зручного інтерфейсу.

Інструмент підтримує інтеграцію з іншими сервісами, що дозволяє додавати до сертифікатів логотипи та інші графічні елементи.

Після завершення дизайну сертифіката, користувачі можуть завантажити його у форматі PDF з мітками для друку або як зображення для електронної розсилки. Також є можливість генерувати посилання для онлайн-поширення сертифікатів.

Цей веб-застосунок є ідеальним рішенням для тих, хто шукає зручний та ефективний спосіб створення сертифікатів без необхідності мати глибокі знання у графічному дизайні. Завдяки своєму простому інтерфейсу та багатому набору функцій, цей інструмент підходить для широкої аудиторії користувачів, від освітян до бізнесменів (рисунок 1.1 та рисунок 1.2), забезпечуючи їхні потреби у створенні персоналізованих сертифікатів. Є також можливість зберегти створений дизайн для повторного використання, створивши таким чином корпоративну бібліотеку шаблонів сертифікатів.

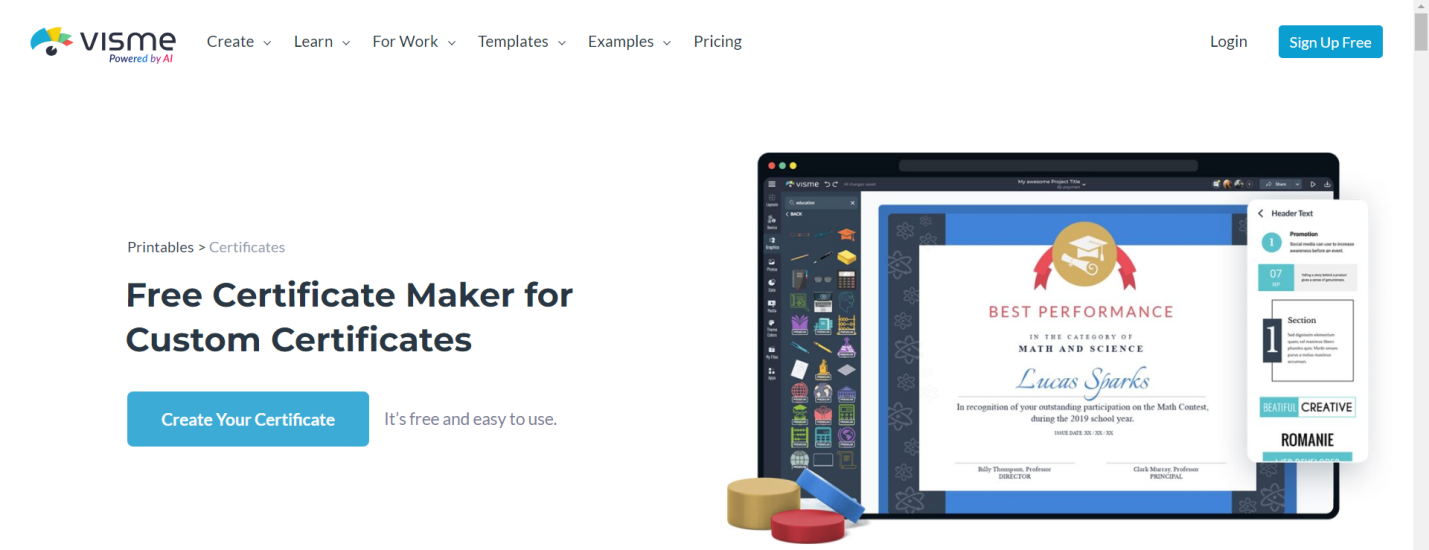


Рисунок 1.1 – Загальний вигляд головної сторінки   
Free Certificate Maker for Custom Certificates

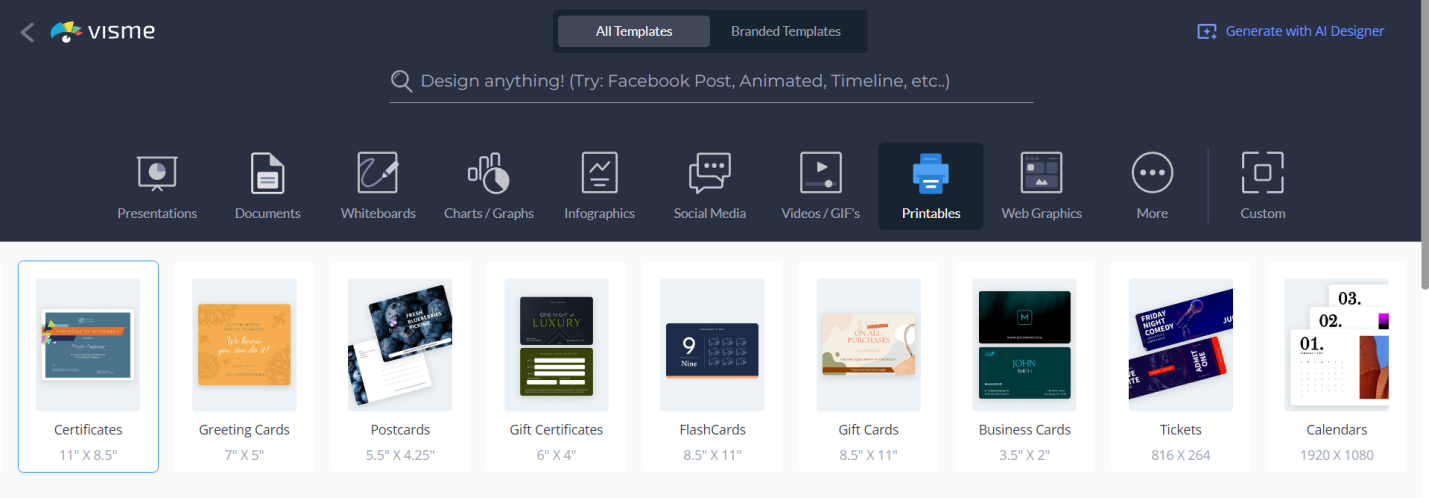


Рисунок 1.2 – Шаблони документів, доступні до використання у   
Free Certificate Maker for Custom Certificates

1. Free Online Certificate Maker (Canva) [6] – це онлайн-інструмент для створення сертифікатів, який пропонує широкий вибір шаблонів та стилів, що дозволяють створювати професійно виглядаючі сертифікати для будь-яких нагод.

Одна з головних переваг Canva – можливість гнучкого налаштування кожного елемента дизайну. Платформа забезпечує простий інтерфейс drag-and-drop, який не вимагає спеціальних навичок дизайну, що дозволяє користувачам швидко налаштувати текст, кольори, шрифти та фони. Це дозволяє створити сертифікати, які не тільки виглядають привабливо, але й відповідають корпоративному стилю чи тематиці події.

Canva також пропонує можливість додавання високоякісних зображень та ілюстрацій, а також функцію спільної роботи над дизайном в реальному часі. Готові сертифікати можна зберегти у форматах PDF, JPG або PNG, або замовити їх друк через Canva Print.

Canva має багатий набір шаблонів для різних типів сертифікатів, включаючи сертифікати визнання, участі, завершення курсів, досягнень та дипломи. Користувачі можуть легко персоналізувати дизайн, додаючи власні елементи, тексти та зображення, і замовляти високоякісний друк з швидкою доставкою.

Крім того, Canva підтримує спільну роботу, що дає змогу кільком користувачам редагувати один документ, коментувати внесені зміни та вдосконалювати дизайн у режимі реального часу.

Canva робить процес створення сертифікатів доступним для всіх, навіть без досвіду в дизайні, і надає інструменти для створення привабливих та професійних сертифікатів, які можна використовувати для різних подій   
(рисунок 1.3 та рисунок 1.4).

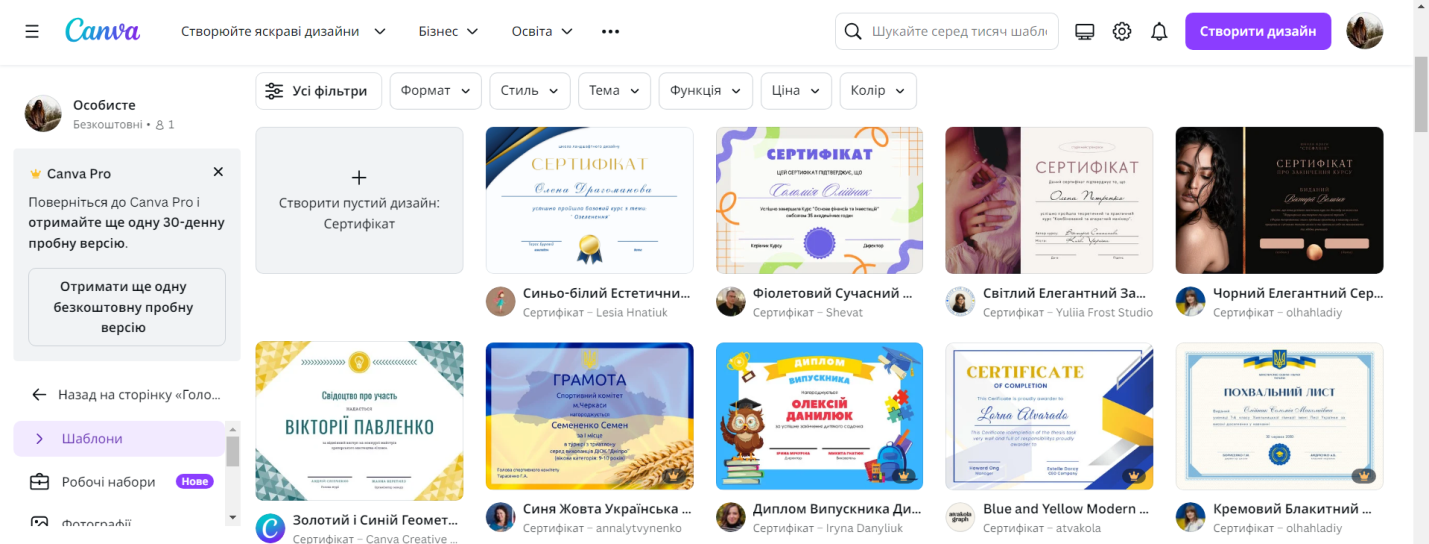


Рисунок 1.3 – Загальний вигляд інтерфейсу програми Canva

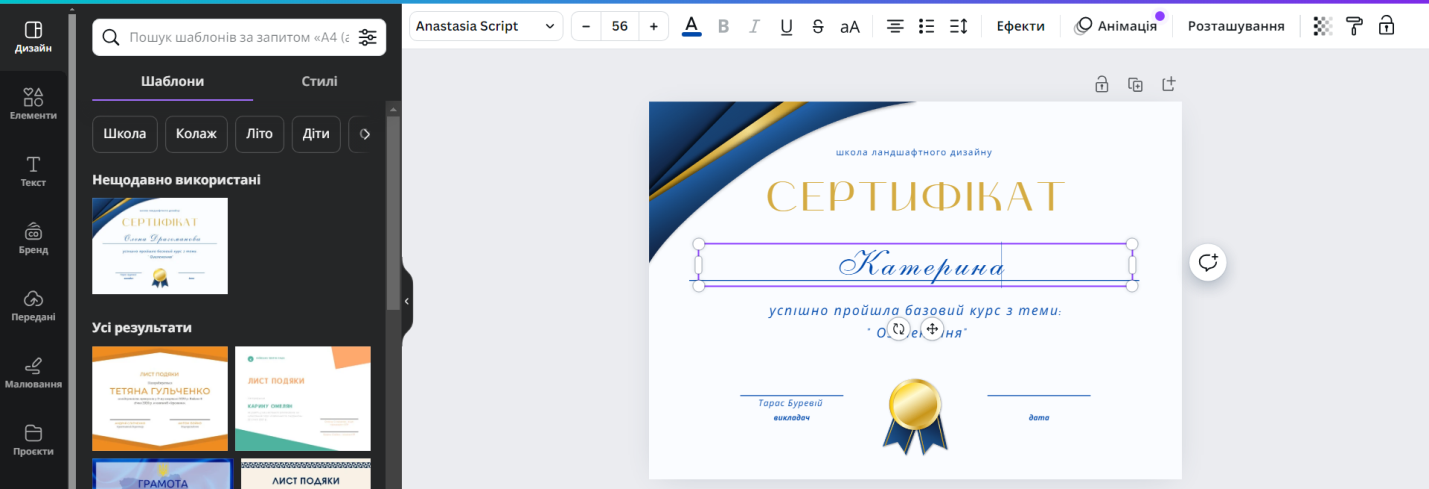
****

Рисунок 1.4 – Загальний вигляд конструктора сертифікатів в Canva

1. Online Certificate Maker [7] пропонує безкоштовний інструмент для створення сертифікатів в онлайн режимі. Користувачі можуть легко генерувати сертифікати для різних подій, таких як досягнення студентів або співробітників. Сервіс підтримує автоматичне створення сертифікатів за допомогою API, що дозволяє згенерувати велику кількість документів.

Ключові особливості:

* високоякісні шаблони векторного формату;
* автоматичне налаштування розміру шрифта для довгих імен;
* підтримка різних мов, включаючи англійську, китайську, японську, арабську та інші;
* налаштування кольорів та шрифтів;
* можливість завантаження сертифікатів у форматах PDF та JPG.

Цей інструмент ідеально підходить для професіоналів, які хочуть швидко створити красиві та якісні сертифікати без необхідності дизайнерських навичок. Користувачі можуть створювати і завантажувати до п'яти сертифікатів щодня безкоштовно.

Сайт пропонує і інші безкоштовні інструменти, такі як генератор запрошень на весілля, генератор ID-карток для заходів та інші, але основним його призначенням є створення сертифікатів (рисунок 1.5 та рисунок 1.6).



Рисунок 1.5 – Головна сторінка програмного продукту Online Certificate Maker

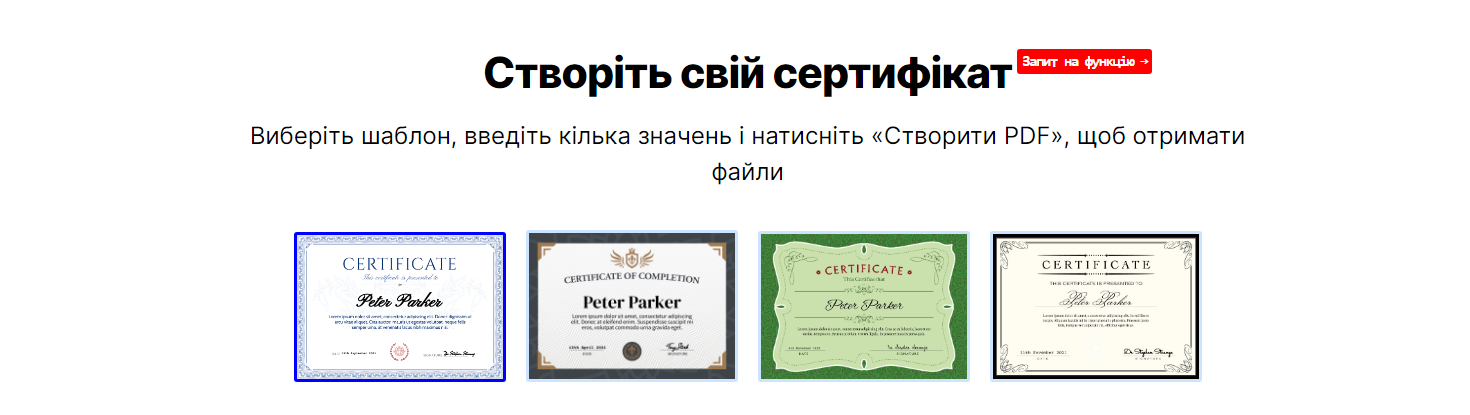


Рисунок 1.6 – Вибір шаблону для створення сертифікату

Порівняємо розглянуті рішення для генерації сертифікатів за такими критеріями, як адаптивність, багатокористувацький режим, інтуїтивно-зрозумілий інтерфейс користувача, наявність української мови, можливість імпорту/експорту даних, крос-платформність (наявність версій для Linux/MacOS), автентифікація з використанням соціальних облікових записів (Google/Facebook/Twitter тощо), наявність довідкових матеріалів та документації, можливості з налагодження та розширення функціональності   
(Rest API/SDK), вартість ліцензії. Результати порівняння занесемо в таблицю 1.1, в якій знаком «+» позначимо наявність, а знаком «–» – відсутність реалізації того чи іншого критерію в розглянутому програмному забезпеченні.

Таблиця 1.1 – Порівняльна характеристика програмних продуктів для генерації сертифікатів

|  | Free Certificate Maker for Custom Certificates | Free Online Certificate Maker | Online Certificate Maker | Розроблений програмний продукт |
| --- | --- | --- | --- | --- |
| Адаптивність | + | + | + | + |
| Багатокористувацький режим | – | + | – | + |
| Інтуїтивно-зрозумілий інтерфейс користувача | + | + | + | + |
| Наявність української мови | + | + | + | + |
| Можливість імпорту/експорту даних | + | + | + | + |
| Крос-платформність (наявність версій для Linux/MacOS) | + | + | + | + |
| Автентифікація з використанням соціальних облікових записів (Google/Facebook/Twitter тощо) | + | + | + | – |
| Наявність довідкових матеріалів та документації | + | + | + | + |
| Можливості з налагодження та розширення функціональності (Rest API/SDK) | + | + | + | + |
| Вартість ліцензії | Pro 1004.70 грн /місяць  Starter 497.27 грн/місяць | Преміум 608.91 грн /місяць | Enterprise  12137.56 грн /місяць | – |

Виконаний порівняльний аналіз дав наступні результати:

* всі розглянуті програмні продукти отримали позитивну оцінку за адаптивність, що робить їх придатними для використання на різних пристроях та різних розмірах екрану;
* так само всі розглянуті продукти отримали позитивну оцінку за інтуїтивно-зрозумілий інтерфейс, що сприяє зручності та ефективності використання;
* всі розглянуті продукти підтримують українську мову, що може бути конкурентною перевагою на ринку україномовних користувачів;
* у всіх програмних продуктах є можливість імпорту та експорту даних, що є важливим для обміну інформацією з іншими системами або збереженням даних у зручному форматі;
* у всіх розглянутих продуктах є крос-платформна підтримка, що дозволяє їх використання на різних операційних системах.

Таким чином, порівняльний аналіз недоліків та переваг програмних продуктів аналогічного призначення привів нас до висновку про доцільність розробки власного програмного продукту для генерації сертифікатів на підтвердження навчальних досягнень, що дозволяє автоматизувати та спростити процес створення персоналізованих сертифікатів учасників семінарів, курсів, конференцій та інших освітніх подій, який буде надавати наступні функціональні можливості:

* реєстрація та авторизація користувачів;
* можливість створення нового сертифікату;
* вибір шаблону для сертифікату;
* введення або вибір необхідних даних для сертифікату;
* автоматична генерація унікального номера сертифікату;
* можливість експорту сертифікату в форматі PDF;
* можливість друку сертифікату;
* збереження запису про кожен створений сертифікат у базі даних;
* розмежування прав доступу користувачів.

При цьому слід також подбати про:

* привабливий та зручний інтерфейс користувача;
* адаптивний дизайн для забезпечення можливості використання застосунку на різних пристроях;
* високу швидкість роботи веб-застосунку навіть при великій кількості одночасних користувачів;
* захист персональних даних користувачів та забезпечення конфіденційності інформації про сертифікати;
* стійкість до збоїв та відновлення роботи після відмови;
* масштабованість системи для можливості розширення функціональності та обробки більшої кількості даних у майбутньому;
* відповідність стандартам з безпеки веб-застосунків;
* простоту встановлення, використання та підтримки.

## 1.4 Висновки за розділом

Отже, проаналізувавши основні проблеми і засоби автоматизації формування сертифікатів при проведенні освітніх подій, ми виділили основні види таких подій і розглянули типові проблеми керування сертифікатами, що охоплюють технічні, безпекові та організаційні аспекти.

В розділі також проаналізовано ряд продуктів аналогічного призначення, а саме Free Certificate Maker for Custom Certificates, Free Online Certificate Maker (Canva) та Online Certificate Maker. За результатами порівняльного аналізу їх недоліків та переваг зроблено висновок про доцільність розробки власного програмного продукту – веб-застосунку для генерації сертифікатів на підтвердження навчальних досягнень.

