Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
бакалавр
(ступінь вищої освіти)

на тему РОЗРОБЛЕННЯ ТА ДОСЛІДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ УПРАВЛІННЯ СТУДЕНТСЬКИМИ СТАЖУВАННЯМИ
DEVELOPMENT AND RESEARCH OF SOFTWARE FOR MANAGING STUDENT INTERNSHIPS

Виконав(ла): студент(ка) 4 курсу, групи КНТ-112
Спеціальності 121 Інженерія програмного
(код і найменування спеціальності)
забезпечення

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

ЧОРНА Ж.І.
(ПРІЗВИЩЕ та ініціали
Керівник ГОФМАН Є.О.
(ПРІЗВИЩЕ та ініціали)
Рецензент ДЬЯЧУК Т.С.
(ПРІЗВИЩЕ та ініціали)

2026

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н., проф.
_____Сергій СУББОТІН
“____” _____2026 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

ЧОРНОЇ Жанни Ігорівни
(ПРІЗВИЩЕ, ім’я, по батькові)

1. Тема проєкту (роботи) Розроблення та дослідження програмного забезпечення для управління студентськими стажуваннями. Development and Research of Software for Managing Student Internships
керівник проєкту (роботи) к.т.н., доцент, ГОФМАН Євгеній Олександрович,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім’я, по батькові)
затверджені наказом закладу вищої освіти від “ 07 ” квітня 2026 року
№ 139
2. Строк подання студентом проєкту (роботи) 10 червня 2026 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково–пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Розробка архітектури програми. 3. Розробка програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Прийняв виконане завдання
1-4 Основна частина	ГОФМАН Є.О., доцент		
Нормоконтроль	КАЛІНІНА М.В., асистент		

7. Дата видачі завдання “ 03” квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій	2 тиждень	Розділ 2
	розробки.		
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження	5 тиждень	Розділ 4
	програмного забезпечення.		
7	Оформлення пояснювальної записки та документів	6 тиждень	Додатки
	до неї.		
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка) _____ Жанна ЧОРНА
(підпис) (Ім’я ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ Євгеній ГОФМАН
(підпис) (Ім’я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до випускної кваліфікаційної роботи бакалавра: 135 с., 11 табл., 49 рис., 3 дод., 15 джерел.

НАВЧАННЯ, ОНЛАЙН ОСВІТА, ЗАКЛАДИ ВИЩОЇ ОСВІТИ, СТАЖУВАННЯ, ВЕБЗАСТОСУНОК, УПРАВЛІННЯ СТУДЕНТСЬКИМИ СТАЖУВАННЯМИ, ВЕБТЕХНОЛОГІЇ.

Об’єкт дослідження – процес інженерії програмного забезпечення для управління студентськими стажуваннями.

Предмет дослідження – методи та інструменти для розробки вебзастосунку на базі архітектури клієнт-сервер.

Мета роботи – підвищення ефективності та рівня автоматизації процесу управління студентськими стажуваннями.

Матеріали, методи та технічні засоби: використано методи об’єктно орієнтованого програмування та сучасні технології веброзробки. Для інтерфейсу – JavaScript фреймворк React, для серверної частини – PHP та фреймворк Laravel, для зберігання та управління даними – система керування базами даних PostgreSQL, а для взаємодії між клієнтською та серверною частинами використано REST API. Для розробки застосунка використовувалося середовище Visual Studio Code на персональному комп’ютері з операційною системою Windows 11.

Результати. Створено сучасний вебзастосунок для управління студентськими стажуваннями, який має інтуїтивний інтерфейс, та забезпечує автоматизацію основних процесів, включно із системою ролей користувачів, зберігання та сортування великої кількості зашифрованої інформації.

Висновки. Розроблено вебзастосунок з використанням сучасних технологій, який автоматизує основні процеси студентського стажування, забезпечує захищене зберігання даних та їх коректну обробку.

Галузь використання – частина освітнього процесу в закладах вищої освіти.

Економічна ефективність. Даний вебзастосунок забезпечує зменшення ризику помилок в офіційній документації університетів, скорочує час на створення звітів завдяки автоматизованим процесам і сприяє підвищенню ефективності організації стажування студентів та оптимізації роботи навчального закладу.

ABSTRACT

Explanatory note to the bachelor's thesis: 135 pages, 11 tables, 49 figures, 3 appendixes, 15 sources.

EDUCATION, ONLINE EDUCATION, HIGHER EDUCATION INSTITUTIONS, INTERNSHIPS, WEB APPLICATIONS, STUDENT INTERNSHIP MANAGEMENT, WEB TECHNOLOGIES.

Research object – software engineering process for managing student internships.

Research subject – use of various tools for the development of a web application.

The purpose of the work – is to improve the efficiency and improve the quality of automation of the process of managing student internships.

Research object – the process of automation and quality assurance of student internship management.

Research subject – methods and tools for developing a web application based on client-server architecture.

The purpose of the work – increasing the efficiency and level of automation of the student internship management process.

Materials, methods and technical means: methods of object-oriented programming and modern web development technologies were used. The React framework is used for the – JavaScript interface, the PHP – server part and the Laravel framework, the PostgreSQL database management system is used for data storage and management, and the REST API is used for interaction between the client and server parts. The Visual Studio Code environment on a personal computer running the Windows 11 operating system was used to develop the application.

Results. A modern web application for managing student internships has been created, which has an intuitive interface, provides automation of basic

processes, including a system of user roles, storage and sorting of a large amount of encrypted information.

Conclusions. A web application using modern technologies has been developed, which automates the main processes of student internships, provides secure data storage and their correct processing.

Field of use – part of the educational process in institutions of higher education.

Economic efficiency. This web application provides a reduction in the risk of errors in the official documentation of universities, reduces the time for creating reports thanks to automated processes and contributes to increasing the efficiency of organizing student internships and optimizing the work of the educational institution.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API – Application Programming Interface;

DOM – Document Object Model;

LMS – Learning Management System;

ORM – Object-Relational Mapping;

PDF – Portable Document Format;

PHP – Hypertext Preprocessor;

UI – User Interface;

БД – база даних;

ВНЗ – вищий навчальний заклад.

ВСТУП

Цифровізація освіти, на сьогоднішній день, є дуже актуальним напрямом, бо вона дозволяє покращити якість навчання, автоматизувати освітні процеси та забезпечити прозорий контроль і моніторинг результатів [1]. Однією з ключових ланок освітнього процесу в закладах вищої освіти є стажування студентів, завдяки якому учні мають змогу здобувати практичні навички для закріплення теоретичних знань. Цей процес складається з формування завдань, закріплення керівників, подальшого контролю виконання завдань та створення відповідної документації виконаної роботи та її оцінки [2].

В багатьох закладах вищої освіти регулювання стажуваннями виконується вручну, що призводить до необхідності зберігати велику кількість паперових документів, а також, в багатьох випадках, до втрати даних. Але найголовнішим від’ємним фактором є те, що сортування і зберігання цих документів займає багато часу і знижує ефективність. Незважаючи на актуальність проблеми у вільному доступі не має програмного забезпечення, яке могло б допомогти з її вирішенням. Представлені або великі корпоративні системи (LMS) які не підтримують процеси практики, або комерційні платформи для стажувань, які адаптовані на пошук стажувань і не підтримують сам контроль практики, або локальні проекти на сайтах типу GitHab які були створені відповідно суб’єктивному баченню виконавця без врахування реальних потреб вищих закладів навчання.

Тож проблема дослідження полягає в необхідності створення вебзастосунку задля коректного і відкритого відстеження всіх етапів стажування із врахуванням потреб вищих навчальних закладів. А також необхідністю ефективного та доцільного використання цифрових рішень, а саме технологій які підвищуватимуть якість та ефективність освітніх процесів [3].