Веб-застосунок розробляється як новий продукт, призначений для використання в рамках освітньої установи, і не має на меті конкурувати з розглянутими в розділі публічно доступними системами.

# КОНЦЕПТУАЛЬНЕ ПРОЄКТУВАННЯ ЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ФОРМУВАННЯ СЕРТИФІКАТІВ НА ПІДТВЕРДЖЕННЯ НАВЧАЛЬНИХ ДОСЯГНЕНЬ

## 2.1 Функціональні можливості системи

Розроблений веб-застосунок створений для генерації, експорту та друку сертифікатів, що підтверджують участь у різних подіях, таких як тренінги, вебінари, конференції тощо. Основними функціями веб-застосунку є надання можливості користувачам обирати шаблони сертифікатів, введення необхідних даних для персоналізації сертифікату, автоматична генерація унікального номера для кожного сертифікату та збереження відповідних записів у базі даних. Також передбачається розмежування прав доступу для різних користувачів системи залежно від їх ролей.

Крім того, веб-застосунок має адаптивний інтерфейс користувача, який дозволяє використовувати його на різних пристроях. Готовий продукт буде опублікований в мережі, щоб забезпечити його доступність для використання широким колом користувачів.

Функціональність програмного забезпечення – це здатність програмного продукту виконувати набір функцій, визначених у його зовнішньому описі, і які відповідають заданим або імовірною потребам користувачів [8].

Функціональні вимоги до веб-застосунку для генерації сертифікатів:

* реєстрація та авторизація користувачів;
* можливість створення нового сертифікату;
* вибір шаблону для сертифікату;
* введення або вибір необхідних даних для сертифікату;
* автоматична генерація унікального номера сертифікату;
* можливість експорту сертифікату у форматі PDF;
* можливість друку сертифікату;
* збереження запису про кожен створений сертифікат у базі даних;
* розмежування прав доступу користувачів.

Зазвичай функціональні вимови візуалізують за допомогою UML-діаграми прецедентів. Діаграма прецедентів – це узагальнене представлення функціонального призначення системи, яке має відповідати на головні питання моделювання: що робить система у зовнішньому світі? Яку функціональність надає користувачам?

На діаграмі прецедентів застосовують два типи базових сутностей: варіанти використання (прецеденти) і діючі особи (актори), між якими встановлюють такі типи відношень:

* асоціація між актором та прецедентом;
* узагальнення між акторами;
* узагальнення між прецедентами;
* залежності чи асоціації (різних типів) між прецедентами.

Під час виявлення прецедентів акцентують увагу на поводженні системи стосовно заданого актора. Прецеденти допомагають визначити можливості та межі системи. Нотація для прецеденту досить скромна – це текст (наказова, дієслівна форма), який, залежно від Case-засобу, розташований в овалі або під овалом.

Прецеденти визначають за специфікацією вимог до системи. Актором може бути будь-який взаємодіючий з системою зовнішній суб’єкт: фізична особа (наприклад, торговий агент), зовнішня програмна система (скажімо, програма підготовки і друку рахунків) чи пристрій (наприклад, датчик температури). Усі процеси взаємодії між діючими суб’єктами і системою розглядають як прецеденти.

Діаграми прецедентів відображають статичні аспекти системи з позиції користувачів. Ця позиція охоплює, здебільшого, поведінку системи, тобто видимі ззовні сервіси, які надає система.

UML дає змогу моделювати контекст за допомогою діаграм прецедентів, у яких увагу акцентують на зовнішніх акторах. Важливо правильно визначити акторів, оскільки це даватиме змогу описати класи сутностей, що взаємодіють із системою. Ще важливіше визначити, хто не є актором, адже при цьому обмежується оточення системи (у ньому залишаються тільки ті елементи, що беруть участь у її роботі) [9].

Діаграма прецедентів наочно демонструє функціональні можливості системи, а також способи взаємодії користувача з нею. Вона допомагає визначити ключові сценарії використання та роль кожного учасника у процесі виконання завдань.Ця діаграма зазвичай є першою, яку будують в процесі роботи над продуктом. Вона є результатом концептуального проєктування і відправною точкою логічного проєктування.

Також діаграму прецедентів використовують як засіб комунікації між командою розробників і замовником, і як засіб контролю прогресу розробки.

Діаграму прецедентів для нашого продукту показано на рисунку 2.1. Вона демонструє функціональність, яку веб-застосунок надає для двох категорій користувачів – назвемо їх студенти (відвідувачі, слухачі лекцій, семінарів, курсів тощо), які можуть увійти до веб-застосунку і переглянути отримані ними сертифікати, і викладачі (спікери, організатори подій), які можуть вводити дані (додавати студентів, вводити інформацію про події), обирати і створювати шаблони сертифікатів, генерувати, друкувати і завантажувати сертифікати тощо.

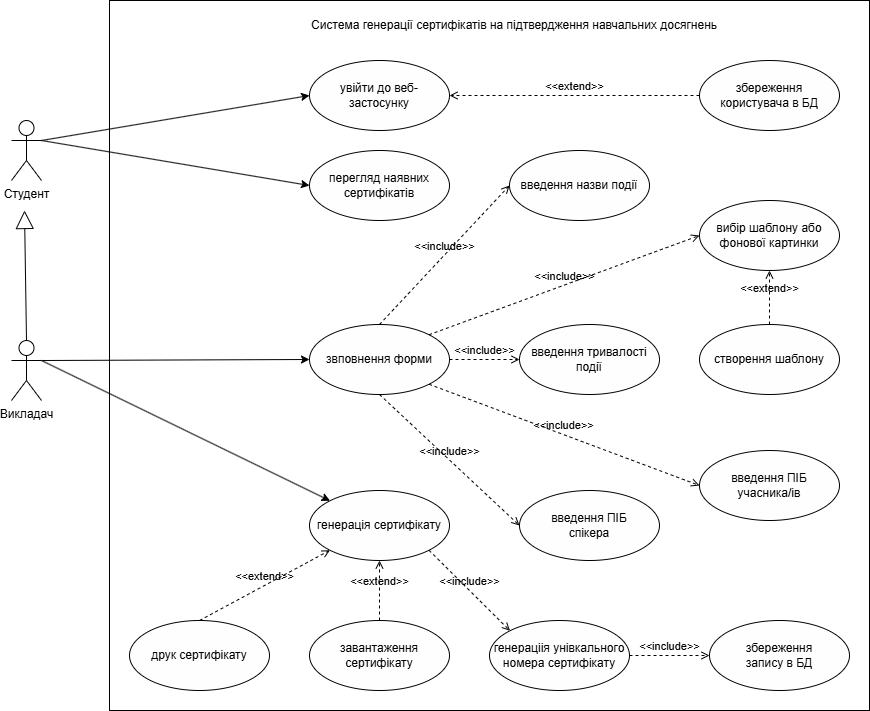


Рисунок 2.1 – Діаграма прецедентів створюваного веб-застосунку

Окрім функціональних вимог, представлених на діаграмі, слід також перелічити нефункціональні вимоги до веб-застосунку для генерації сертифікатів:

* привабливий та зручний інтерфейс користувача;
* адаптивний дизайн для коректного відображення на різних пристроях;
* висока швидкість роботи веб-застосунку навіть при великій кількості одночасних користувачів;
* захист персональних даних користувачів та забезпечення конфіденційності інформації про сертифікати;
* стійкість до збоїв та відновлення роботи після відмови;
* масштабованість системи для можливості розширення функціональності та обробки більшої кількості даних у майбутньому;
* відповідність стандартам з безпеки веб-застосунків;
* простоту встановлення, використання та підтримки.

## 2.2 Опис прецедентів системи

Прецедент: увійти до веб-застосунку

Актори: Студент і Викладач

Передумова: користувач має доступ до Інтернету та URL адресу веб-застосунку.

Післяумова: користувач успішно увійшов до веб-застосунку та отримав доступ до його функціональності.

Сценарій:

1. Користувач відкриває веб-браузер на своєму пристрої.
2. Користувач вводить URL адресу веб-застосунку у адресному рядку браузера.
3. Система переспрямовує користувача на сторінку входу.
4. Користувач вводить свої облікові дані (логін та пароль) у відповідні поля на сторінці входу.
5. Система перевіряє правильність введених даних.
6. Якщо введені дані правильні, система авторизує користувача і перенаправляє його на головну сторінку веб-застосунку.
7. Користувач отримує доступ до функціональності веб-застосунку, залежно від його ролі та прав доступу.
8. Якщо введені дані неправильні, система відображає повідомлення про помилку та пропонує спробу увійти ще раз.

Прецедент: заповнити реєстраційну форму

Актор: Викладач

Передумова: користувач має доступ до системи і бажає зареєструвати нову подію.

Післяумова: інформація про подію та учасників успішно збережена у системі.

Сценарій:

1. Користувач увійшов до системи.
2. Користувач вводить назву події у відповідне поле.
3. Користувач обирає потрібний шаблон для події із доступних варіантів або обирає власний.
4. Користувач вводить тривалість події у відповідне поле.
5. Користувач вводить ПІБ усіх учасників події у відповідні поля або додає їх як окремі записи.
6. Користувач вводить ПІБ того, хто проводив захід, у відповідне поле.
7. Система зберігає введену інформацію та підтверджує успішну реєстрацію події.

Прецедент: згенерувати сертифікат

Актор: Викладач

Передумова: користувач вже заповнив всі необхідні дані.

Післяумова: сертифікат успішно згенеровано, і користувач має можливість завантажити його або роздрукувати.

Сценарій виконання:

1. Після заповнення форми, користувач підтверджує створення сертифікату.
2. Користувач має можливість завантажити згенерований сертифікат у форматі PDF або роздрукувати його прямо з веб-застосунку.

Прецедент: збереження користувача в базі даних

Передумова: користувач зайшов на веб-застосунок та зареєструвався в ньому.

Післяумова: користувач збережений в базі даних з усіма необхідними даними.

Сценарій:

1. Система отримує дані, що були введені користувачем під час реєстрації.
2. Система перевіряє ці дані на коректність та відповідність вимогам до формату.
3. Якщо дані введені коректно, система зберігає їх в базі даних разом з індивідуальним ідентифікатором користувача.
4. Система надсилає підтвердження про успішну реєстрацію користувачеві.
5. Якщо дані не коректні, система повідомляє користувача про помилку та просить відредагувати їх.

Прецедент: генерація унікального номеру сертифікату та збереження запису про нього у базі даних

Передумова: система отримує запит на створення нового сертифікату.

Післяумова: система успішно генерує унікальний номер сертифікату і зберігає запис про нього у базі даних.

Сценарій:

1. Система отримує запит на створення нового сертифікату.
2. Система генерує унікальний номер сертифікату.
3. Система зберігає запис про сертифікат разом з його унікальним номером у базі даних.

Слід відмітити, що тут описані тільки основні прецеденти, зображені на діаграмі прецедентів системи.

## 2.3 Проєктування інтерфейсу користувача

Проєктування інтерфейсу користувача – це ключовий етап у розробці будь-якого програмного продукту або веб-застосунку. Його мета полягає в створенні зручного, ефективного та привабливого інтерфейсу, який відповідає потребам та очікуванням користувачів.

Розглянемо основні особливості та методи проєктування інтерфейсу, інструменти, що використовуються для його реалізації, а також рекомендації, які допоможуть створити зручний користувацький досвід.

Графічний інтерфейс користувача (GUI) має свої переваги та недоліки, які варто враховувати при його розробці та використанні.

Переваги графічного інтерфейсу користувача:

Інтуїтивність: GUI заснований на візуальних елементах, таких як іконки, кнопки та меню, що робить його доступним та легким для користувачів з різним рівнем технічних знань. Візуальна репрезентація допомагає користувачам швидко розуміти, які функції та опції доступні.

Зручність: GUI надає користувачам зручність взаємодії з комп'ютером. Вони можуть виконувати дії шляхом простого клацання мишею, перетягування об'єктів та використання клавіатури для швидкого введення даних.

Візуалізація: GUI дозволяє відображати інформацію за допомогою графічних елементів, що дозволяє користувачам більш ефективно сприймати та аналізувати дані. Графіки, діаграми, таблиці та інші візуальні елементи допомагають зрозуміти складну інформацію швидше та ефективніше.

Мультимедіа: GUI підтримує інтеграцію мультимедійних елементів, таких як зображення, відео та звук. Це дозволяє створювати більш привабливі та інтерактивні інтерфейси, що сприяє покращенню вражень користувачів.

Недоліки графічного інтерфейсу користувача:

Вимоги до апаратного забезпечення: GUI може вимагати відносно потужних комп'ютерних ресурсів для ефективної роботи. Велика кількість графічних ефектів та опцій може призводити до повільної роботи системи на застарілих або менш потужних комп'ютерах.

Навантаження на пам'ять: GUI використовує графічні елементи, які можуть займати більше місця в оперативній пам'яті порівняно з текстовим інтерфейсом. Це може призводити до збільшеного споживання пам'яті та зниження продуктивності системи.

Складність: Деякі GUI можуть бути вкрай складними та заплутаними через велику кількість функцій та опцій. Для користувачів з обмеженим досвідом або незнайомих з конкретною програмою це може бути перешкодою для ефективного використання інтерфейсу.

Залежність від обладнання: GUI зазвичай розробляється для конкретної платформи або операційної системи, що може призводити до обмежень у використанні на інших пристроях або платформах.

Усі ці фактори варто враховувати при проєктуванні та використанні графічного інтерфейсу користувача, забезпечуючи його оптимальну ефективність та зручність для користувачів [10].

Для створення прототипу інтерфейсу було використано сервіс Figma.

Figma – це хмарний інструмент для проєктування та створення прототипів, який дозволяє виконувати проєкти в реальному часі.

Він поєднує в собі можливості інтерфейсу користувача (UI) і взаємодії з користувачем (UX), що дозволяє дизайнерам, розробникам та іншим членам команди співпрацювати в режимі реального часу.

Замість того, щоб зберігати файли локально, усі проєкти зберігаються в хмарі, що робить їх легкими для спільного використання та доступними з будь-якого місця, де є підключення до Інтернету.

Figma підтримує багатокористувацький режим, тобто кілька людей можуть редагувати один і той самий файл одночасно, переглядаючи зміни в режимі реального часу, сприяючи кращій співпраці та спілкуванню між членами команди.

Крім того, Figma інтегрована з іншими інструментами та платформами, такими як Slack і Jira, що дозволяє легко інтегрувати процеси проєктування в існуючі робочі процеси.

Цей інструмент підтримує прототипування з інтерактивними елементами, дозволяючи перевіряти ідеї та взаємодії перед їх реалізацією.

Інструмент пропонує багато плагінів і додаткових функцій, які розширюють його можливості та дозволяють налаштувати його відповідно до конкретних потреб проєкту чи команди.

Особливо слід відзначити такий плагін, використаний нами під час роботи над продуктом, як Wireframe – це один з найпопулярніших плагінів для Figma, що містить понад 350 графічних елементів для веб та мобільних інтерфейсів. Він дозволяє швидко створювати схематичні макети сторінок чи екранів застосунку (вайрфрейми) без необхідності використовувати додаткові ресурси. Якщо ж мова йде про проєктування в процесі мозкового штурму, можна порадити також скористатись плагіном Hand Wireframe, який перетворює дизайн на стилізовані рисунки, виконані «від руки».

З налаштувань ми використовували тільки темну тему інтерфейсу та параметри експорту – обрали формат PNG для експорту готових зображень.

Створений прототип інтерфейсу програмного продукту показано на рисунках 2.2-2.7.

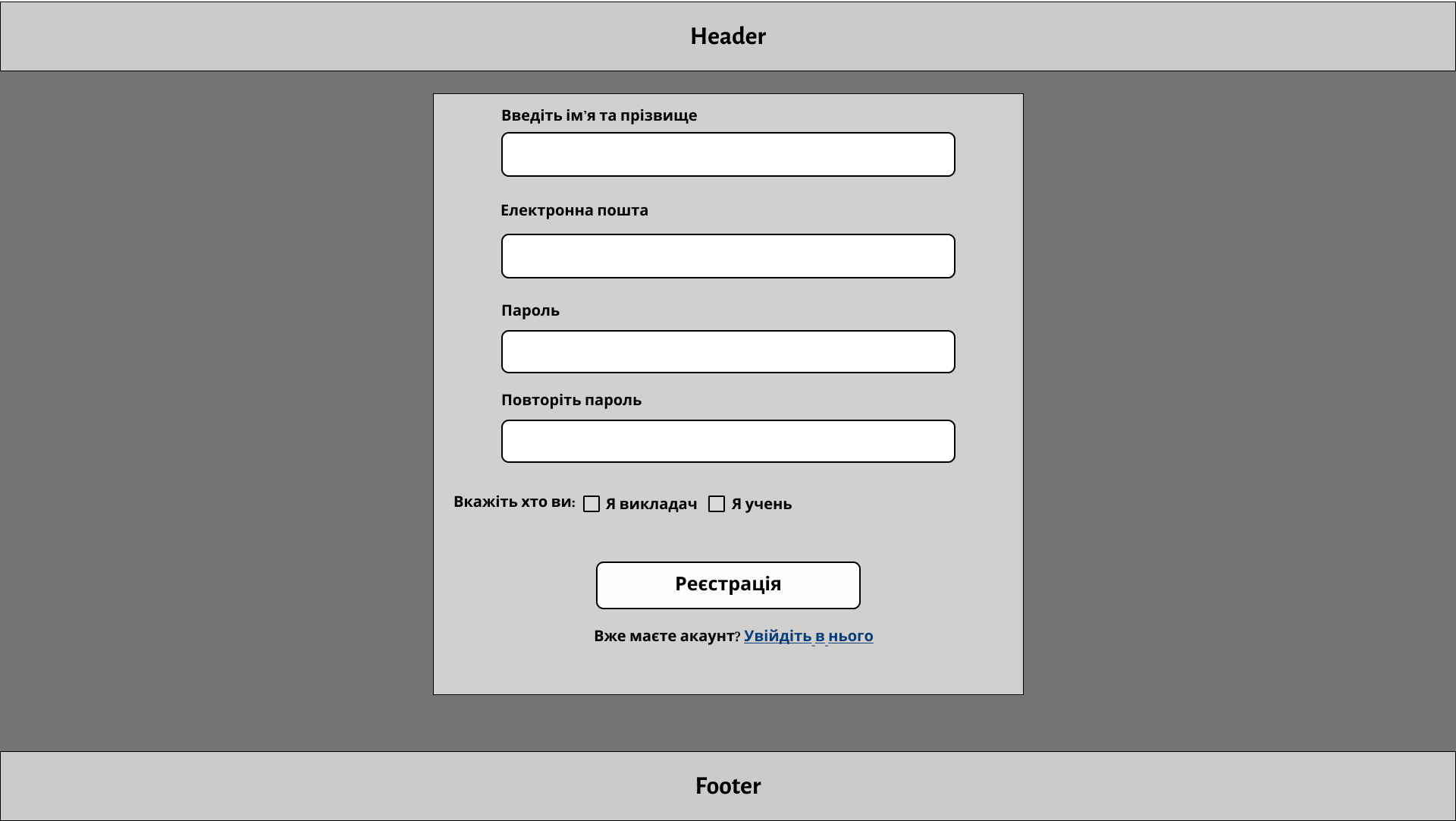


Рисунок 2.2 – Прототип інтерфейсу сторінки реєстрації

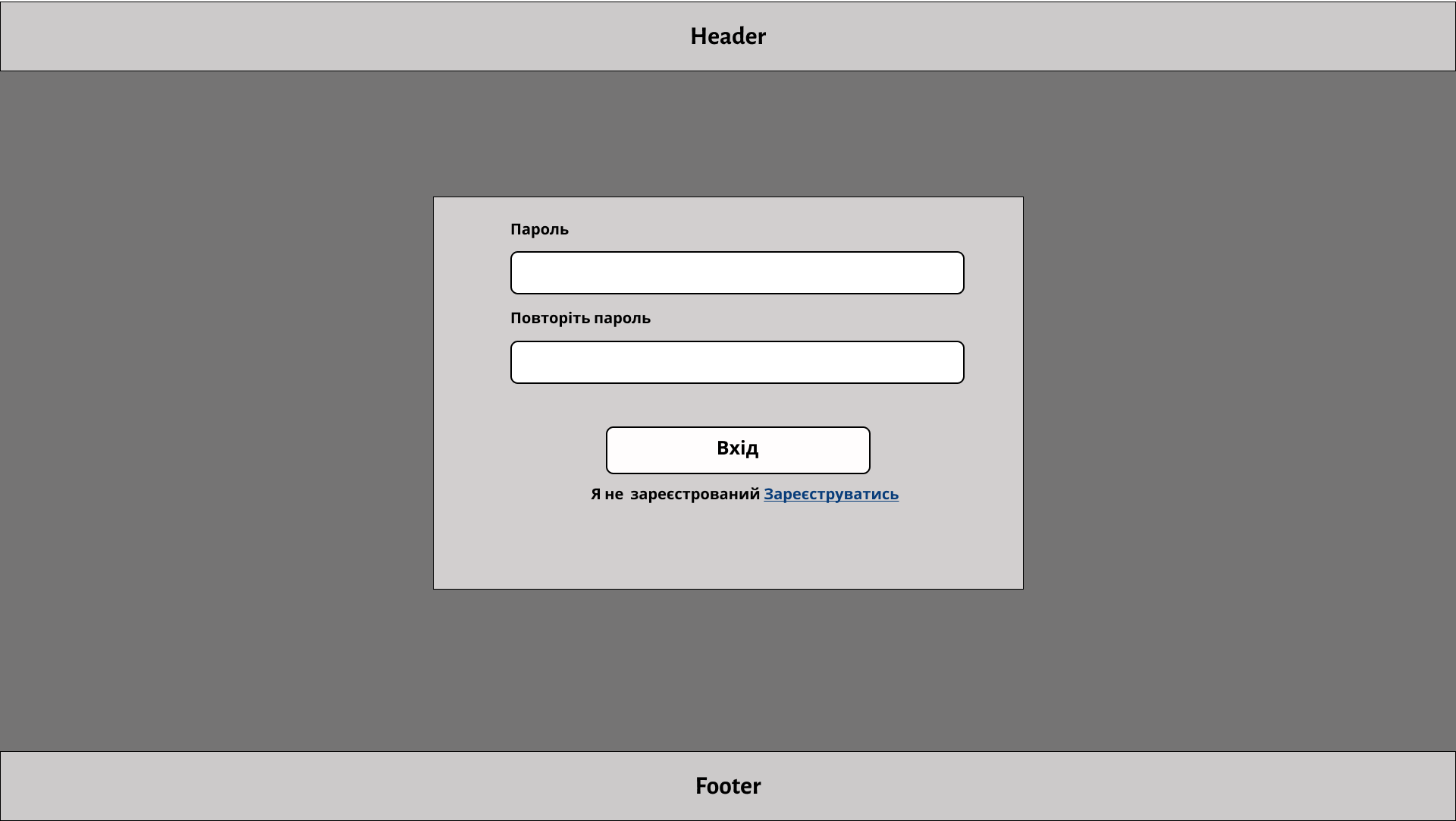


Рисунок 2.3 – Прототип інтерфейсу сторінки входу

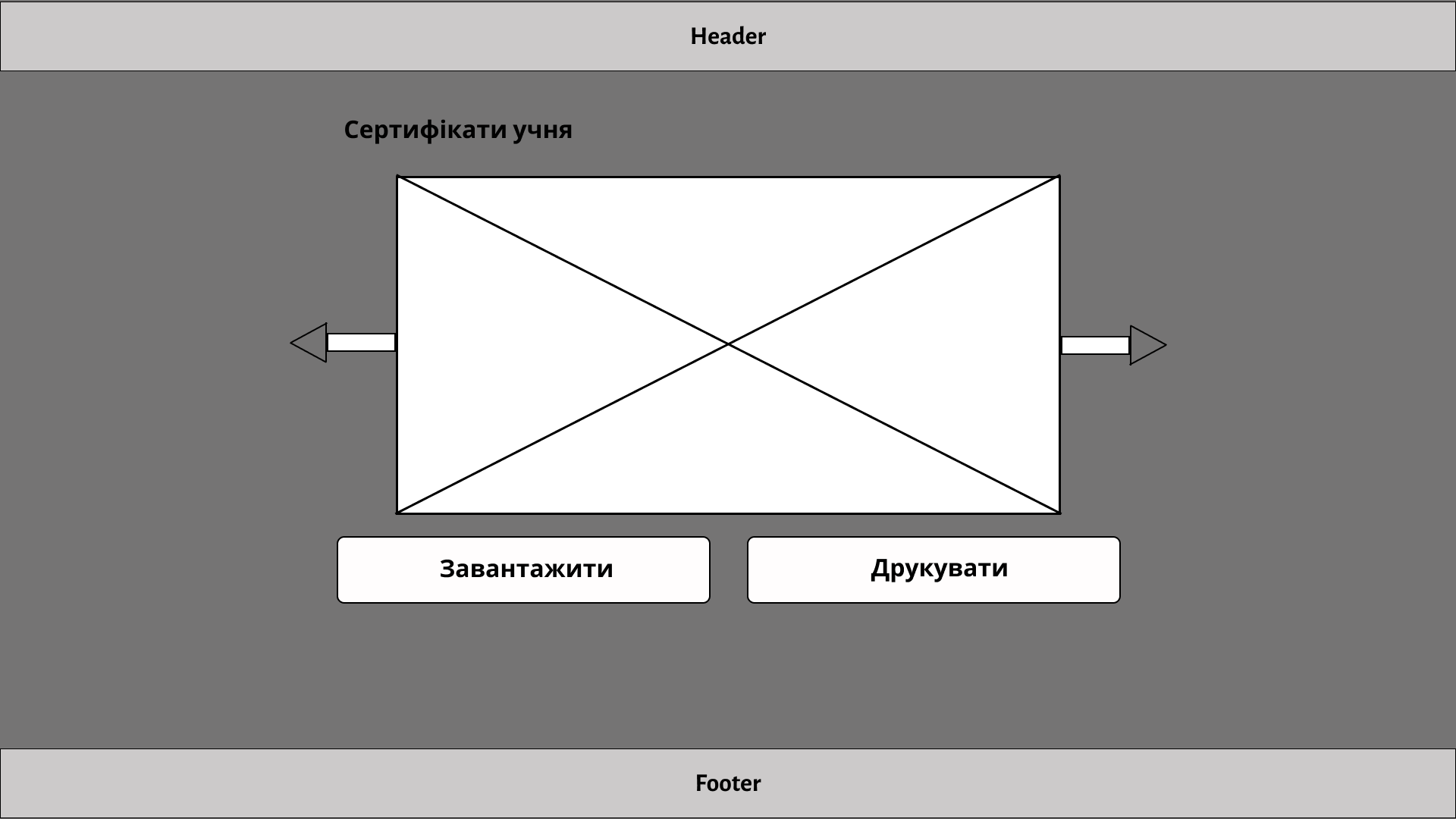


Рисунок 2.4 – Прототип інтерфейсу сторінки учня

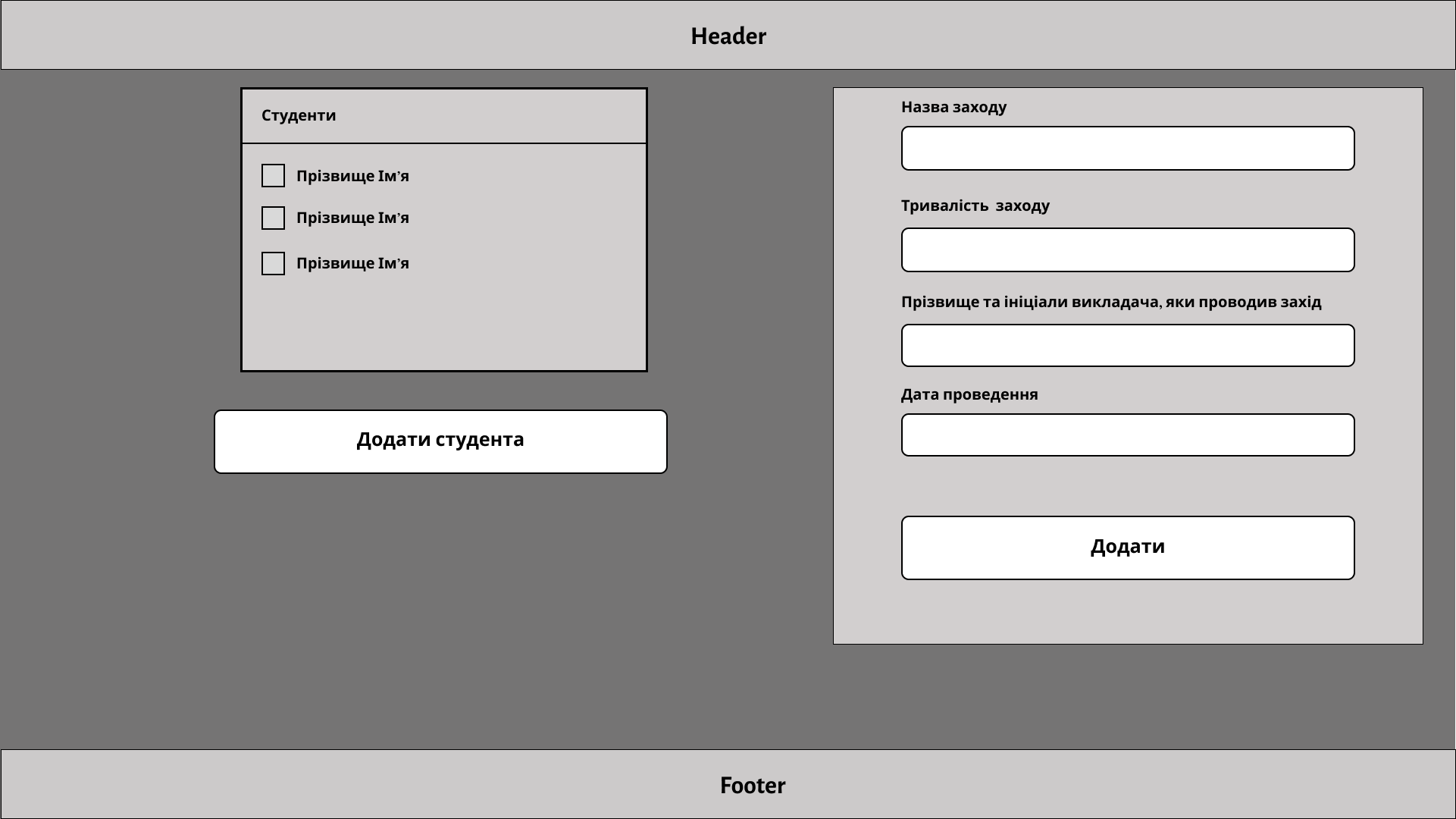


Рисунок 2.5 – Прототип інтерфейсу заповнення даних про сертифікат

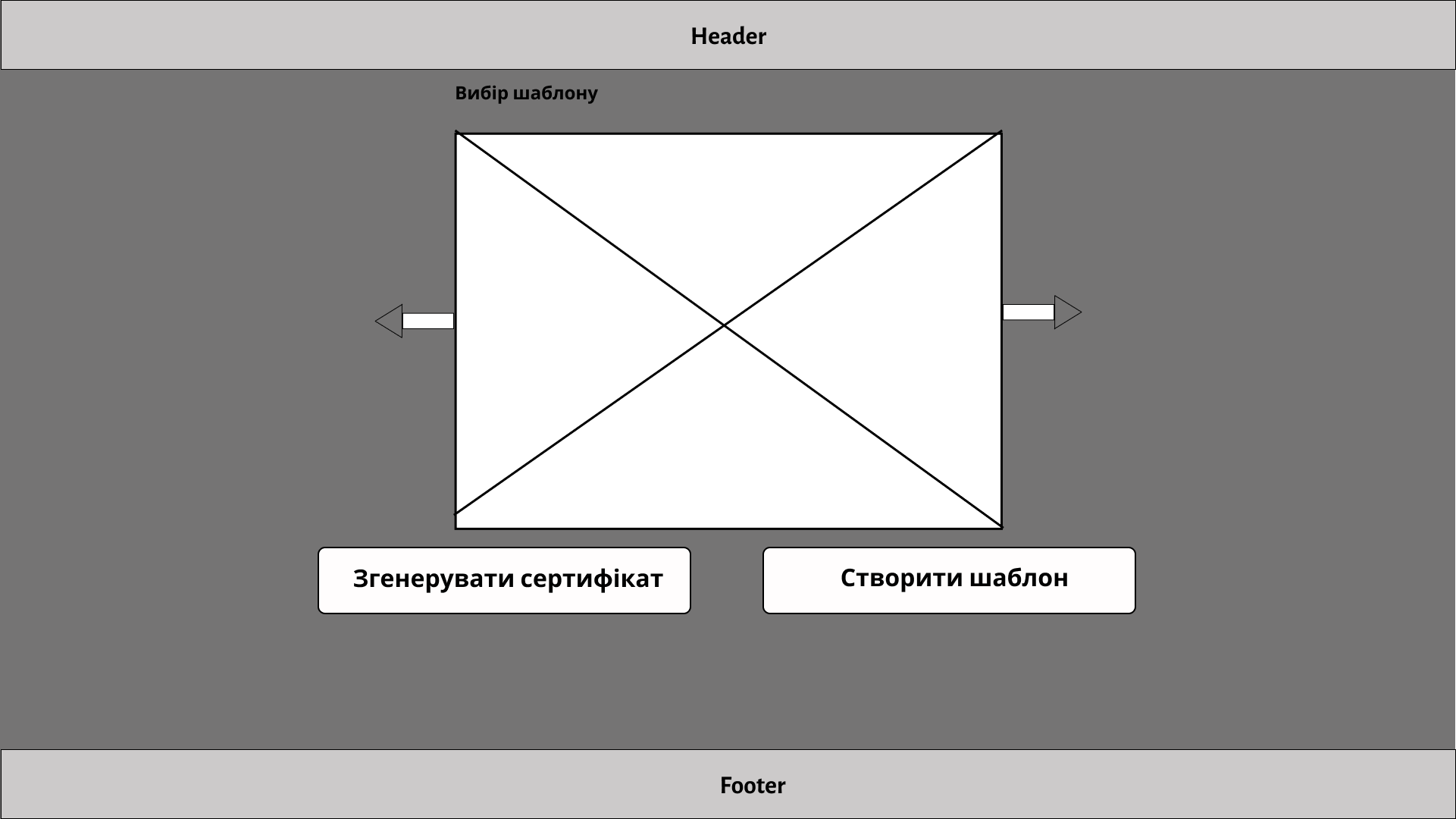


Рисунок 2.6 – Прототип інтерфейсу генерації сертифікату (Частина 1)

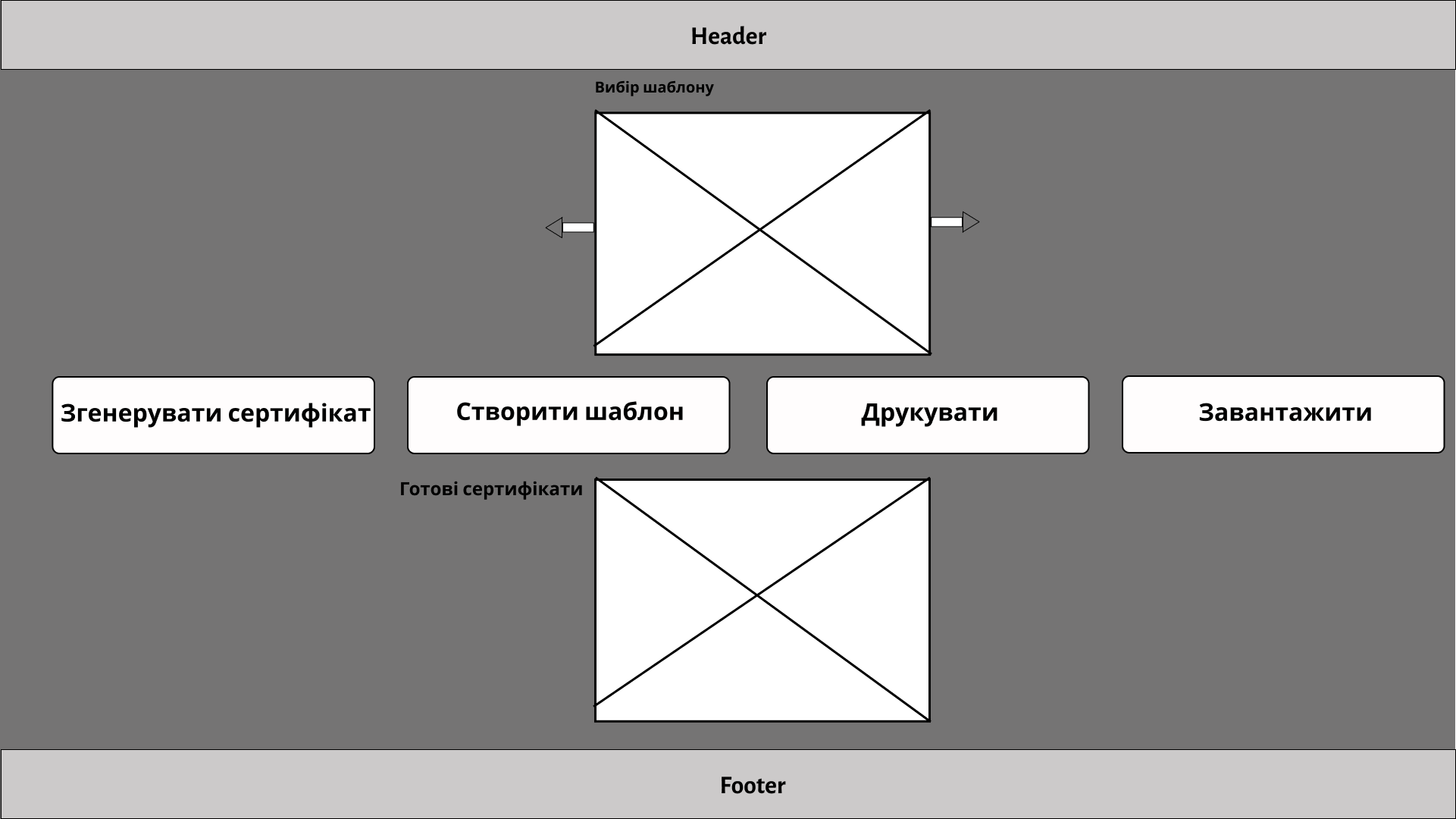


Рисунок 2.7 – Прототип інтерфейсу генерації сертифікату (Частина 2)

## 2.4 Технологічні аспекти проєктування інтерфейсу

Технологічне проєктування інтерфейсів є критичним етапом у розробці будь-якого програмного забезпечення чи системи. Його мета – забезпечити ефективний, зручний та інтуїтивний спосіб взаємодії між користувачем і системою.

Першим кроком у технологічному проєктуванні інтерфейсу є збір та аналіз вимог до нього. Це включає в себе розуміння цільової аудиторії, їх потреб, можливостей та обмежень. Вимоги також повинні враховувати контекст використання системи, наприклад, чи буде інтерфейс використовуватися на мобільних пристроях, десктопах або інших платформах.

Створюваний застосунок призначений для використання в першу чергу на десктопних комп’ютерах, тож його інтерфейс, хоч і є адаптивним, проєктувався з прицілом на десктопні браузери і керування за допомогою миші й клавіатури.

Вибір правильних технологій для створення інтерфейсу визначається великою мірою вимогами проєкту. Наприклад, для веб-інтерфейсів можуть використовуватися HTML, CSS та JavaScript, тоді як для мобільних додатків - Java або Kotlin для Android і Swift для iOS. При виборі технологій слід також враховувати фактори, такі як продуктивність, масштабованість, підтримка та безпека. Для розробки інтерфейсу веб-застосунку для генерації сертифікатів на підтвердження навчальних досягнень обрано мову JavaScript і фреймворк Vue.js з бібліотекою компонентів користувацького інтерфейсу Vuetify, що дозволяє декларативно програмувати користувацькі інтерфейси будь-якої складності на основі готових компонентів (шаблонів), спираючись на HTML та CSS.

Методи проєктування інтерфейсу включають в себе використання кращих практик у дизайні для створення зручних та естетичних інтерфейсів, які легко розуміються користувачами. Важливим аспектом є також увага до деталей, таких як розташування елементів керування, використання кольорів та шрифтів і реалізація анімації для покращення враження від користування. Інтерфейс створеного застосунку проєктувався з врахуванням принципу розумного мінімалізму, який забезпечує компроміс між зручністю і простою.