Об’єктом дослідження є процес автоматизації та забезпечення якості управління студентськими стажуваннями.

Предметом дослідження є вебзастосунок для автоматизації процесів та відстеження всіх етапів стажування з використанням сучасних технологій.

Метою роботи є зменшення часових витрат студентів і викладачів на оформлення документації та контроль проходження стажувань шляхом створення вебзастосунку з автоматичним формуванням PDF-звітів і системою Telegram-нагадувань про дедлайни.

Для досягнення поставленої мети у роботі вирішувалися такі завдання:

– проаналізувати сучасні методи управління практикою та визначити їх обмеження;

– розробити архітектуру вебзастосунку: серверну частину, клієнтську частину та модулі автоматизації;

– реалізувати модуль Telegram-сповіщень та автоматичне формування PDF-документів;

– провести тестування працездатності застосунку, його коректність, надійність та захищеність.

Методика дослідження включає аналіз літератури, проєктування архітектури застосунку за допомогою розподіленої логіки, використання об’єктно-орієнтованого програмування для серверної частини з використання PHP й Laravel, та розроблення інтерфейсу на React. Для створення звітів використовується бібліотека DOMPDF, а для реалізації автоматичних сповіщень – Telegram Bot API.

Створений вебзастосунок може бути впроваджений вищими навчальними закладами в освітній процес задля автоматизації процесів і контролю студентськими стажуваннями. Його цінність в скороченні часу для учасників освітнього процесу для оформлення документів, підвищення прозорості етапів практики та зменшенню кількості пропущених дедлайнів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Короткий опис досліджуваної предметної області

Через спалах пандемії у 2019 році та подальших подій, онлайн навчання стало невід'ємною частиною освітнього процесу в світі, тож онлайн платформи для навчання здобули широку популярність. На рисунку 1.1 приведено статистику однієї з таких платформ Moodle [4].

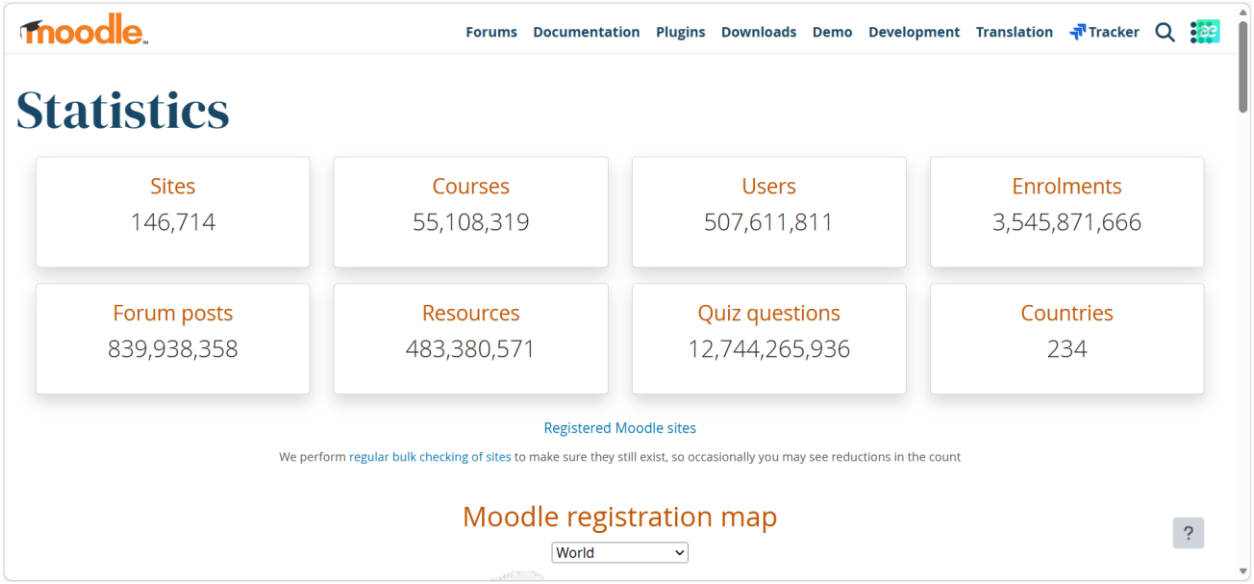


Рисунок 1.1 – Статистика кількості користувачів Moodle [4]