Тестування інтерфейсу є критичним етапом у процесі розробки, оскільки воно дозволяє виявити та виправити помилки та недоліки, які можуть виникнути під час взаємодії користувача з системою. Тестування може включати в себе ручне та автоматизоване тестування, тестування з участю реальних користувачів та збір зворотного зв'язку для подальшого вдосконалення інтерфейсу. Розроблений веб-застосунок тестувався за допомогою фреймворку Jest.

Технологічне проєктування інтерфейсу вимагає уваги до деталей, розуміння потреб користувачів та використання відповідних технологій. Здійснення ефективного зв'язку між користувачем та системою є ключовим для створення успішного продукту, який задовольнить потреби своїх користувачів. Враховуючи вищезазначені аспекти, розробники можуть створити інтерфейси, які забезпечать зручну та приємну взаємодію з програмним забезпеченням чи системою. Саме такого результату ми прагнули досягти, проєктуючи інтерфейс веб-застосунку для генерації сертифікатів на підтвердження навчальних досягнень.

## 2.5 Висновки за розділом

У розділі визначено вимоги до програмного продукту, що є веб-застосунком для генерації, експорту та друку сертифікатів про проходження тренінгів, участь у вебінарах та заходах. Були виявлені користувачі системи, такі як студенти (звичайні користувачі – слухачі, відвідувачі подій), та викладачі (спікери, організатори подій), які можуть створювати та редагувати сертифікати.

Основними прецедентами є:

* створення сертифікату;
* експорт сертифікату;
* друк сертифікату;
* автоматична генерація унікального номера сертифікату та збереження запису про нього у базі даних;
* збереження записів про сертифікати у базі даних;
* збереження записів про користувачів у базі даних.

В результаті аналізу, виконаного під час написання цього розділу, виявлені вимоги візуалізувано за допомогою діаграми прецедентів UML, де кожен прецедент пов’язаний з відповідним актором і відображає взаємодію між користувачами та системою.

Також здійснене концептуальне проєктування інтерфейсу з урахуванням технологічних аспектів, в результаті отримано набір прототипів інтерфейсу користувача веб-застосунку.

# 3 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА БАЗИ ДАНИХ ПРОДУКТУ

## 3.1 Опис архітектури створеного продукту. Опис класів

При розробці застосунку для генерації сертифікатів був застосований один з найпоширеніших підходів до побудови архітектури веб-застосунків, а саме клієнт-серверна модель. Відповідно, програмний продукт складається з кількох основних компонентів, рознесених між клієнтською (Frontend) та серверною (Backend) частинами.

Спочатку опишемо Frontend-частину даного проєкту. Перш за все метою створення цього компоненту є розробка привабливого та інтуїтивно зрозумілого інтерфейсу для користувача. Використано сучасні технології та практики дизайну для забезпечення адаптивності до різних пристроїв та розмірів екранів. Не менш важливою є реалізація функцій для вводу необхідних даних, вибору шаблону сертифікату, попереднього перегляду, а також експорту або друку сертифікату. Також клієнтська частина відповідає за реалізацію механізмів аутентифікації користувачів і контроль доступу до функціональності за різними ролями (викладач, студент).

Backend-частина проєкту була розроблена для реалізації бази даних для зберігання інформації про користувачів, шаблонів сертифікатів та створених сертифікатів. Також її використання було необхідним для реалізації логіки для автоматичної генерації унікальних номерів сертифікатів на основі введених даних та обраного шаблону. І не менш важливим аспектом була реалізація API для взаємодії між Frontend та Backend частинами. Це дозволить в майбутньому розширити функціональність застосунку та інтегрувати його з іншими системами, якщо потрібно.

Також важливим етапом був вибір хостинг-провайдера для розміщення веб-застосунку та його публікації в Інтернеті, а також забезпечення захисту, конфіденційності та цілісності даних, і захисту від потенційних кібератак.

Далі перейдемо до опису класів. Але перш ніж описати класи, що наявні в проєкті, хотілось би нагадати, що таке взагалі класи.

В ООП клас – це шаблон, за яким створюються об’єкти. Він допомагає групувати разом пов’язані змінні та функції. Коли створюється екземпляр класу, то це якраз і є той об’єкт, який був створений на його основі. А клас може використовуватися для створення безлічі об’єктів, які відрізнятимуться один від одного своїм станом (тобто значеннями полів).

Клас дає змогу створювати об’єкти за своїм унікальним шаблоном. Коли створюється клас, то йому задається набір властивостей і методів, а потім його можна використовувати для створення об’єктів, які матимуть ті самі властивості та методи. Це як Lego: створюється свій унікальний блок, а потім використовується для створення різних конструкцій. У класі можна задати властивості, що визначають стан об’єкта (колір, розмір, ім’я) і методи, що визначають, як об’єкт взаємодіятиме з іншими об’єктами та з навколишнім середовищем [11].

Програмний продукт містить такі основні класи:

* user (Користувач) – відповідає за представлення та обробку даних про користувачів, включаючи їх автентифікацію та авторизацію;
* certificate (Сертифікат) – містить інформацію про створені сертифікати, таку як унікальний номер, дані користувача, вибраний шаблон сертифікату тощо. Відповідає за генерацію та збереження даних про сертифікати;
* template (Шаблон сертифікату) – представляє різні шаблони для сертифікатів, включаючи їх дизайн та вміст. Діаграму класів для нашого застосунку показано на рисунку 3.1.

## 3.2 Опис бази даних

В проєкті було використано базу даних MongoDB. MongoDB – це NoSQL СУБД, що створена для розробки сучасних додатків. Вона гнучка, з нею просто працювати та вирішувати нестандартні завдання.

Поки реляційні (SQL) бази розкладають інформацію за жорстко пов'язаним та структурованим таблицями, MongoDB вносить дані в документи, що мають структуру, подібну до JSON. У них теж буде структура та ієрархія, але менш жорстка. Простий приклад: зберігання даних користувача у якого не один номер телефону та адреса, а декілька, без підключення розширень та інших доробок. Робота з такими об'єктами інтуїтивно зрозуміла розробникам сайтів та додатків, а сама модель зберігання добре підходить для документів з унікальними даними, які змінюються з часом, наприклад, профілів користувачів та каталогів [12].

Кожна база даних MongoDB містить колекції, які, у свою чергу, містять документи. Кожен документ може мати різну кількість полів. Розмір і зміст кожного документа можуть відрізнятися один від одного.

Структура документа більше відповідає тому, як розробники будують свої класи та об’єкти на відповідними мовами програмування. Розробники часто кажуть, що їхні класи не є рядками та стовпцями, а мають чітку структуру з парами ключ-значення.

Рядки (або документи, як викликано в MongoDB) не потребують попередньої визначеної схеми. Натомість поля можна створювати на льоту.

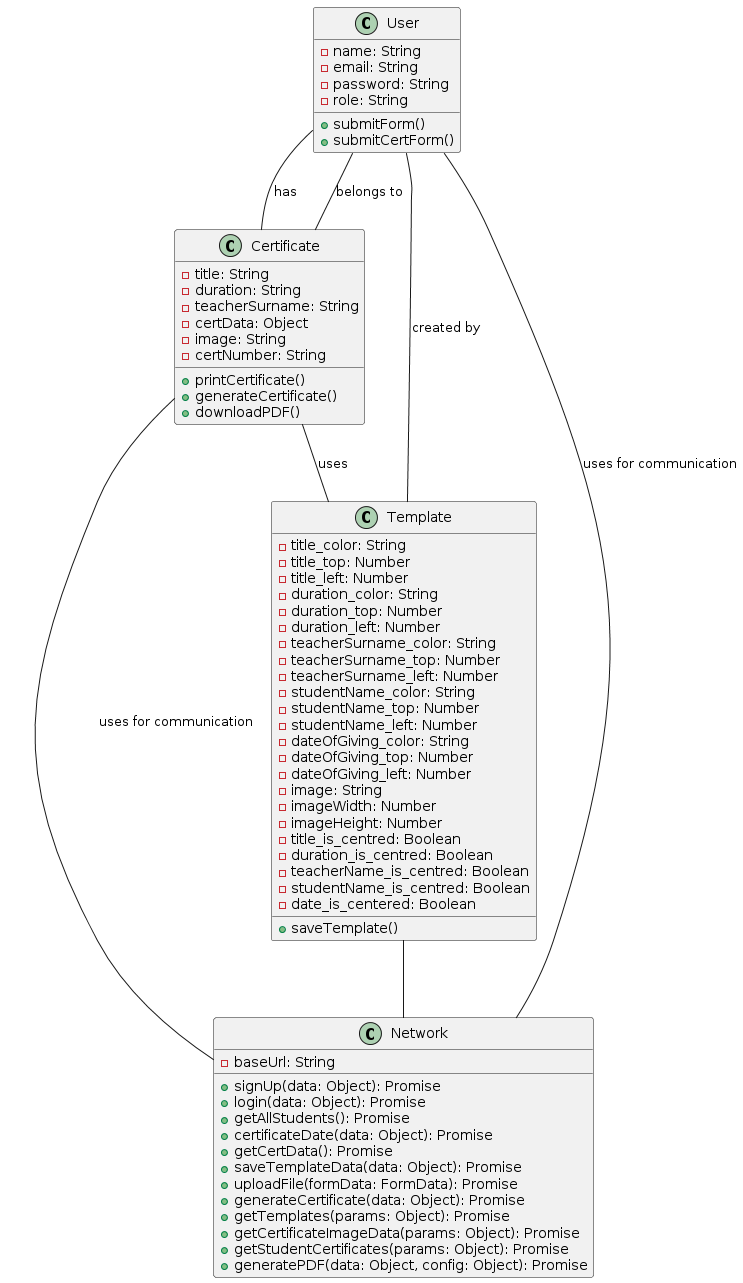


Рисунок 3.1 – Діаграма класів створюваного веб-застосунку

MongoDB-середовища дуже масштабовані. Компанії в усьому світі зазначають, що деякі з них працюють з понад 100 вузлами з мільйонами документів у базі даних [13].

Порівняння структури реляційних БД з MongoDB наочно продемонстровано на рисунку 3.2.

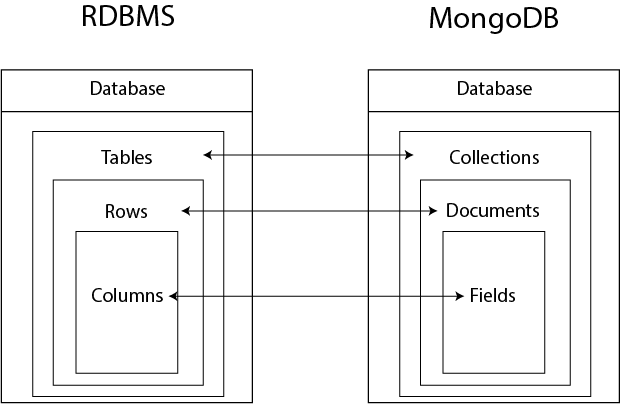


Рисунок 3.2 – Структура реляційних БД з MongoDB

Оскільки як сказано вище, кожна БД в MongoDB містить колекції, в нашому проєкті їх три. Перша колекція «users» в якій збережені всі дані про користувачів такі як name, email, password і role.

Наступна колекція містить інформацію про сертифікати, а саме «certificates», в ній присутня така інформація як унікальний номер сертифікату, кому виданий, ким виданий, дата видачі, тривалість та назва заходу за який видано сертифікат.

І остання колекція це «certificate-templates», в якій збережені базові шаблони сертифікату, а також якщо певний користувач (викладач) завантажить свій шаблон, то він теж автоматично збережеться у базі даних. Також у базі даних наявний зв’язок, який забезпечує при створені викладачем сертифікату для певного користувача системи (студента), автоматичне його збереження в колекції з сертифікатами на ім’я даного студента, тож студент матиме змогу переглядати його в своєму кабінеті.

Структуру бази даних та зв’язки між колекціями зображено на ER-діаграмі (рисунок 3.3).

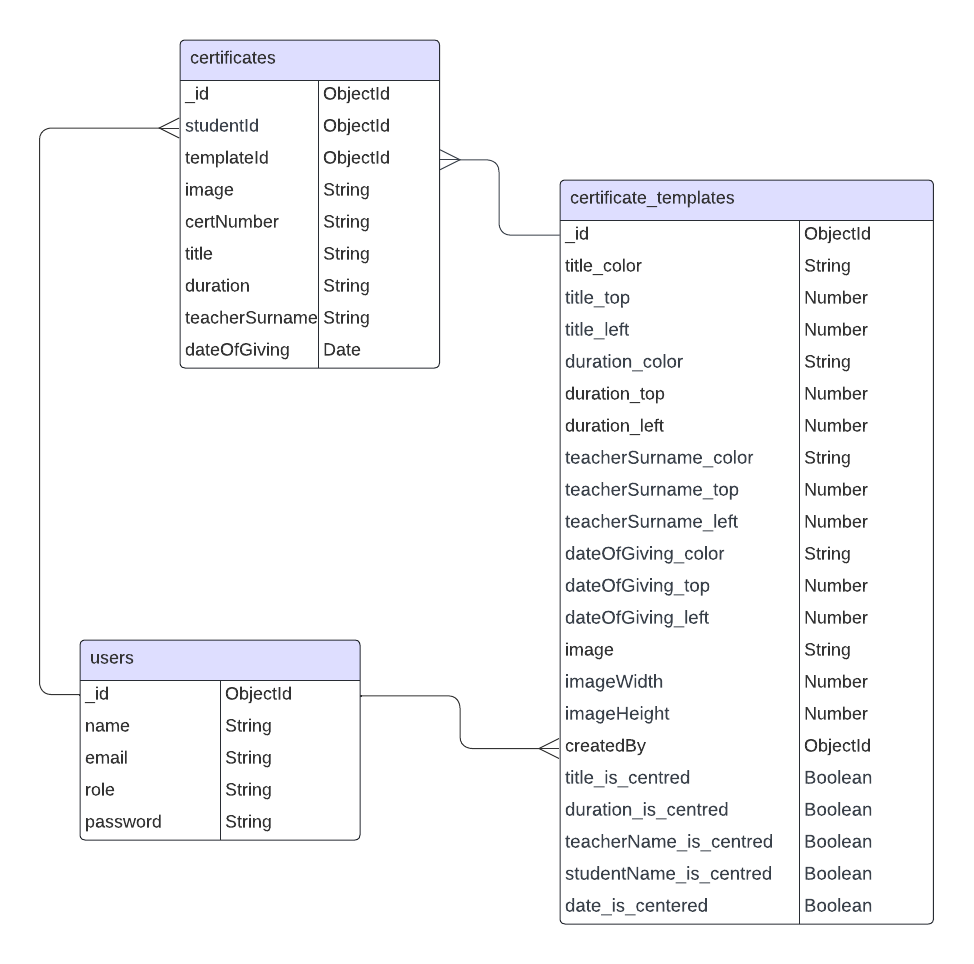


Рисунок 3.3 – ER-діаграма бази даних створюваного веб-застосунку

## 3.3 Особливості розгортання програмного продукту

Розгортання програмного продукту особливостями технологій Node.js та Vue.js.

Розгортання Node.js-сервера потрібне для обробки запитів до веб-застосунку. Це включає встановлення необхідних пакетів, конфігурування середовища виконання, налаштування бази даних та інші необхідні для середовища налаштування.

Vue.js-додаток має бути розгорнутий на веб-сервері, аби бути доступним для користувачів через веб-браузер. Це може бути здійснене за допомогою різних методів, включаючи використання CDN для статичних файлів або розгортання на сервері з використанням веб-сервера.

Програмний продукт потребує бази даних для зберігання інформації про створені сертифікати, події та інші деталі. Налаштування та розгортання бази даних MongoDB забезпечить надійне зберігання даних.

Важливо враховувати аспекти безпеки під час розгортання для забезпечення захищеної комунікації між клієнтом і сервером, а також використання механізмів автентифікації та авторизації для захисту конфіденційності та цілісності даних.

При розгортанні слід врахувати можливість масштабування програмного продукту в майбутньому, якщо кількість користувачів зростатиме.

Після розгортання важливо встановити механізми моніторингу, які дозволять вам відстежувати продуктивність програмного продукту, виявляти проблеми та вчасно реагувати на них.

Розгортання програмного продукту "Генератор сертифікатів" вимагає уважного планування, налаштування та тестування для забезпечення стабільної та ефективної роботи системи.

Для розгортання програмного продукту використано сервіс Render [14].

Render – це сучасна платформа для хостингу, яка пропонує простий спосіб розгортання та керування веб-додатками та сервісами. Вона дозволяє розгортати як статичні, так і динамічні сайти, бекенд-сервіси і бази даних.

Автоматичні оновлення відбуваються кожного разу, коли ви вносите зміни в репозиторій на GitHub або GitLab.

Механізм візуалізації підтримує кілька мов програмування, включаючи Node.js, Python, Ruby, Go та інші. Сервіс Render забезпечує повністю інтегровану базу даних PostgreSQL і Redis, резервна копія якої автоматично створюється та збільшується.

Сертифікати SSL є автоматичними та безкоштовними, що забезпечує безпеку зв’язку між програмами. Масштабування ваших додатків легко виконати вгору або вниз залежно від навантаження.

Платформа підтримує фонові процеси, що дозволяє виконувати періодичні завдання без необхідності додаткових серверів. Крім того, Render дозволяє розгортати статичні веб-сайти, які швидко завантажуються та прості в обслуговуванні.

Переваги Render – це легкість у використанні та швидке розгортання, а також висока надійність і автоматичний моніторинг. Платформа дозволяє розробникам зосередитися на створенні продукту, а не на налаштуванні інфраструктури для нього. Діаграму розгортання нашого продукту показано на рисунку 3.4.

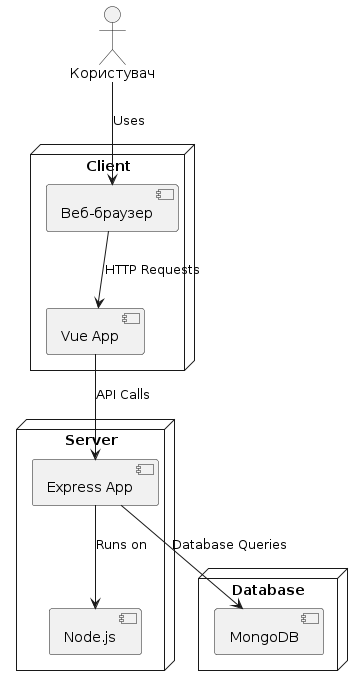


Рисунок 3.4 – Діаграма розгортання створюваного веб-застосунку

## 3.4 Висновки за розділом

В цьому розділі виконано огляд архітектури продукту, опис основних класів, що входять до його складу, структури бази даних та особливостей розгортання застосунку.

Застосунок використовує клієнт-серверну архітектуру. Його побудовано з використанням Node.js та Vue.js, у якості бази даних використано MongoDB. Веб застосунок може бути запущений локально або розгорнутий на платформі Render.

# 4 РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ ТА ОПИС РЕЖИМІВ ЙОГО РОБОТИ

## 4.1 Опис мови програмування

Для розробки продукту, використано мову програмування JavaScript та відповідні фреймворки. JavaScript – це високорівнева скриптова мова програмування, яка широко використовується для створення інтерактивних застосунків і сайтів. JavaScript додає анімацію і спливаючі повідомлення та оновлює контент і змушує інтерфейс реагувати на наші дії. Наприклад, наводиш курсор на кнопку, а вона змінює колір (або зникає після натискання). Мова JavaScript робить сторінки функціональними.

JavaScript виконується в реальному часі в середовищі браузера, без необхідності попередньої компіляції. Розробник вносить зміни в код JavaScript і відразу бачить результат.

Тип змінних у JavaScript визначається автоматично під час виконання коду. Програмістам не потрібно присвоювати його заздалегідь, що спрощує роботу. Код JavaScript може бути виконаний на різних платформах і пристроях. Автоматичне збирання сміття полегшує керування пам’яттю, автоматично видаляючи об’єкти, що не використовуються.

JavaScript підтримує основні парадигми програмування. Це дає змогу створювати складні структури даних, покращувати організацію коду, застосовувати функціональні концепції та описувати точні кроки для досягнення мети.

За допомогою таких механізмів, як promise та async/await, JavaScript ефективно опрацьовує операції, що не потребують негайного завершення без блокування інших задач.

Вбудована підтримка DOM (Document Object Model) використовується для динамічної взаємодії з елементами вебсторінки, оброблення подій і внесення змін.

Навколо JavaScript сформувалася величезна спільнота розробників, які постійно створюють нові курси, підручники, фреймворки та бібліотеки JavaScript. Саме цей факт полегшує знаходження відповідей на запитання, які виникають у процесі роботи або навчання [15].

Фреймворк (framework) – це набір інструментів, бібліотек та правил, який використовується для створення програмних продуктів. Він зазвичай являє собою структуру, яка визначає, як компоненти програми повинні взаємодіяти між собою, які шаблони використовувати для створення інтерфейсів і які методи використовувати для роботи з базами даних та іншими зовнішніми ресурсами.

Фреймворк призначений для спрощення розробки програм, оскільки він надає готові рішення для поширених завдань. Це дозволяє програмістам зосередитися на створенні бізнес-логіки програми, а не витрачати час на написання коду для вирішення спільних завдань.

Крім того, він може бути спеціалізованим для певного типу програми. Наприклад, для веб-програм або мобільних застосунків. Він також може надавати інструменти управління проєктом, такі як система контролю версій або інтеграція з іншими інструментами.

Основне завдання фреймворку **–** допомогтиправильно налаштувати робочі процеси та побудувати бізнес-логіку. Звичайно, фреймворки не є універсальним рішенням для всіх завдань і іноді може знадобитися написати додаток “з нуля”, щоб задовольнити унікальні вимоги. Однак, у більшості випадків використання фреймворків дозволяє значно прискорити та спростити процес розробки та створити більш надійний та ефективний додаток.

Люди, які не стикалися з фреймворками, часто плутають їх з бібліотеками. Але насправді це різні інструменти. Бібліотека містить у собі певний набір функцій, що дозволяє вирішити конкретне завдання з певної галузі. Наприклад, зустрічаються бібліотеки для роботи з датою або часом, HTTP-запитами   
тощо [16].

## 4.2 Використані технології та бібліотеки

Для розробки веб-застосунку для генерації сертифікатів були використані технології Node.js та Vue.js. Node.js – це однопоточне кросплатформове середовище виконання з відкритим вихідним кодом і бібліотека, яка використовується для запуску вебдодатків, написаних на JavaScript, поза браузером клієнта. Простіше кажучи, Node.js – це програмне середовище, яке дозволяє запускати програми, написані мовою Javascript, поза браузером. Історично програми, написані на Javascript, на відміну від інших мов програмування, можна було запустити лише у браузерах, які мали спеціальний вбудований движок виконання коду цієї мови. Поза браузером Javascript, можна сказати, не працював. При розробці Node.js за основу було взято рушій виконання JavaScript під назвою V8, який був створений компанією Google і використовувався в браузері Google Chrome.

Оскільки після створення Node.js Javascript код можна запустити фактично в будь-якому середовищі, за допомогою цієї бібліотеки можна написати не лише фронтенд, а й серверну частину веб-програми. Простіше кажучи, це означає, що цілі сайти тепер можуть працювати з використанням єдиного «стека», що робить розробку та обслуговування набагато швидшими та легшими, дозволяючи зосередитися на досягненні бізнес-цілей проєкту. Node.js має відкритий код, тому працювати з ним можна абсолютно безкоштовно. Його і сьогодні продовжує розвивати та покращувати глобальна спільнота розробників. Важливо розуміти, що Node.js насправді не фреймворк і бібліотека, як у випадку з традиційним програмним забезпеченням, а середовище виконання. Воно є легким, гнучким і простим у розгортанні, а всі його функції допоможуть оптимізувати та прискорити вашу програму.

Важливою особливістю Node.js є асинхронний характер. Термін асинхронний означає, що сервер, створений з використанням Node.js, не повинен чекати, поки дані повернуться, при виконанні різних внутрішніх запитів. Водночас він також має неблокуючий введення-виведення. Це означає, що кілька різних процесів можуть виконуватись паралельно, не блокуючи один одного. Обидві ці властивості роблять Node.js вкрай швидкою і забезпечують кращий інтерфейс користувача.

У Node.js всі запити є однопоточними і збираються в циклі обробки подій (Event loop). Це означає, що всі програми виконуються в одному потоці, починаючи з отримання запиту і закінчуючи виконанням необхідного завдання та надсиланням відповіді клієнту назад. Ця функція Node.js запобігає повторному завантаженню запитів і скорочує час їх обробки, що робить його більш економічним у використанні.

Node.js використовується для розробки програм найрізноманітнішого призначення. Розглянемо кілька популярних варіантів використання цієї платформи:

* чати у реальному часі;

Завдяки своїй однопоточній асинхронній природі Node.js можна використовувати для обробки повідомлень у реальному часі. Він легко масштабується і часто використовується для створення чат-ботів. Node.js також спрощує створення додаткових функцій чату, які розраховані на багато користувачів чату і push-сповіщення.

* інтернет речей (концепція IoT);

Програми IoT зазвичай містять кілька датчиків, оскільки часто відправляють невеликі фрагменти даних, які можуть накопичуватися при великій кількості запитів. Node.js в такому випадку – гарний вибір, тому що він може швидко обробляти ці одночасні запити.

* потокова передача даних;

Такі компанії, як Netflix, працюють із Node.js для потокової передачі. В основному це пов’язано з тим, що Node.js є легким і швидким. Крім того, Node.js надає власний потоковий API. Ці потоки дозволяють користувачам передавати дані швидше, ніж при використанні інших технологій.

* складні односторінкові програми (SPA);

У SPA вся програма завантажується один раз на одній HTML-сторінці. Зазвичай це означає, що у фоновому режимі виконується кілька запитів зміни компонентів на екрані користувача. Тут на допомогу приходить цикл обробки подій Node.js, оскільки він обробляє подібні запити неблокуючим чином.

* програми на основі REST API.

JavaScript може використовуватись як у зовнішньому, так і внутрішньому інтерфейсі сайтів. Таким чином, сервер може легко взаємодіяти із зовнішнім інтерфейсом через REST API, використовуючи Node.js. Node.js також пропонує такі пакети, як Express.js і Koa, які дозволяють створити вебсервер з нуля ще простіше і швидше [17].

Загалом Node.js має великі перспективи для майбутнього розвитку та використання. Його швидкість, гнучкість і масштабованість роблять його привабливим вибором для розробників і компаній, які прагнуть створювати інноваційні програми та послуги. Сьогодні сотні успішних проєктів із тривалим життєвим циклом працюють над Node.js, тому він точно не зникне найближчим часом.

Vue – це фреймворк, який працює на JavaScript, створений для розробки користувацьких інтерфейсів. Він працює на базі звичайного HTML, CSS та JavaScript, з можливостями декларативно програмувати користувацькі інтерфейси будь-якої складності на основі компонентів.

Vue розширює стандартний HTML шаблонним синтаксисом, який дозволяє декларативно задавати структуру HTML на основі стану описаного у JavaScript. Він автоматично відстежує зміни стану описаного у JavaScript і з максимальною ефективністю оновлює DOM, коли відбуваються зміни.

Vue – це фреймворк та екосистема, яка охоплює більшість функцій, необхідних для розробки інтерфейсу. Але веб додатки надзвичайно різноманітні – речі, які створюються в рамках веб додатків, можуть кардинально відрізнятися за формою та масштабом. Зважаючи на це, Vue розроблено таким чином, щоб він міг бути гнучким і адаптивним. Залежно від випадку та задач, Vue можна використовувати різними способами:

* розширення статичного HTML;
* вбудовувати як веб-компонент на будь-яку сторінку;
* створення одно-сторінкового додатку (SPA);
* додаток з рендерингом на стороні серверу (SSR);
* генерація статичного додатку (SSG);
* створення десктопних, мобільних, WebGL додатків.

У більшості проєктів Vue, що використовують інструменти збірки та потребують етапу збірки, створюються компоненти Vue, використовуючи HTML-подібний формат файлу під назвою Single-File Component (також відомий як файли \*.vue, скорочено SFC [25]). Vue SFC, як випливає з назви, інкапсулює логіку компонента (JavaScript), шаблон (HTML) і стилі (CSS) в одному файлі. SFC є однією з основних функцій Vue і рекомендованим способом створення компонентів Vue, якщо задача вимагає використання інструментів збірки [18].

В розробці веб-застосунку використовувалися різноманітні бібліотеки для реалізації різних функціональних можливостей.

Наприклад, для налаштування серверної частини використано Express.js.

Express.js – це JavaScript web-фреймворк для програмної платформи Node.js. Швидкий, гнучкий і лаконічний каркас вважається стандартом для розробки web-додатків в цьому середовищі. Express використовують для створення надійних API (наборів функцій, що організовують взаємодію між компонентами програми, а також зовнішні сервіси, наприклад, для онлайн-оплати).

Express – це безліч додаткових плагінів, що полегшують розробку, і дозволяють простіше вирішувати нестандартні завдання. Його можна використовувати для бекенд, зокрема з БД MongoDB, а також фронтенду разом з бібліотеками Vue.js, React або AngularJS [19].

Mongoose – це ODM (Object Document Mapper - об'єктно-документний відображувач). Це означає, що Mongoose дозволяє визначати об'єкти зі строго-типізованою схемою, що відповідає документу MongoDB.

Mongoose дає величезний набір функціональних можливостей для створення та роботи зі схемами. Наразі Mongoose має вісім SchemaTypes (типи даних схеми), котрі може мати властивість, що зберігається до MongoDB. Ці типи наступні:

* string;
* number;
* date;
* buffer;
* boolean;
* mixed;
* objectId (унікальний ідентифікатор об'єкта, первинний ключ, \_id);
* array.

Для кожного типу даних можна:

* зазначити значення за налаштуванням;
* зазначити функцію користувача для перевірки даних;
* зазначити, що поле необхідно заповнити;
* зазначити get-функцію (геттер), яка дозволяє вам проводити маніпуляції над даними до їх повернення у вигляді об'єкта;
* зазначити set-функцію (\* сеттер), яка дозволяє вам проводити маніпуляції над даними перед їх зберіганням до бази даних;
* визначити індекси для більш швидкого отримання даних.

Окрім цих спільних можливостей для деяких типів даних також можна налаштувати особливості зберігання та отримання даних із бази даних. Наприклад, для типу даних String можна зазначити наступні додаткові опції:

* конвертація даних до нижнього регістру;
* конвертація даних до верхнього регістру;
* обрізання даних перед зберіганням;
* визначення регулярного виразу, який дозволяє в процесі перевірки даних обмежити дозволені для зберігання варіанти даних;
* визначення переліку, який дозволяє встановити список припустимих рядків.

Для властивостей типу Number і Date можна зазначити мінімально та максимально допустиме значення.

Більшість з восьми допустних типів даних мають бути вам добре знайомі. Однак, деякі (Buffer, Mixed, ObjectId та Array) можуть викликати труднощі.

Тип даних Buffer дозволяє вам зберігати двійкові дані. Типовим прикладом двійкових даних може бути зображення чи закодований файл, наприклад, документ у PDF-форматі (формат документа що передається).

Тип даних Mixed використовується для перетворення властивості в "неперебірливе" поле (поле, в якому припустимі дані будь-якого типу). Так само, як багато розробників використовують MongoDB для різних цілей, у цьому полі можна зберігати дані різного типу, оскільки відсутня визначена структура. Обачливо використовуйте цей тип даних, оскільки він обмежує можливості, надавані Mongoose, наприклад, перевірку даних та відстежування змін сутності для автоматичного оновлення властивості при зберіганні.

Тип даних ObjectId використовується звичайно для визначення посилання на інший документ у вашій базі даних. Наприклад, якщо б у вас була колекція книг та авторів, документ книги міг би містити властивість ObjectId, що посилається на визначеного автора документа.

Тип даних Array дозволяє вам зберігати JavaScript-подібні масиви. Завдяки цьому типу даних ви можете виконувати над даними типові JavaScript операції над масивами, наприклад, push, pop, shift, slice тощо [20].

Multer **–** цепрограмне забезпечення для керування завантаженням файлів у веб-додатках Node.js, особливо тих, що базуються на фреймворку Express.

Використовуючи Multer, ви можете легко приймати файли, надіслані на сервер через запити форм HTTP (наприклад, форми завантаження файлів).

Це дозволяє встановити різні параметри, такі як максимальний розмір файлу, дозволені типи файлів тощо.

Multer автоматично обробляє завантажені файли та робить їх доступними у вашій веб-програмі, щоб ви могли виконувати їх подальшу обробку, наприклад зберігати їх на сервері або обробляти для використання пізніше.

NodeHtmlToImage – це бібліотека Node.js, яка дозволяє генерувати зображення з HTML-коду. За допомогою цієї бібліотеки можна створювати зображення з вмістом HTML сторінок або елементів та використовувати їх у вашому додатку. Зазвичай це використовується для автоматизації створення зображень із статичного контенту, як, наприклад, створення знімків електронної пошти, генерація ескізів веб-сторінок, створення мініатюр зображень тощо.

Ця бібліотека приймає HTML-розмітку як вхідні дані. Також вона надає різні налаштування для контролю розміру, формату та інших атрибутів зображення, а також можливість зберігати зображення у вигляді файлу або повертати у вигляді буфера.

PDFKit – це бібліотека для створення PDF-документів у середовищі Node.js. Вона спрощує процес створення складних, багатосторінкових документів, які можна використовувати для друку або інших цілей. API PDFKit включає як базові операції, так і високорівневі абстракції, що дозволяють створювати документи з різними рівнями складності.

API PDFKit розроблено з урахуванням простоти використання, тому створення навіть складних документів може бути настільки ж простим, як кілька викликів функцій. Можна легко додавати сторінки, текст, зображення, таблиці та інші елементи до вашого PDF-документа, налаштовуючи їх розміри, положення та стилізацію. Це дозволяє створювати документи з високою точністю та професійним виглядом за допомогою Node.js.

Bcryptjs – це бібліотека для хешування паролів у Node.js. Вона базується на алгоритмі хешування bcrypt, який є одним із найбільш надійних способів зберігання паролів у веб-додатках. Основна мета використання bcrypt або bcryptjs полягає у тому, щоб уникнути зберігання паролів у відкритому вигляді у базі даних.

Основні функції Bcryptjs включають можливість створення хешів паролів, яка робить їх важкими до розшифрування навіть у випадку, якщо база даних буде скомпрометована. Крім того, вона надає методи для порівняння введеного користувачем пароля з хешем пароля, збереженим у базі даних, для перевірки правильності введеного пароля.

Бібліотека Bcryptjs зазвичай використовується у веб-додатках для безпечного зберігання та перевірки паролів користувачів, що дозволяє забезпечити високий рівень безпеки при автентифікації.

З боку Vue було використано Vuetify.

Vuetify – це бібліотека компонентів користувацького інтерфейсу для Vue.js, яка дозволяє створювати красиві та функціональні веб-додатки за допомогою готових компонентів, включаючи елементи дизайну, анімації, макети та багато іншого. Вона базується на принципах Material Design від Google та надає широкий спектр готових компонентів для швидкої розробки веб-додатків.

Компоненти Vuetify легко налаштовувати та адаптувати під конкретні потреби проєкту. Вони пропонують безліч опцій та властивостей для зміни зовнішнього вигляду та поведінки.

Усі компоненти Vuetify адаптивні та можуть легко масштабуватися для різних пристроїв та роздільної здатності екрану.

Бібліотека підтримує створення користувацьких тем, що дозволяє легко змінювати кольори, шрифти та інші стилі компонентів відповідно до дизайну вашого додатку.

Vuetify повністю інтегрується з екосистемою Vue.js, що дозволяє використовувати його компоненти та функціональність у будь-якому проєкті Vue.js.

Ця бібліотека стала популярним інструментом для розробки веб-додатків на Vue.js завдяки своїй простоті використання, гнучкості та красивому дизайну, що робить процес розробки веб-додатків ефективнішим та простішим [21].

## 4.3 Середовище розробки

Весь процес розробки відбувався в IDE (Integrated Development Environment) Visual Studio Code [22]. Visual Studio Code (VS Code) – це потужний та популярний текстовий редактор для розробників, створений компанією Microsoft. Він став відомим завдяки своїм широким можливостям, розширюваності, легкості використання та відкритому коду.

VS Code підтримує широкий спектр мов програмування, включаючи JavaScript, TypeScript, Python, Java, C++, HTML, CSS та багато інших.

Редактор має потужні функції, такі як підсвічування синтаксису, автодоповнення, вбудований відладчик, контроль версій і багато іншого.

VS Code має широкий вибір розширень, які дозволяють налаштовувати редактор під свої потреби. Це включає теми оформлення, розширення для підтримки конкретних мов програмування, інструменти для роботи з git та багато іншого.

VS Code інтегрується з іншими інструментами розробки, такими як Git, Docker, Kubernetes, та інші, що дозволяє зручно працювати з різними аспектами розробки програмного забезпечення.

VS Code підтримується на різних операційних системах, включаючи Windows, macOS та Linux, що робить його доступним для широкого кола розробників.

Розробники можуть легко працювати з хмарними сервісами, такими як Azure, AWS, та іншими, завдяки інтеграції VS Code з цими платформами.

VS Code має активне співтовариство користувачів та розробників, яке надає підтримку, допомогу та поширення розширень та корисних порад.

Visual Studio Code є безкоштовним та має відкритий вихідний код, що означає, що його можна використовувати, модифікувати та розповсюджувати безкоштовно. Інтерфейс редактора є простим та інтуїтивно зрозумілим (рисунок 4.1), його виконано у типовому для продуктів Microsoft стилі.

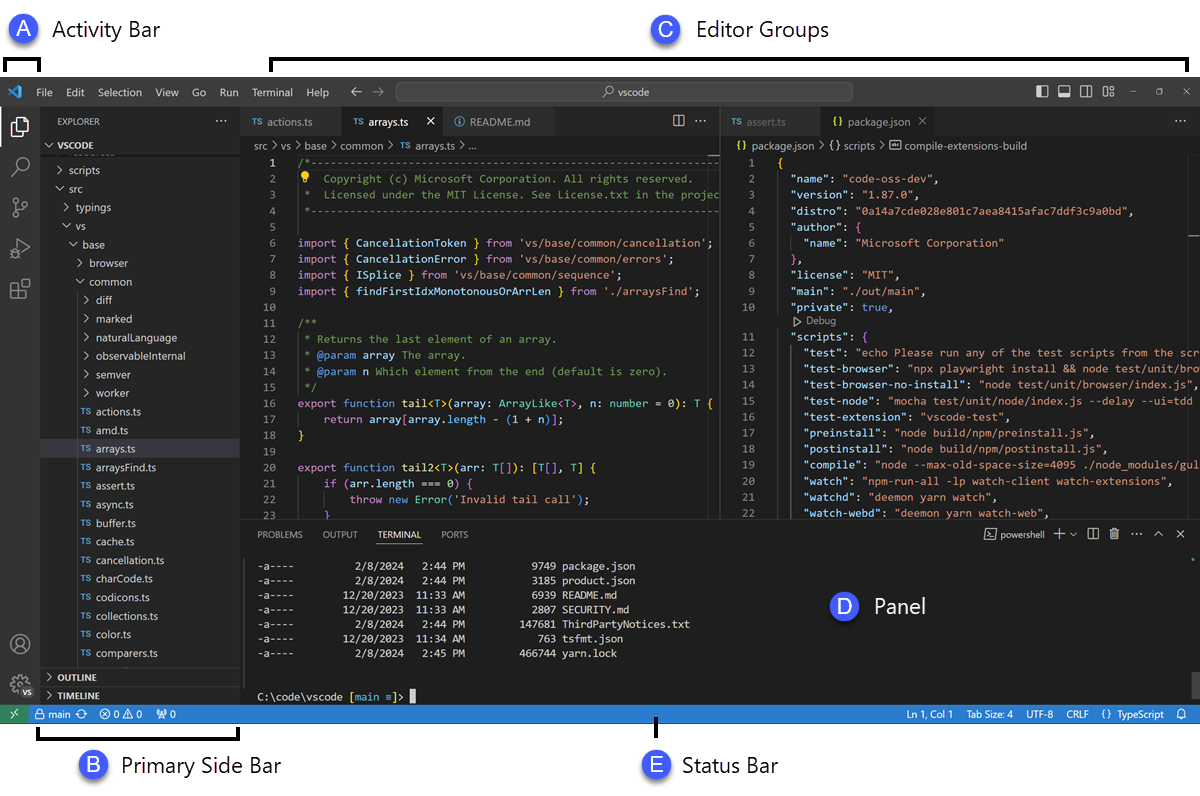


Рисунок 4.1 – Загальний вигляд Visual Studio Code

На початку роботи над кваліфікаційною роботою було прийнято рішення використати методологію Scrum для розробки веб-застосунку. Використання   
VS Code та GitHub ідеально вписується в цю методологію, забезпечуючи контроль версій та автоматизацію процесів розробки і розгортання, дозволяючи швидко реагувати на зміни, ефективно розподіляти завдання між членами команди (за умови командної розробки) та підтримувати високий рівень продуктивності на всіх етапах розробки.