Проте, більшість таких платформ зосереджені на дистанційному навчанні та управлінні курсами, у той час, як супровід студентських стажувань залишається поза увагою більшості платформ. Студентські стажування є важливим кроком при здобутті вищої освіти, але, на жаль, саме вони ще виконуються вручну, без використання автоматизованих процесів.

Основні поняття, пов'язані з управлінням студентських стажувань, включають:

– виробнича практика – форма навчання, що передбачає отримання практичних навичок на підприємстві;

– місце практики – організація або компанія, де студент проходить стажування;

– керівник практики – особа, відповідальна за супровід і оцінювання студента;

– оцінювання стажування – процес перевірки звітів, виконаних завдань і загального прогресу;

– документація практики – офіційні документи (щоденник, пояснювальна записка).

Через відсутність цифрових інструментів та побоювання зробити помилки в офіційних документах, управління студентськими стажуваннями забирає багато часу в усіх його учасників. Для студентів це проявляється у незручності виконання, бо зазвичай студент заповнює усе вручну, не знаючи специфіку оформлення документів щодо практики та термінів виконання, і має звертатися до керівника в особистих повідомленнях в месенджерах, що затягує отримання відповідей і ускладнює процес виконання завдань. А для викладача незручність в тому, що він є керівником багатьох студентів, і має витрачати багато часу на роз’яснення всіх незрозумілих студенту моментів.

1.2 Огляд існуючих рішень проблеми

Сьогодні існує дуже багато програм в освітній сфері призначених для покращення ефективності та контролю навчання. Але більшість з них не включають в себе саме ефективний контроль стажуваннями студентів. В цьому розділі буде розглянуто найпоширеніші рішення та їх можливості.

1.2.1 Застосунок Moodle

Moodle – це LMS для створення персоналізованих навчальних місць з багатьма ролями. Він дозволяє керувати навчальним процесом: створювати курси та завдання, завантажувати навчальні документи, переглядати та оцінювати роботи та відстежувати успішність студентів [4].

Плюси:

– безкоштовне програмне забезпечення з відкритим кодом;

– можливість встановлення на власний сервер забезпечує надійність та безпеку даних, оскільки всі дані не будуть належати іншим сторонам;

– гнучкість, кастомізація та можливість інтеграції плагінів.

Мінуси:

– заскладний та перевантажений інтерфейс;

– адміністратор повинен мати технічні знання для встановлення, налаштування та роботи з програмою;

– немає розділу контроля стажуваннями студентів;

– хоч сама програма безкоштовна, необхідно платити за хостинг та домен.

Вигляд домашній сторінки застосунку Moodle продемонстровано на рисунку 1.2.

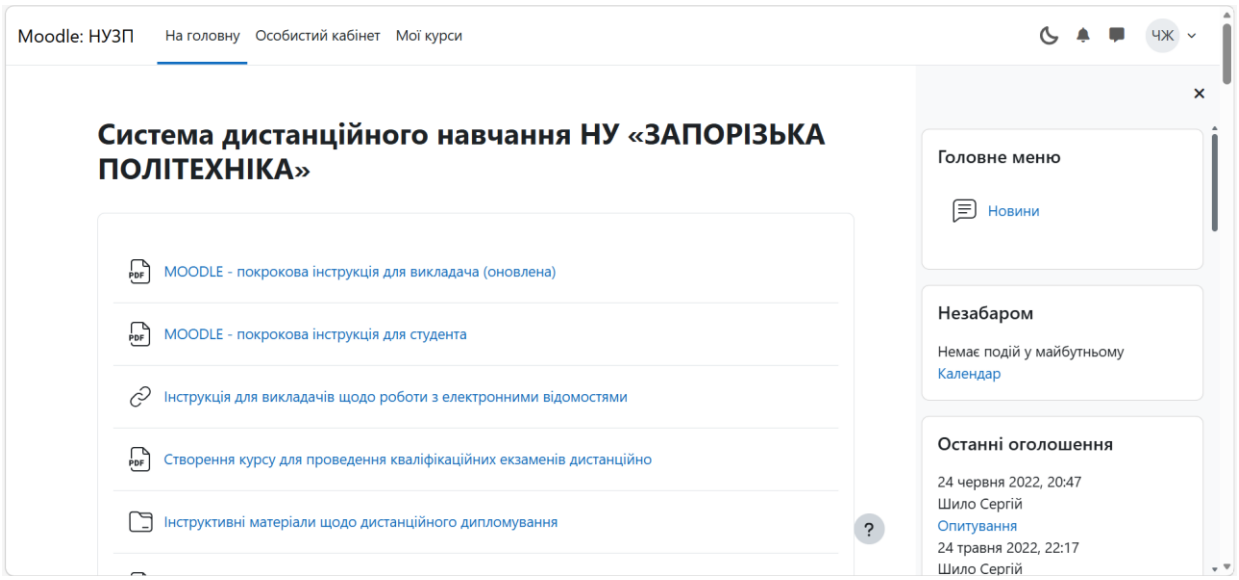


Рисунок 1.2 – Головний екран Moodle [4]

1.2.2 Застосунок Google Classroom

Google Classroom – це хмарна онлайн-платформа для навчання від Google, яка допомагає викладачам і студентам організовувати навчальний процес. Його ключова перевага в тому, що він має простий і зрозумілий інтерфейс. Платформа також підтримує організацію матеріалів за темами,

перегляд успішності у вигляді зручних списків і автоматичну синхронізацію через Google Drive. Також вона інтегрована з сервісом Google Meet, що забезпечує можливість проведення онлайн-занять [5]. Проте і ця система не має спеціалізованих інструментів для управління практикою студентів, а саме місць для закріплення практики, керівників та формування звітів.

Переваги:

– зручний та зрозумілий інтерфейс;
– синхронізація з Google (Диск, Документи, Календар);
– для більшості базових потреб – безкоштовний.

Недоліки:

– користувачі повинні мати Gmail пошти;
– обмежене тестування та слабка аналітика;
– немає розділу контроля стажуваннями студентів.

Вигляд домашній сторінки застосунку Google Classroom продемонстровано на рисунку 1.3.