Частину коду проекту було написано в GitHub Codespaces – хмарному середовищі розробки від GitHub, заснованому на VS Code.

## 4.4 Опис основних режимів роботи веб-застосунку

На рисунку 4.2 представлена сторінка входу до веб-застосунку, яка зустрічає користувача, який не авторизувався.

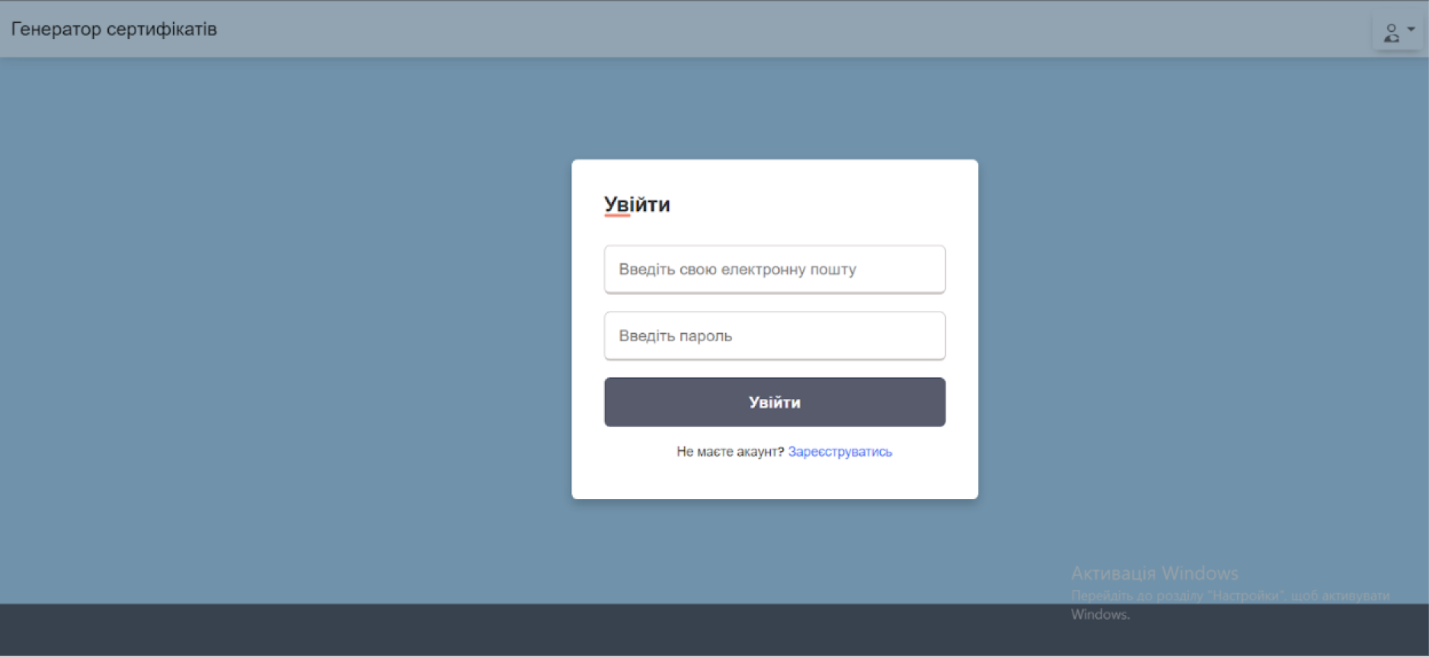


Рисунок 4.2 – Сторінка входу

Якщо користувач не має облікового запису, він може його створити, перейшовши за посиланням «Зареєструватись» в нижній частині діалогового вікна. В новому вікні (рисунок 4.3) слід надати свої дані, обрати роль і натиснути кнопку «Зареєструватись».

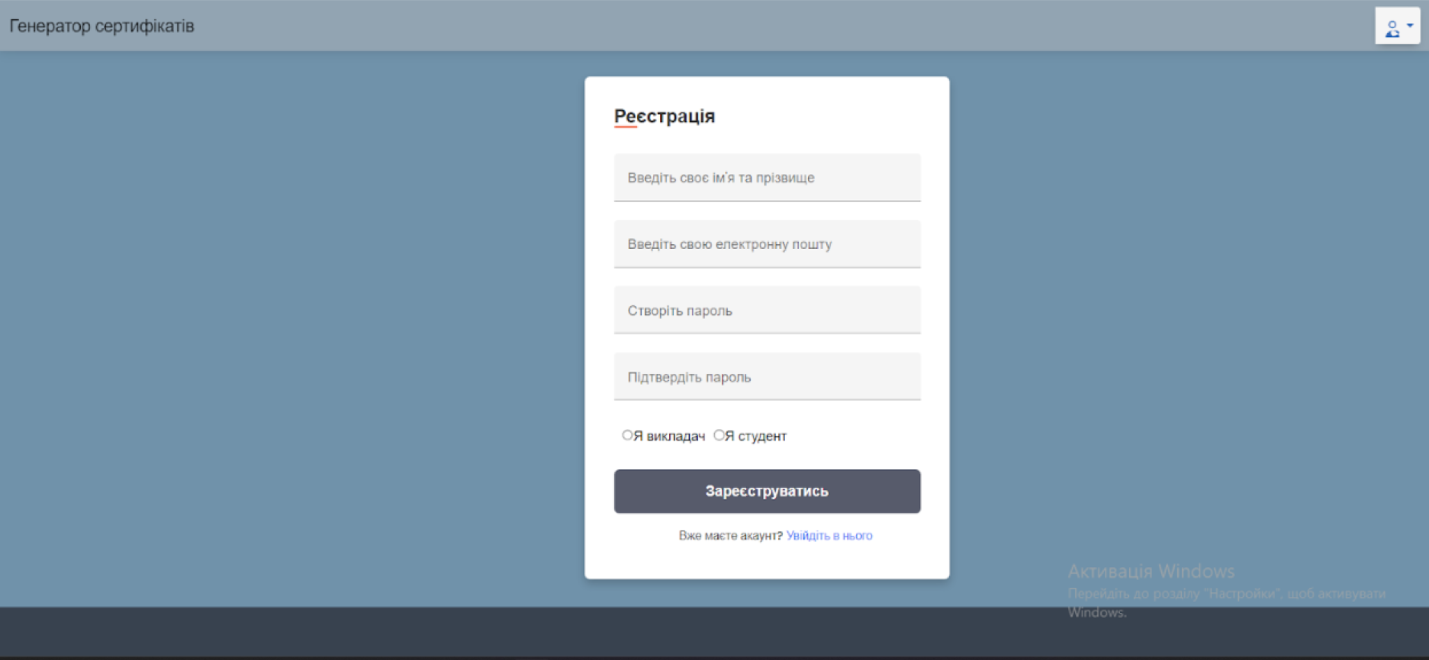


Рисунок 4.3 – Сторінка реєстрації

Якщо користувач є студентом, після входу в систему він побачить сертифікати, отримані ним раніше, і матиме змогу роздрукувати їх або завантажити в PDF-форматі (рисунок 4.4). Якщо отриманих сертифікатів кілька, вони відображаються у вигляді каруселі, і студент може гортати їх по горизонталі.

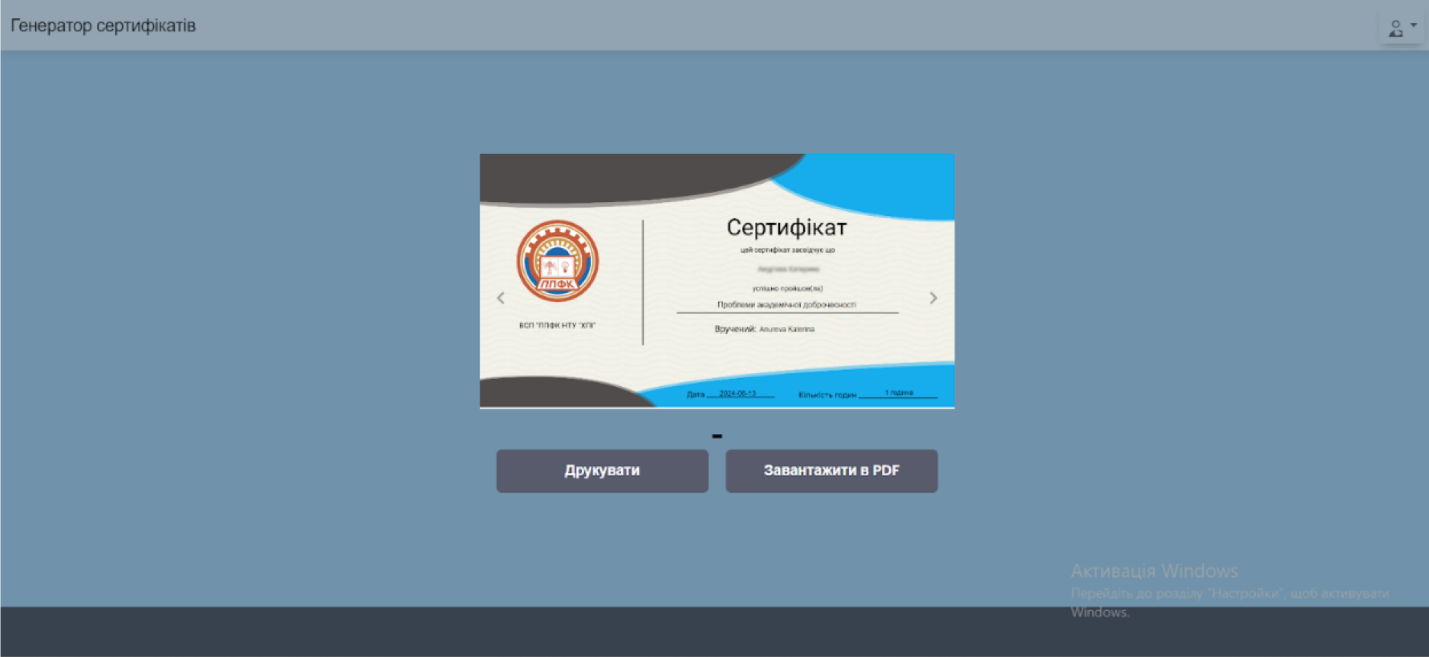


Рисунок 4.4 – Сертифікат студента

Якщо ж користувач є викладачем, він може ввести дані про подію та сертифікат, і обрати студентів, для яких треба згенерувати сертифікати   
(рисунок 4.5).

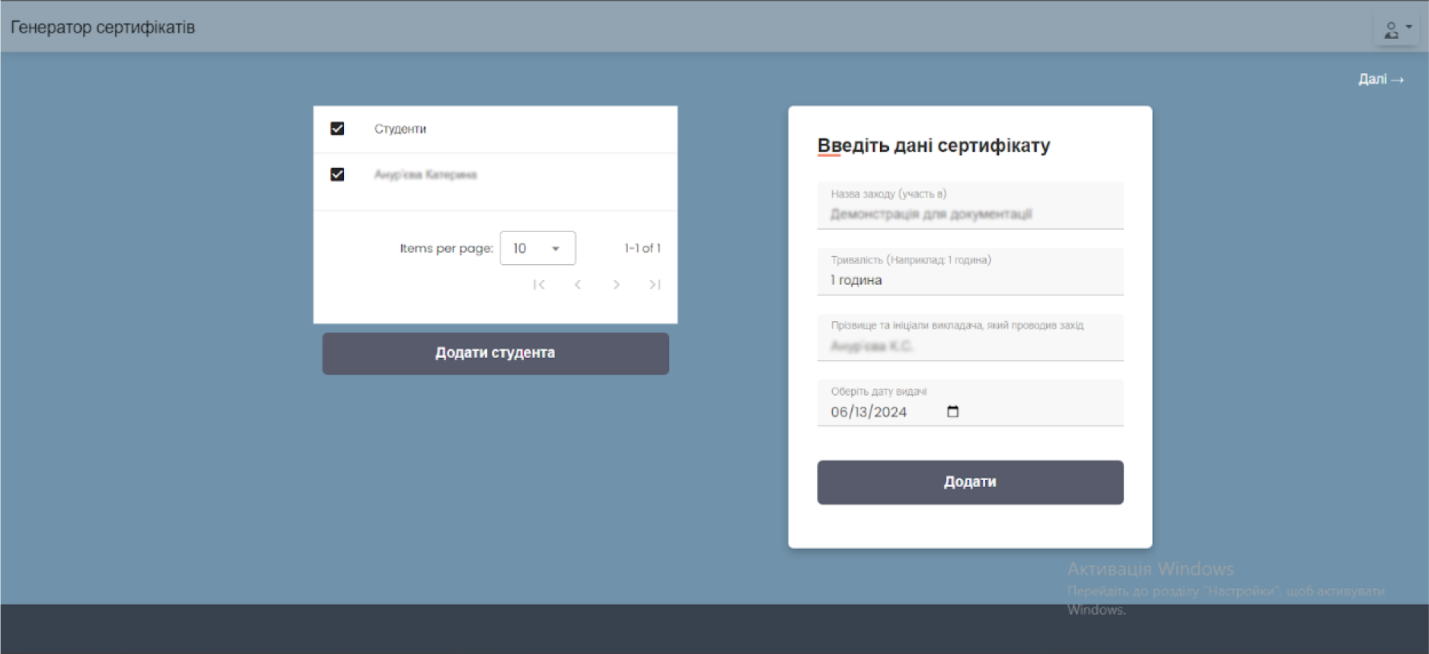


Рисунок 4.5 – Заповнення даних про сертифікат

Якщо треба додати студента до списку аби згенерувати для нього сертифікат, це можна зробити, натиснувши кнопку «Додати студента», і увівши його дані в діалоговому вікні, що з’явиться (рисунок 4.6). Список студентів зберігається в базі даних, тож при генерації сертифікатів для наступної події, викладач може скористатись вже існуючими даними.



Рисунок 4.6 – Додавання студента, якщо він відсутній в списку

Далі викладач може створити шаблон для сертифікату, обравши фонове зображення і поля документу, розташувавши їх на зображенні за допомогою перетягування (рисунок 4.7). Також є можливість обрати колір тексту. Створений шаблон зберігається в базі даних і може бути використаний в майбутньому для генерації сертифікатів для учасників наступних подій.

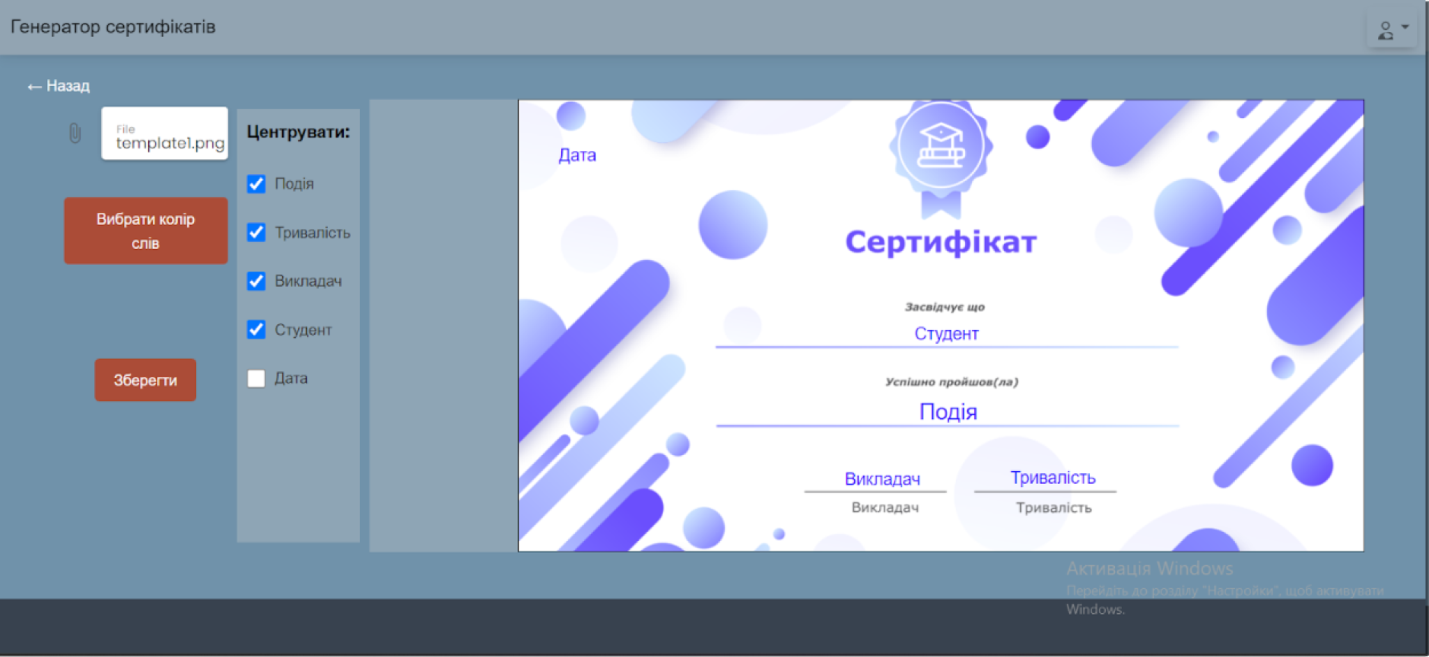


Рисунок 4.7 – Створення шаблону сертифікату

Наступний крок – згенерувати сертифікат (або сертифікати, якщо обрано кілька студентів), натиснувши кнопку «Згенерувати сертифікат» (рисунок 4.8). В дужках буде відображена кількість сертифікатів.

Якщо викладач помітив якісь помилки в шаблоні, він може повернутися до попереднього кроку і створити новий шаблон сертифікату або відредагувати обраний шаблон.

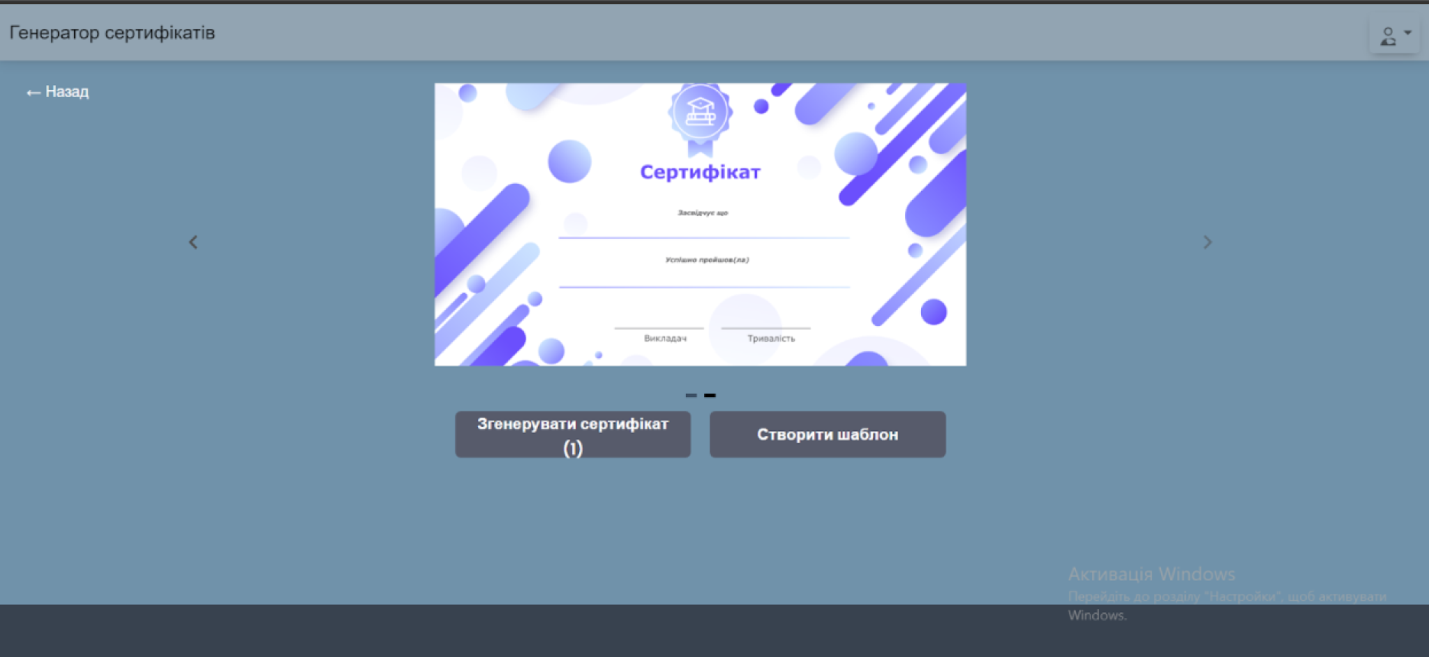


Рисунок 4.8 – Генерація сертифікату (Частина 1)

Коли застосунок відобразить попередній перегляд сертифікату, його можна буде роздрукувати або завантажити в PDF-форматі (рисунок 4.9). Також можна повернутися до генерації сертифікатів та створити новий шаблон.

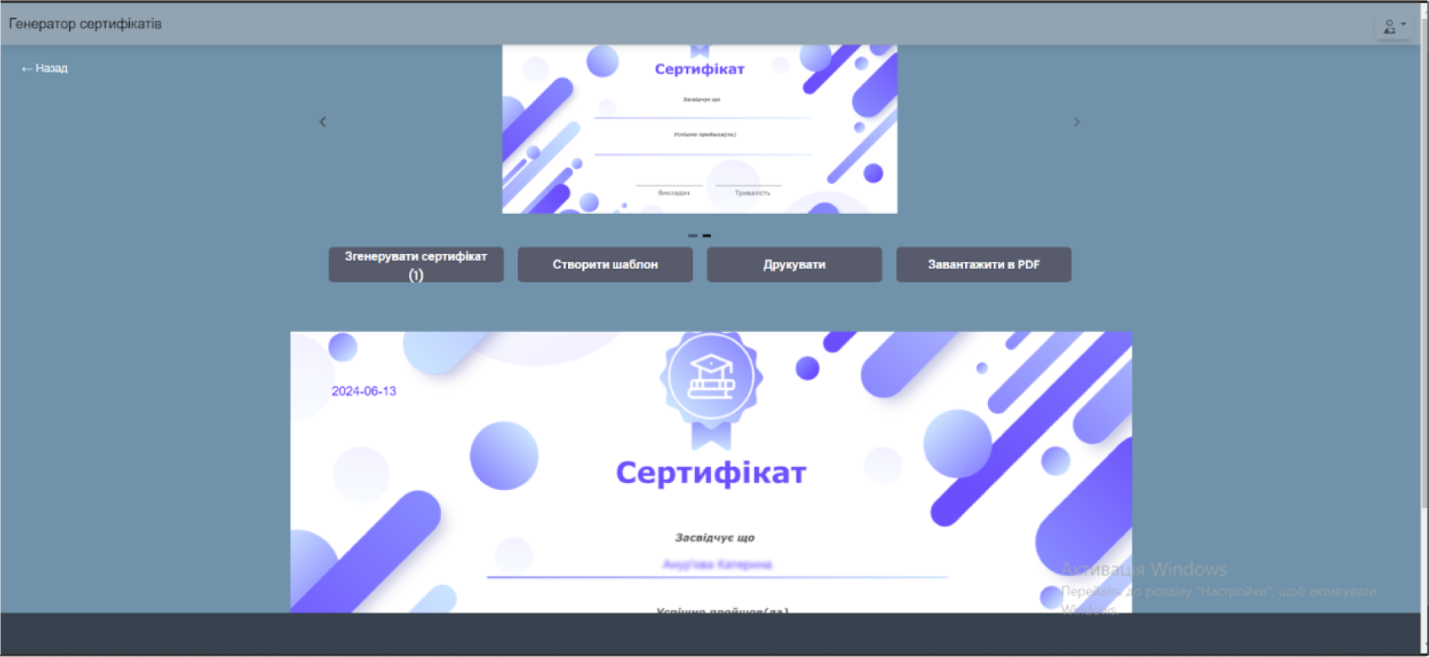


Рисунок 4.9 – Генерація сертифікату (Частина 2)

Відео з демонстрацією роботи застосунку доступне за QR-кодом (рисунок 4.10). Також його можна переглянути в рідмі-файлі репозиторію проєкту [23].

На відео показані всі операції, які можуть бути виконані в застосунку, починаючи з реєстрації облікового запису користувача, і закінчуючи переглядом та друком готового сертифікату.

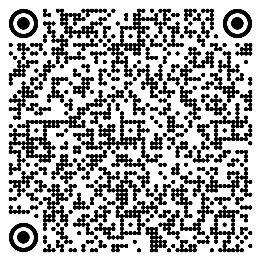


Рисунок 4.10 – QR-код для перегляду відео

## 4.5 Тестування створеного продукту та його результати

Для тестування Node.js у проєкті було використано фреймворк для тестування Jest [24].

Jest – це потужна платформа тестування JavaScript, створена Facebook, яка набула популярності завдяки своїй простоті, швидкості та функціям.

Jest призначений для тестування клієнтських і внутрішніх програм, забезпечуючи універсальність і гнучкість для розробників.

Головною перевагою Jest є простота використання. Цей фреймворк автоматично знаходить і запускає всі тестові файли у проєкті, значно спрощуючи налаштування. З Jest не треба витрачати час на складне налаштування, оскільки він автоматично налаштовується та готовий до використання відразу після встановлення. Це дозволяє швидко почати писати тести та зосередитися на логіці коду.

Jest підтримує різні типи тестування, включаючи модульне тестування, інтеграційне тестування та наскрізне тестування. Це дозволяє тестувати окремі функції, взаємодію між компонентами та загальну поведінку програми.

Однією з чудових функцій Jest є тестування знімків. Тестування миттєвих знімків дозволяє зберігати стан елементів інтерфейсу користувача як миттєві знімки та порівнювати їх із збереженими зразками під час наступних тестових прогонів. Це допомагає виявляти небажані зміни у зовнішньому вигляді програми та забезпечує стабільність інтерфейсу користувача.

Jest добре інтегрується з такими популярними фреймворками та бібліотеками, як React, Vue та Angular. Це надає можливість легко писати тести для компонентів UI, перевіряючи їх коректність і стабільність. Jest також підтримує модульні системи, такі як модулі ES6 і CommonJS, що робить його сумісним з більшістю сучасних проєктів. Крім того, Jest підтримує паралельне виконання тестів, що значно прискорює час тестування для великих проєктів.

Цей фреймворк автоматично розподіляє тестові завдання між кількома працівниками, використовуючи всі доступні ядра ЦП, що дозволяє скоротити час, витрачений на очікування результатів тесту.

Іншою важливою особливістю Jest є можливість створення покриття коду. Покриття коду дозволяє відстежувати, які частини коду були перевірені, а які ні. Це допомагає виявити не повністю охоплені частини коду та покращує загальну якість тестування. Одним словом, Jest – це потужний і зручний інструмент для автоматизації тестування програм JavaScript.

Результати тестування розробленого застосунку показані на   
рисунках 4.11-4.12.

Простота використання Jest, його гнучкість та підтримка різних типів тестування роблять його ідеальним вибором для забезпечення якості коду та прискорення процесу розробки.

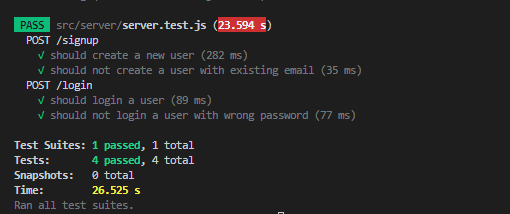


Рисунок 4.11 – Тестування коду Node.js на реєстрацію та логін користувача

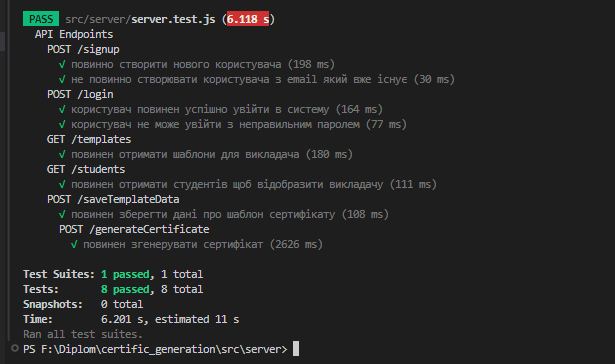


Рисунок 4.12 – Тестування всіх основних функції коду на Node.js

## 4.6 Висновки за розділом

В розділі було досліджено використання мови програмування JavaScript та її фреймворків для реалізації веб-застосунку для генерації сертифікатів на підтвердження навчальних досягнень.

Також у розділі описано використання бібліотек, таких як Express.js і Mongoose, для налаштування серверної частини та роботи з базою даних MongoDB. Використання фреймворків, таких як Node.js і Vue.js, дозволило прискорити та спростити процес розробки, забезпечивши готові рішення для поширених завдань та структури, що описують взаємодію компонентів програми. Як результат можна зробити висновок, що JavaScript та його фреймворки, такі як Node.js і Vue.js, разом з використанням додаткових бібліотек, надають широкі можливості для розробки різноманітних веб-додатків з високим рівнем інтерактивності та функціональності.

Також описані основні режими роботи створеного веб-застосунку й наведені результати його тестування.

# РОЗРАХУНОК ЕКОНОМІЧНИХ ПОКАЗНИКІВ

## 5.1 Мета розділу

Мета розділу – економічне обґрунтування розробки веб-застосунку для генерації сертифікатів на підтвердження навчальних досягнень. Для досягнення поставленої мети необхідно виконати ряд наступних завдань:

* розрахувати собівартість програмного продукту;
* визначити тривалість виконання роботи;
* знайти оптимальний організаційно-технологічний варіант виконання роботи.

## 5.2 Визначення тривалості та розрахунок собівартості виконання проєкту (перший варіант)

Для розробки веб-додатку потрібна команда таких спеціалістів: веб-дизайнер, який відповідає за створення інтерфейсу користувача, та full-stack розробник, який відповідає за розробку функціональності. Ці спеціалісти потрібні для проєктування, розробки та налаштування основних модулів і функціональності, створення інтерфейсу користувача, а також налагодження проєкту і введення його в експлуатацію.

Тривалість робочого місяця становить 22 дні, а робочого дня – 8 годин. Склад виконавців та розмір заробітної плати кожного наведено в таблиці 5.1.

Фонд основної заробітної плати обчислюється за формулою:

(5.1)

де – основна зарплата, грн.;

Т – трудовитрати робітника, людино/дні;

Зсд – середньоденна заробітна плата робітника, грн. Вона визначається за формулою 5.2.

(5.2)

де Зміс – місячна зарплата грн;

Ф – кількість робочих днів в місяць.

У full-stack розробника Зміс = 50000 грн., у веб-дизайнера проєкту  
 Зміс = 25000 грн. Ф = 22.

За наведеною формулою (5.2) середня денна заробітна плата веб-дизайнера Зсд = 1136 грн., у full-stack розробника Зсд = 2272 грн.

Таблиця 5.1 – Склад та зарплата виконавців роботи

|  |  |  |
| --- | --- | --- |
| Посада | Посадові оклади, грн. | |
| місячні | денні |
| Веб-дизайнер | 25000 | 1136 |
| Full-stack розробник | 50000 | 2272 |
| Разом | 75000 | 3408 |

Розрахунок трудомісткості робіт представлений в таблиці 5.2.

Таблиця 5.2 – Розрахунок трудомісткості робіт

|  |  |  |  |
| --- | --- | --- | --- |
| Вид роботи | Тривалість, дні | Трудомісткість, людино/дні | |
| веб-дизайнер | full-stack розробник |
| Розробка ТЗ | 5 | 5 | 5 |
| Погодження і затвердження ТЗ | 2 | 2 | 2 |
| Розробка дизайну платформи | 18 | 18 | 0 |
| Розробка клієнтської частини | 18 | 0 | 18 |
| Розробка серверної частини | 24 | 0 | 24 |
| Наповнення контентом | 3 | 3 | 0 |
| Тестування | 4 | 2 | 2 |
| Розміщення на хостингу | 2 | 0 | 2 |
| Всього | 76 | 30 | 53 |

Згідно формул (5.1) та (5.2) розрахуємо фонд основної заробітної плати:

Зос =1136 \* 30 + 2272 \* 53 = 34080 + 120416 = 154496 (грн).

Додаткова заробітна плата становить 15% (рекомендований інтервал 10-20%) від основної заробітної плати.

Здод = 0,15 \* 154496 = 23174,5 (грн).

Фонд заробітної плати складає:

Ззаг = Зос + Здод = 154496 + 23174,5 = 177670,5 (грн).

Розрахуємо єдиний соціальний внесок. Він становить 22% від загального фонду заробітної плати. Тоді:

ЄСВ = Ззаг \* 0,22 = 177670,5 \* 0,22 = 39087,6 (грн).

Необхідно розрахувати вартість матеріалів та послуг, необхідних для розробки платформи. Результати розрахунків наведені в таблиці 5.3.

Таблиця 5.3 – Витрати на матеріали та послуги

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Матеріали або послуги | Кількість, од. | Ціна за одиницю, грн. | Сума, грн. | Призначення |
| Папір формату А4 | 500 арк. | 0,25 | 125 | Друк документації |
| Ручка | 2 | 10 | 20 | Запис інформації |
| Олівець | 2 | 5 | 10 | Запис інформації |
| SSD 1 ТБ | 1 | 2900 | 2900 | Збереження копій |
| Послуги Інтернет | 4 міс. | 170 | 680 | Пошук інформації |
| Доменне ім'я | 1 рік | 299 | 299 | Розміщення веб-застосунку |
| Хостинг | 1 рік | 2000 | 20 | Розміщення веб-застосунку |
| Всього |  |  | 6034 |  |

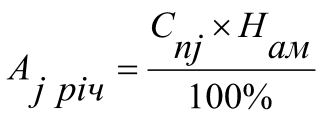
В результаті вартість матеріалів та послуг, необхідних для розробки веб-застосунку, складає 6034 грн.

Окрім матеріалів та послуг, для розробки веб-додатку також потрібне обладнання, яке представлене в таблиці 5.4.

Таблиця 5.4 – Витрати на обладнання та устаткування

|  |  |  |  |
| --- | --- | --- | --- |
| Обладнання | Кількість, од. | Ціна за одиницю, грн. | Сума, грн. |
| Ноутбук | 2 | 31000 | 62000 |
| Стіл | 2 | 2000 | 4000 |
| Стілець | 2 | 800 | 1600 |
| Комп'ютерна миша | 2 | 600 | 1200 |
| Принтер | 1 | 4000 | 4000 |
| Всього |  |  | 72800 |

Наведемо формули для розрахунку величини амортизації. Враховуючи початкову вартість обладнання і річну норму амортизації можливо розрахувати річну суму амортизаційних відрахувань за формулою:

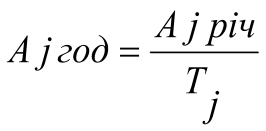
 , (5.3)

де  – річні амортизаційні відрахування за j-м видом обладнання, грн.;

 – початкова вартість обладнання цього виду, грн.;

 – річна норма амортизації, %.

Визначаємо величину амортизаційних відрахувань у розрахунку для однієї години роботи обладнання цього виду:

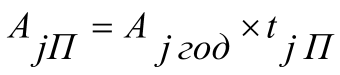
, (5.4)

де  – величина амортизаційного обладнання j-того виду за одну годину використання, грн.;

 – річний фонд роботи даного обладнання, год.

Річний фонд роботи обладнання в 2025 р. при 40-годинному робочому тижні складає  =2088 год.

Залежно від часу використання обладнання, в процесі створення продукту розраховується величина амортизаційних відрахувань, пов'язаних з виробництвом продукту:

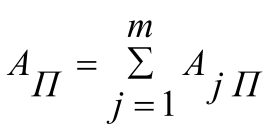
, (5.5)

де  – амортизація обладнання j-того виду в процесі розробки програмного продукту, грн.;

 – час використання обладнання j-того виду в процесі розробки програмного продукту, год.

Час використання обладнання j-того виду розраховується як добуток трудомісткості з табл. 5.2 на 8 год/день при 8-ми годинному робочому дні.

Якщо використовується різне обладнання, то величина амортизації має бути визначена шляхом підсумовування амортизаційних відрахувань для окремих видів обладнання:

 , (5.6)

де  – сума амортизації, грн.;

 – кількість видів устаткування.

Результати розрахунків наведено у таблиці 5.5.

Таблиця 5.5 – Амортизаційні відрахування обладнання

| Найменування обладнання | Час використання обладнання, , год. | Початкова вартість , грн | Річна норма амортизації, % | Амортизаційні відрахування, грн | | |
| --- | --- | --- | --- | --- | --- | --- |
| за рік | за годину використання обладнання | за період виконання роботи |
| Ноутбук | 664 | 31000 | 25 | 7750 | 3,7 | 2457 |
| Стіл | 664 | 2000 | 15 | 500 | 0,24 | 159,4 |
| Стілець | 664 | 800 | 25 | 200 | 0,1 | 66,4 |

Продовження таблиці 5.5 – Амортизаційні відрахування обладнання

| Найменування обладнання | Час використання обладнання, , год. | Початкова вартість , грн | Річна норма амортизації, % | Амортизаційні відрахування, грн | | |
| --- | --- | --- | --- | --- | --- | --- |
| за рік | за годину використання обладнання | за період виконання роботи |
| Комп'ютерна миша | 664 | 600 | 20 | 120 | 0,06 | 39,9 |
| Принтер | 6 | 4000 | 30 | 1200 | 0,58 | 3,48 |
| Всього |  |  |  |  |  | 2726,2 |

Згідно з даними таблиці 5.5, амортизація за період виконання проєкту cтановить 2726,2 грн.

Також потрібно розрахувати витрати на електроенергію (таблиця 5.6).

Таблиця 5.6 – Розрахунок витрат на електроенергію

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Найменування | Потужність, кВт/год. | Час використання, год. | Вартість кВт/год., грн | Витрати, грн |
| Ноутбук | 0,065 | 664 | 4,32 | 186,5 |
| Принтер | 0,03 | 6 | 4,32 | 0,77 |
| Освітлення | 0,035 | 664 | 4,32 | 104 |
| Всього |  |  |  | 287,6 |

Після цього необхідно розрахувати собівартість розробленого продукту для першого варіанту розробки (таблиця 5.7).

Таблиця 5.7 – Собівартість розробки веб-застосунку для генерації сертифікатів на підтвердження навчальних досягнень (перший варіант)

| № | Статті | Сума, грн. | Примітка |
| --- | --- | --- | --- |
| 1 | Основна заробітна плата (ОЗП) | 154496 | Формули 5.1 і 5.2 |
| 2 | Додаткова заробітна плата (ДЗП) | 23174,5 | 15% від ОЗП |
| 3 | Єдиний соціальний внесок | 39087,6 | 22% (ОЗП +ДЗП) |
| 4 | Матеріали і послуги | 6034 | З табл. 5.3 |

Продовження таблиці 5.7 – Собівартість розробки веб-застосунку для генерації сертифікатів на підтвердження навчальних досягнень (перший варіант)

| № | Статті | Сума, грн. | Примітка |
| --- | --- | --- | --- |
| 5 | Амортизація | 2726,2 | З табл. 5.5 |
| 6 | Витрати на електроенергію | 287,6 | З табл. 5.6 |
| 7 | Собівартість (С) | 225806 | п.1+п.2+п.3+…+п.6 |

## 5.3 Визначення тривалості та розрахунок собівартості виконання проєкту (другий варіант)

Для розрахунків другого варіанту використаємо команду у складі: веб-дизайнер, front-end розробник та baсk-end розробник. У веб-дизайнера проєкту Зміс = 25000 грн., у front-end розробника Зміс = 30000 грн. та у back-end розробника Зміс = 35000 грн. Ф = 22.

За формулою (5.2) середня денна заробітна плата у веб-дизайнера  
 Зсд = 1136 грн., у front-end розробника Зсд = 1364 грн. та у back-end розробника Зсд = 1591 грн.

Склад виконавців та розмір заробітної плати кожного наведено в таблиці 5.8.

Таблиця 5.8 – Склад та зарплата виконавців роботи

|  |  |  |
| --- | --- | --- |
| Посада | Посадові оклади, грн. | |
| місячні | денні |
| Веб-дизайнер | 25000 | 1136 |
| Front-end розробник | 30000 | 1364 |
| Back-end розробник | 35000 | 1591 |
| Разом | 82000 | 4091 |

Розрахунок трудомісткості робіт представлений в таблиці 5.9.

Таблиця 5.9 – Розрахунок трудомісткості робіт

| Вид роботи | Тривалість, дні | Трудомісткість, людина/дні | | |
| --- | --- | --- | --- | --- |
| веб-дизайнер | front-end розробник | back-end розробник |
| Розробка ТЗ | 3 | 3 | 3 | 3 |

Продовження таблиці 5.9 – Розрахунок трудомісткості робіт

| Вид роботи | Тривалість, дні | Трудомісткість, людина/дні | | |
| --- | --- | --- | --- | --- |
| веб-дизайнер | front-end розробник | back-end розробник |
| Погодження і затвердження ТЗ | 2 | 2 | 0 | 0 |
| Розробка дизайну | 18 | 18 | 0 | 0 |
| Розробка клієнтської частини | 14 | 0 | 14 | 0 |
| Розробка серверної частини | 16 | 0 | 0 | 16 |
| Наповнення контентом | 2 | 0 | 2 | 0 |
| Тестування | 4 | 4 | 4 | 4 |
| Розміщення на хостинг | 2 | 0 | 0 | 2 |
| Всього | 61 | 27 | 23 | 25 |

Згідно (5.1), (5.2) розрахуємо фонд основної заробітної плати:

Зос = 1136 \* 27 + 1364 \* 23 + 1591 \* 25 = 30672 + 31372 + 39775 = 101819 (грн).

Додаткова заробітна плата становить 15% від основної заробітної плати.

Здод = 0,15 \* 101819 = 15273 (грн).

Фонд заробітної плати складає:

Зобщ = Зос + Здод = 101819 + 15273 = 117092 (грн).

Єдиний соціальний внесок становить визначену законодавством суму, яку роботодавці та самозайняті особи сплачують для фінансування державного соціального страхування. Він використовується для забезпечення пенсійних виплат, лікарняних, страхових компенсацій у разі втрати працездатності, а також інших соціальних гарантій. Розрахуємо його:

ЄСВ = Ззаг \* 0,22 = 117092 \* 0,22 = 25760 (грн).

Розрахуємо вартість матеріалів та послуг. Результати розрахунків представлені в таблиці 5.10.

Таблиця 5.10 – Витрати на матеріали та послуги

| Матеріали або послуги | Кількість, од. | Ціна за одиницю, грн. | Сума, грн. | Призначення |
| --- | --- | --- | --- | --- |
| Папір формату А4 | 500 арк. | 0,19 | 90 | Друк документації |
| Ручка | 3 | 8 | 24 | Запис інформації |
| Олівець | 3 | 5 | 15 | Запис інформації |
| SSD 1 ТБ | 1 | 2200 | 2200 | Збереження копій |
| Послуги Інтернет | 4 міс. | 170 | 680 | Пошук інформації |
| Доменне ім'я | 1 рік | 299 | 299 | Розміщення веб-застосунку |
| Хостинг | 1 рік | 1630 | 1630 | Розміщення веб-застосунку |
| Всього |  |  | 4938 |  |

В результаті вартість матеріалів та послуг, необхідних для розробки веб-застосунку, складає 4938 грн.

Також для розробки веб-застосунку потрібне обладнання. Кількість та вартість одиниці кожного виду, а також результуюча сума і підсумкова вартість обладнання представлені в таблиці 5.11.

Таблиця 5.11 – Витрати на обладнання та устаткування

|  |  |  |  |
| --- | --- | --- | --- |
| Обладнання | Кількість, од. | Ціна за одиницю, грн | Сума, грн |
| Ноутбук | 3 | 25000 | 75000 |
| Стіл | 3 | 1800 | 5400 |
| Стілець | 3 | 800 | 2400 |
| Комп'ютерна миша | 3 | 400 | 1200 |
| Принтер | 1 | 3300 | 3300 |
| Всього |  |  | 87300 |

Використовуючи формули 5.3 – 5.6 та дані стосовно витрат на обладнання та устаткування, представлені у таблиці 5.11, виконаємо розрахунок амортизаційних відрахувань для всіх видів використовуваного обладнання. Розрахунки здійснюватимуться на основі початкової вартості обладнання, часу його експлуатації та річної норми амортизації, що дозволить отримати точні та обґрунтовані фінансові показники.

Отримані результати занесемо до таблиці 5.12, де буде представлено деталізований розподіл амортизаційних відрахувань за кожним видом обладнання. Окрім того, проведемо аналіз отриманих даних, узагальнимо результати та розрахуємо загальну суму амортизаційних витрат за весь період реалізації проєкту. Це дасть змогу оцінити економічну ефективність використання обладнання, його довгострокову рентабельність, а також потенційний вплив амортизаційних витрат на загальну собівартість розробки.

Таблиця 5.12 – Амортизаційні відрахування обладнання

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Найменування обладнання | Час використання обладнання, , год. | Початкова вартість , грн | Річна норма амортизації, % | Амортизаційні відрахування, грн | | |
| за рік | за годину використання обладнання | за період виконання роботи |
| Ноутбук | 600 | 25000 | 25 | 6250 | 2,98 | 1788 |
| Стіл | 600 | 1800 | 15 | 270 | 0,13 | 78 |
| Стілець | 600 | 800 | 25 | 200 | 0,1 | 60 |
| Комп'ютерна миша | 600 | 400 | 20 | 80 | 0,04 | 24 |
| Принтер | 6 | 3300 | 30 | 990 | 0,47 | 2,8 |
| Всього |  |  |  |  |  | 1952,8 |

Згідно з даними, які були розраховані та внесені до таблиці 5.12, амортизація за період виконання проєкту складає 1952,8 грн.

На наступному етапі необхідно виконати детальні розрахунки витрат на електроенергію, що використовувалася під час розробки другого варіанту продукту, а також визначити його собівартість з урахуванням споживаної потужності обладнання, тривалості роботи системи та тарифу на електроенергію.

Результати розрахунків, представлені в таблицях 5.13 та 5.14, дозволяють детально оцінити економічну ефективність другого варіанту розробки та порівняти її з першим варіантом, коли над проектом працювала команда, що складалася з двох розробників: веб-дизайнера та full-stack розробника.

Розрахунки дозволяють простежити вплив структури команди на загальні витрати, ефективність розподілу завдань між розробниками та потенційні економічні вигоди від різних підходів до реалізації проєкту , враховуючи витрати на персонал та ресурсне забезпечення.

Таблиця 5.13 – Розрахунок витрат на електроенергію

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Найменування | Потужність, кВт/год. | Час використання, год. | Вартість кВт/год., грн | Витрати, грн |
| Ноутбук | 0,065 | 600 | 4,32 | 168 |
| Принтер | 0,03 | 6 | 4,32 | 0,77 |
| Освітлення | 0,035 | 600 | 4,32 | 90,7 |
| Всього |  |  |  | 259,5 |

Таблиця 5.14 – Собівартість розробки веб-застосунку для генерації сертифікатів на підтвердження навчальних досягнень (другий варіант)

| № | Статті | Сума, грн. | Примітка |
| --- | --- | --- | --- |
| 1 | Основна заробітна плата (ОЗП) | 101819 | Формули 5.1 і 5.2 |
| 2 | Додаткова заробітна плата (ДЗП) | 15273 | 15% від ОЗП |
| 3 | Єдиний соціальний внесок | 25760 | 22% (ОЗП +ДЗП) |
| 4 | Матеріали і послуги | 4938 | З табл. 5.10 |
| 5 | Амортизація | 1952,8 | З табл. 5.12 |
| 6 | Витрати на електроенергію | 259,5 | З табл. 5.13 |
| 7 | Собівартість (С) | 150002,3 | п.1+п.2+п.3+…+п.6 |

## 5.4 Висновки за розділом

Було наведено економічне обґрунтування розробки веб-застосунку. Для першого варіанту було обрано команду з двох розробників: веб-дизайнера та full-stack розробника. Для другого варіанту обрано три розробника: веб-дизайнер, front-end розробник, back-end розробник. Для оцінки вартості проєкту враховано витрати різних видів. Собівартість розробки платформи у першому варіанті становить 225806 грн, у другому – 150002,3 грн, тривалість, відповідно, 76 та 61 днів.

На основі отриманих результатів можна зробити висновок, що для розробки краще залучати вузькоспеціалізованих працівників, тому що:

* швидкість розробки значно вища. Тривалість розробки для другого варіанту менша на 15 днів;
* внаслідок продуктивної роботи таких спеціалістів загальна вартість розробки платформи значно менша. Різниця становить 75803,7 грн.

Таким чином, можна прийти до висновку, що розробкою проєкту мають займатися розробники з другого варіанту.

# ВИСНОВКИ

В роботі вирішено завдання розробки веб-застосунку для генерації сертифікатів на підтвердження навчальних досягнень, який має спростити та автоматизувати процес створення персоналізованих сертифікатів для учасників подій, семінарів, курсів та конференцій. Слід відзначити, що завдання виконане у повному обсязі, мета роботи досягнута.

Процес розробки включав в себе аналіз проблем керування сертифікатами, порівняльний аналіз програмного забезпечення аналогічного призначення, визначення вимог, побудову UML-діаграм, розробку програмного продукту, тестування, підготовку до розгортання на сервері, а також опис проєктних рішень та використаних технологій.

Веб-застосунок побудовано з використанням Node.js та Vue.js. Він складається з ряду підсистем, включаючи інтерфейс користувача, серверну частину, базу даних. Вихідні коди продукту розміщені в публічному GitHub-репозиторії, фрагмент коду також представлено у Додатку А.

Важливою частиною кваліфікаційної роботи є також розрахунок економічних показників продукту.

Отже, можна зазначити, що розроблений веб-застосунок відповідає поставленим вимогам, забезпечуючи автоматизацію процесу створення сертифікатів. Його застосування полегшує роботу організаторів подій та забезпечує доступ до сертифікатів для учасників освітніх подій.

Загалом, розроблений веб-застосунок є наочним прикладом використання веб-технологій в освітній галузі. Як вже було зазначено вище, вихідні коди продукту є публічно доступними, тож будь-який освітній заклад може скористатись ними аби створити власне рішення для генерації сертифікатів, яке відповідає їх специфічним вимогам та потребам.

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Довідка прав в освіті. Документ про підвищення кваліфікації // Міністерство освіти і науки України. 2025. URL: https://info.eo.gov.ua/dokument-pro-pidvyshhennya-kvalifikacziyi (дата звернення: 19.02.2025).
2. Семінар як метод підвищення кваліфікації // UBI Конференц Холл. 2017. URL: https://ubi-hall.com.ua/uk/2017/09/21/seminar-iak-metod-pidviennia  
   kvalifik/ (дата звернення: 19.02.2025).
3. Що таке онлайн-курс? // Самоосвіта. 2024. URL: https://samoosvita.in.ua/scho-take-onlayn-kurs/ (дата звернення: 19.02.2025).
4. Тренінг як одна з форм підвищення кваліфікації бібліотечного персоналу: посібник. Львів: Регіональний тренінговий центр Львівської обласної універсальної наукової бібліотеки, 2010. 24 с.
5. Free Certificate Maker for Custom Certificates: [Веб-сайт]. URL: https://www.visme.co/certificate-maker/ (дата звернення: 24.05.2025).
6. Free Online Certificate Maker: [Веб-сайт]. URL: https://www.canva.com/  
   create/certificates/ (дата звернення: 24.05.2025).
7. Online Certificate Maker: [Веб-сайт]. URL: https://www.bannerbear.com/  
   generators/free-online-certificate-generator/ (дата звернення: 24.05.2025).
8. Функціональність – це: [Веб-сайт]. 2016. URL: https://ua.waykun.com/articles/funkcionalnist-ce.php (дата звернення: 10.03.2025).
9. Моделювання бізнес-процесів та архітектура програмного забезпечення: опорний конспект лекцій. Тернопіль, 2015. 105 с.
10. Що таке дизайн графічного інтерфейсу користувача gui? // QATestLab. URL: https://training.qatestlab.com/blog/technical-articles/what-is-graphical-user-interface-design/ (дата звернення: 20.04.2025).
11. Класи в програмуванні: занурення в об’єктно-орієнтоване програмування // Foxminded. URL: https://foxminded.ua/klasy-v-prohramuvanni/ (дата звернення: 03.05.2025).
12. MongoDB Підручник [Електронний ресурс], Режим доступу: https://w3schoolsua.github.io/mongodb/index.html#gsc.tab=0 Назва з екрану (дата звернення 02.05.2025)
13. Що таке MongoDB? вступ, Архітектура, функції та приклад // Guru99. URL: https://www.guru99.com/uk/what-is-mongodb.html (дата звернення: 02.05.2025).
14. Render: [Веб-сайт]. URL: https://dashboard.render.com/ (дата звернення: 18.05.2025).
15. Що таке JavaScript і для чого він потрібен // GOIT. 2024. URL: https://goit.global/ua/articles/shcho-take-javascript-i-dlia-choho-vin-potriben/ (дата звернення: 24.05.2025).
16. Що таке фреймворк: пояснюємо простими словами // Вrainlab. 2023. URL: https://brainlab.com.ua/uk/blog-uk/shho-take-frejmvork-poyasnyuyemo-prost  
    ymy-slovamy (дата звернення: 21.05.2025).
17. Що таке Node JS простими словами // Dan-it. URL: https://dan-it.com.ua/uk/blog/chto-jeto-takoe-node-js-prostymi-slovami/ (дата звернення: 25.05.2025).
18. Що таке Vue? // Vue-Mastery. URL: https://ua.vuejs.org/  
    guide/introduction.html#what-is-vue (дата звернення: 25.05.2025).
19. Розробка вебсайтів TECHNOLOGIES EXPRESS.JS // Brander. URL: https://brander.ua/technologies/expressjs (дата звернення: 23.05.2025).
20. Вступ до Mongoose для MongoDB та Node.js // envato-tuts+. URL: https://code.tutsplus.com/uk/an-introduction-to-mongoose-for-mongodb-and-nodejs--cms-29527a (дата звернення: 25.05.2025).
21. Vue Component Framework // Vuetify. URL: https://vuetifyjs.com/en/ (дата звернення: 20.05.2025).
22. Code Editing. Redefined: [Веб-сайт]. URL: https://code.visualstudio.com/ (дата звернення: 21.05.2025).
23. Розробка веб-застосунку для генерації сертифікатів на підтвердження навчальних досягнень: [Веб-сайт]. URL: https://github.com/ppc-ntu-khpi/Certificates\_KhAI (дата звернення: 25.05.2025).
24. Jest: [Веб-сайт]. URL: https://jestjs.io/uk/ (дата звернення: 10.05.2025).
25. Малєєва, Ю.А. Методичні вказівки з дипломного проектування бакалаврів спеціальності 122 «Комп’ютерні науки» [Текст]: навч. посіб. до виконання розрахункових робіт / Ю. А. Малєєва, Т. М. Соляник, А. В. Попов. – Харків: Нац. аерокосм. ун-т ім. М. Є. Жуковського «Харків. авіац. ін-т», 2019. – 72 с.

# ДОДАТОК А. ВИХІДНІ КОДИ (ФРАГМЕНТ)

app.post(

"/generateCertificate",

[checkToken, checkRole("teacher")],

async (req, res) => {

**const** { CertData } = req.body;

**const** selectedStudentIds = req.body.selectedStudents;

**try** {

**const** selectedStudents = await User.find({

\_id: { $in: selectedStudentIds },

});

**const** templateId = req.body.selectedTemplateId;

**const** template = await Template.findById(templateId);

**if** (template.image) {

**const** dataURI = template.image;

**let** studentsCnt = selectedStudents.**length**;

**const** sertificates = [];

**const** targetWidth = 1200;

**const** scaleFactor = template.imageWidth / targetWidth;

selectedStudents.**forEach**((student) => {

**const** certNumber = `CERT-${Date.now()}-${Math.**floor**(

Math.**random**() \* 1000

)}`;

**const** adjustedTemplate = {

...template,

title\_top: template.title\_top / scaleFactor + 15,

title\_left: template.title\_left / scaleFactor,

duration\_top: template.duration\_top / scaleFactor + 15,

duration\_left: template.duration\_left / scaleFactor,

teacherSurname\_top: template.teacherSurname\_top / scaleFactor + 15,

teacherSurname\_left: template.teacherSurname\_left / scaleFactor,

studentName\_top: template.studentName\_top / scaleFactor + 15,

studentName\_left: template.studentName\_left / scaleFactor,

dateOfGiving\_top: template.dateOfGiving\_top / scaleFactor + 10,

dateOfGiving\_left: template.dateOfGiving\_left / scaleFactor,

title\_is\_centred: template.title\_is\_centred,

duration\_is\_centred: template.duration\_is\_centred,

teacherName\_is\_centred: template.teacherName\_is\_centred,

studentName\_is\_centred: template.studentName\_is\_centred,

date\_is\_centered: template.date\_is\_centered,

};

**const** center = "transform: translateX(-50%);";

**const** html = `

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Certificate Template</title>

<style>

body {

font-family: Arial, sans-serif;

width: ${targetWidth}px;

height: auto;

position: relative;

}

.certificate {

width: ${targetWidth}px;

height: auto;

position: relative;

}

.certificate .title {

position: absolute;

font-size: 20px;

color: ${adjustedTemplate.title\_color};

top: ${adjustedTemplate.title\_top}px;

left: ${adjustedTemplate.title\_left}px;;

${adjustedTemplate.title\_is\_centred ? center : ""}

z-index: 999;

}

.certificate .duration {

position: absolute;

font-size: 18px;

color: ${adjustedTemplate.duration\_color};

top: ${adjustedTemplate.duration\_top}px;

left: ${adjustedTemplate.duration\_left}px;

${adjustedTemplate.duration\_is\_centred ? center : ""}

z-index:999

}

.certificate .teacher {

position: absolute;

font-size: 18px;

color: ${adjustedTemplate.teacherSurname\_color};

top: ${adjustedTemplate.teacherSurname\_top}px;

left: ${adjustedTemplate.teacherSurname\_left}px;

${adjustedTemplate.teacherName\_is\_centred ? center : ""}

z-index:999

}

.certificate .student {

position: absolute;

font-size: 18px;

color: ${adjustedTemplate.studentName\_color};

top: ${adjustedTemplate.studentName\_top}px;

left: ${adjustedTemplate.studentName\_left}px;

${adjustedTemplate.studentName\_is\_centred ? center : ""}

z-index: 999;

}

.certificate .givingDate {

position: absolute;

font-size: 18px;

color: ${adjustedTemplate.dateOfGiving\_color};

top: ${adjustedTemplate.dateOfGiving\_top}px;

left: ${adjustedTemplate.dateOfGiving\_left}px;

${adjustedTemplate.date\_is\_centered ? center : ""}

z-index: 999;

}

.cert-picture{

position: relative;

width: ${targetWidth}px;

height: auto;

}

</style>

</head>

<body>

<div class="certificate">

<div class="title">${CertData.title}</div>

<div class="duration">${CertData.duration}</div>

<div class="teacher">${CertData.teacherSurname}</div>

<div class="student">${student.name}</div>

<div class="givingDate">${CertData.dateOfGiving}</div>

<img class="cert-picture" src="{{imageSource}}" alt="Certificate Template">

</div>

</body>

</html>

`;

nodeHtmlToImage({

html,

content: { imageSource: dataURI },

encoding: "buffer",

}).then(async (imageBuffer) => {

**const** base64Image = imageBuffer.**toString**("base64");

**const** newCertificate = **new** Certificate({

studentId: student.id,

templateId: template.id,

image: base64Image,

certNumber: certNumber,

title: CertData.title,

duration: CertData.duration,

teacherSurname: CertData.teacherSurname,

dateOfGiving: CertData.dateOfGiving,

});

await newCertificate.save();

sertificates.push(base64Image);

studentsCnt--;

**if** (studentsCnt <= 0) {

res.status(200).json(sertificates);

}

});

});

}

} **catch** (error) {

console.error("Error generating certificate:", error);

}

}

);

Решта коду знаходиться у публічному репозитарії, доступному за цим посиланням: https://github.com/ppc-ntu-khpi/Certificates\_KhAI.

|  |  |
| --- | --- |
| Також ви можете скористатись наведеним QR-кодом: |  |

# 

# ДОДАТОК Б. ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ