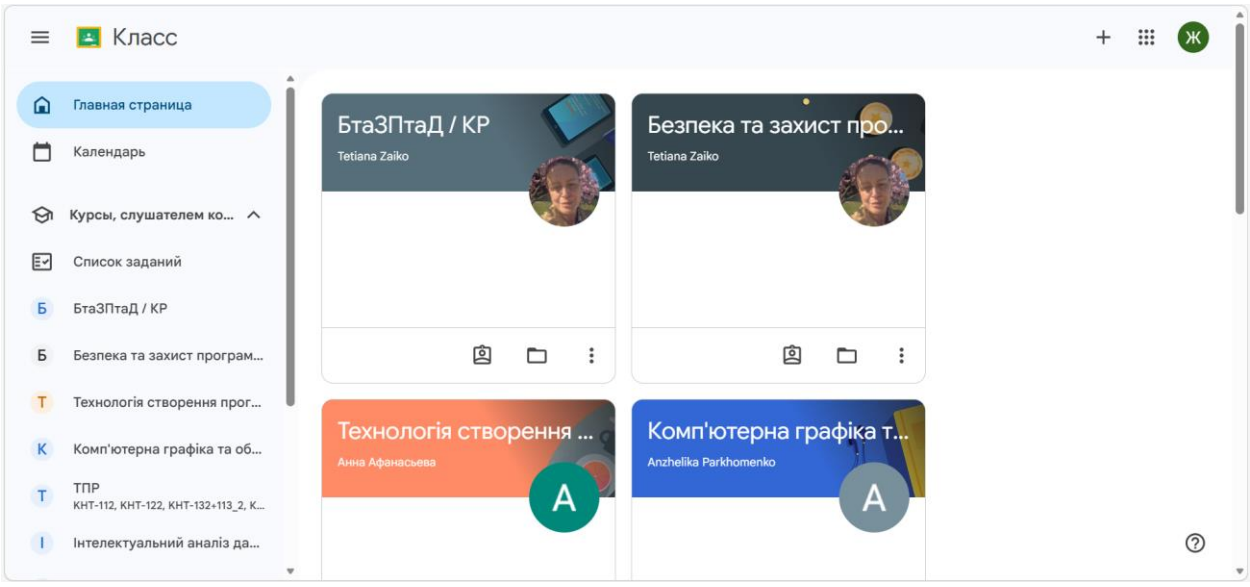


Рисунок 1.3 – Головний екран Google Classroom [5]

1.2.3 internship-management-system (GitHab)

В відкритих репозиторіях, таких як GitHub можна знайти вже створені розробниками приклади open-source системи для управляннями стажувань. internship-management-system є одним з них, він створений для автоматизації обліку стажування працівників у межах певної кампанії, і задовольняє частину вимог, але він був створений відповідно до суб'єктивного бачення його розробника і не був застосований у живому середовищі [6]. Також даний проект був створений для регулювання стажерів в компанії, а не для університетів та не має можливостей оцінювання звітів.

Вигляд домашній сторінки застосунку internship-management-system (GitHab) продемонстровано на рисунку 1.4.

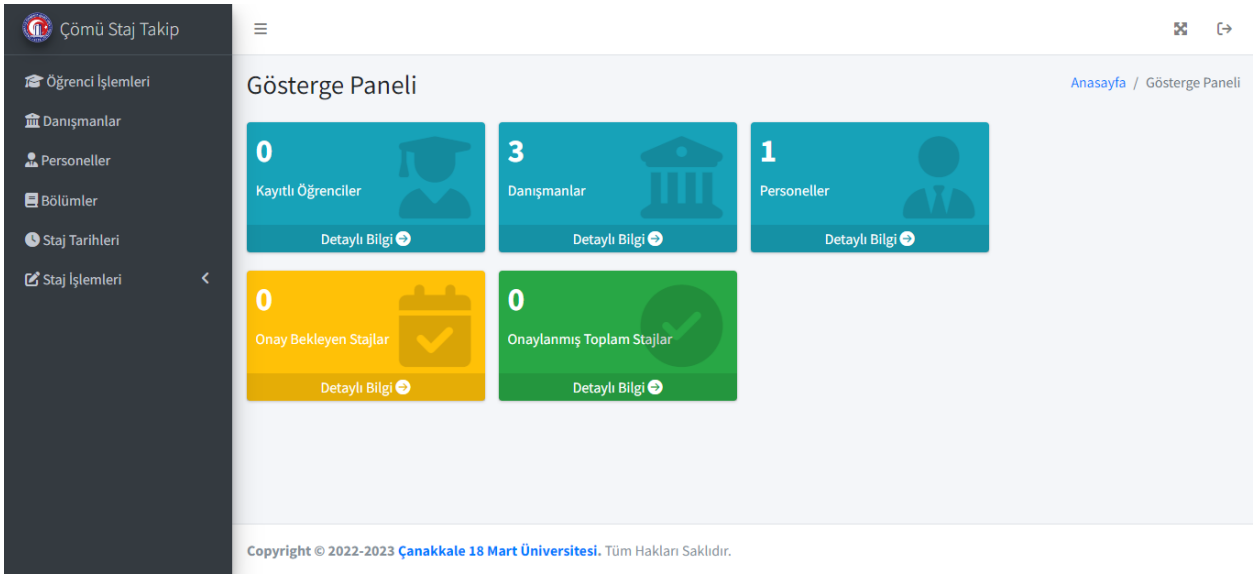


Рисунок 1.4 – Головний екран internship-management-system (GitHab) [6]

1.2.4 Порівняльна таблиця існуючих та розроблюваного програмних продуктів

Було розглянуто сильні та слабкі сторони кожного з застосунків і зведено їх у порівняльну таблицю 1.1 разом із розроблювальним вебдодатком за такими критеріями: автоматизація документів, сповіщення, простота

інтерфейсу, синхронізація даних, управління документами практики, завдання для практики та ролі.

Таблиця 1.1 – Порівняння функціональності застосунків для управління студентськими стажуваннями

Критерій	Moodle	Google Classroom	IMS (GitHub)	Вебзастосунок
1	2	3	4	5
Автоматизація документів	Відсутня	Відсутня	Відсутня	Автогенерація титульних сторінок та рефератів за шаблоном
Сповіщення	Внутрішні та Email	Email та Push	Відсутні	Telegram–бот (дедлайни, оцінки, нагадування)
Простота інтерфейсу	Складний, перевантажений налаштуваннями	Простий, але обмежений функціонально	Середній (адмін-панель)	Інтуїтивний (орієнтований на покрокові дії)
Синхронізація даних	Хмарна або Локальна	Google Cloud	Локальна (MySQL)	Хмарна (PostgreSQL + API)
Управління документами практики	Відсутні	Відсутні	Відсутні	Шаблони, автозаповнення, PDF
Завдання для практики	Є, але не адаптовані під стажування	Є, але не адаптовані під стажування	Є, але без оцінювання	Керівник створює завдання

Продовження таблиці 1.1

1	2	3	4	5
Ролі	Багато ролей, але не для стажувань	Лише викладач та учень	Адміністратор, персонал, викладач та стажер	Студент, керівник практики, адміністратор

1.3 Опис основних вимог до програми

1.3.1 Визначення функціональних вимог

До функціональних вимог розроблювального вебзастосунка для регулювання студентськими стажуваннями відносяться:

а) автентифікація користувачів:

1) захищений вхід за логіном (email) та паролем із використанням токенів доступу (Laravel Sanctum);

2) обмеження доступу до API-ендпоінтів на основі ролей (студент, викладач, адміністратор);

б) можливості студента:

1) переглядати завдання практики;

2) завантажувати звіти;

3) завантажувати шаблон документу;

4) передивлятися попередньо сформовані документи;

5) отримувати повідомлення відповідно статусу роботи;

6) заповнювати дані про місце стажування та іншу необхідну інформацію;

в) можливості викладача:

1) передивлятися відправлені студентом документи;

2) оцінювати виконані роботи;

3) створювати та редагувати завдання для стажування;

4) залишати коментарі до робот;

5) бачити прогрес всіх закріплених за ним студентів (хто здав, хто отримав оцінку);

г) можливості адміністратора:

1) можливість бачити статистику на сайті;

2) можливість керувати користувачами, ролями, місцями практики (створювати);

3) можливість в реальному часі бачити та формувати PDF звіти статистики кафедри;

д) автоматизація роботи з документами:

1) автоматичне заповнення шаблонів сторінок (титульна сторінка, реферат) введеними користувачем даними та даними з PostgreSQL;

2) об’єднання згенерованих сторінок із файлом, завантаженим студентом;

3) автоматична перевірка введених даних на коректність (чи не є поле порожнім, чи відповідає кількість сторінок числовому формату);

4) безпека збереження файлів: шифрування імен файлів при завантаженні на сервер для запобігання несанкціонованому доступу або перейменуванню;

е) додаткові функції:

1) встановлення дедлайнів для кожного завдання;

2) інтеграція з месенджерами: автоматична розсилка повідомлень через Telegram Bot API при зміні статусу звіту, повідомлення від викладача, нового завдання або виставленні оцінки.

1.3.2 Нефункціональні вимоги

До нефункціональних вимог розроблювального вебзастосунка для регулювання студентськими стажуваннями відносяться:

а) вимоги до продуктивності:

1) система повинна забезпечувати швидкий відгук інтерфейсу (не більше 1 секунди для основних операцій);

2) час генерації документів (PDF) не повинен перевищувати 3–5 секунд;

3) API має обробляти запити без значних затримок навіть при одночасній роботі багатьох користувачів;

б) вимоги до безпеки:

1) паролі зберігаються у хешованому вигляді;

2) доступ до API контролюється через токени (Laravel Sanctum);

3) файли звітів повинні зберігатися з зашифрованими або випадково згенерованими іменами;

4) система повинна запобігати несанкціонованому доступу до документів інших користувачів;

5) має бути реалізований захист від типових атак (SQL-ін’єкції, XSS, CSRF);

в) вимоги до надійності:

1) система повинна забезпечувати стабільну роботу без збоїв при стандартному навантаженні;

2) у разі помилки користувач має отримувати інформативне повідомлення без розкриття внутрішньої логіки системи;

г) вимоги до масштабованості:

1) архітектура системи повинна дозволяти додавання нових ролей, модулів та функцій без суттєвих змін у коді;

2) база даних має підтримувати збільшення кількості студентів, звітів та документів без втрати продуктивності;

д) вимоги до зручності використання:

1) інтерфейс має бути інтуїтивно зрозумілим для користувачів;

2) усі основні дії повинні виконуватися у 2-3 кліки;

е) вимоги до сумісності:

1) система повинна працювати у сучасних браузерах (Chrome, Firefox, Edge);

2) API має бути сумісним із фронтендом на React;

3) підтримка інтеграції з Telegram Bot API та email-сервісами;

ж) вимоги до зберігання даних:

1) база даних PostgreSQL повинна забезпечувати цілісність та узгодженість даних;

2) файли звітів зберігаються у файловій системі або хмарному сховищі з обмеженим доступом;

з) вимоги до підтримуваності:

1) код має бути структурованим відповідно до стандартів Laravel;

2) повинна бути можливість швидкого оновлення залежностей та бібліотек;

и) вимоги до локалізації:

1) інтерфейс повинен підтримувати українську мову;

2) структура системи має дозволяти додавання інших мов у майбутньому.

1.4 Постановка завдання роботи

Розробка вебзастосунку для управління студентськими стажуваннями спрямована на вирішення неефективного регулювання студентською практикою. Основним завдання є автоматизація процесів взаємозв’язку між студентами і викладачами, виключення людських помилок при формуванні офіційних документів і скорочення витраченого на це часу. Також до завдань роботи відносяться:

– проаналізувати вже існуючі рішення (Moodle, Classroom) та сформувати функціональні та нефункціональні вимоги для розроблюваного застосунку;

– створити клієнт-серверну архітектуру на основі Laravel (PHP) та React (JS) із використанням бази даних PostgreSQL та REST API для взаємодії компонентів;

– розробити модулі автентифікації, систему керування ролями, механізм гібридної генерації звітів шляхом злиття PDF-файлів та Telegram сповіщення;

– впровадити захист даних через хешування паролів, валідацію вхідних форм та шифрування імен файлів;

– провести перевірку коректності генерації документів та налаштувати Telegram-сповіщення.

Основною ціллю розробки є створення безпечного та коректно налаштованого програмного забезпечення для певного вищого навчального закладу, що потрібно для ефективної роботи з документами, пов’язаними зі студентськими стажуваннями.

1.5 Висновки за розділом 1

Аналіз предметної області показав що нестача автоматизованих процесів у системи навчання, а саме при управлінні студентськими стажуваннями, досі є актуальною проблемою. В цьому розділі було розглянуто програми в освітній сфері, призначені для покращення ефективності й контролю навчання та програму для відстеження стажерів в компанії, але в них відсутні інструментів для управління практикою студентів. Тож розроблюваний вебзастосунок для управляння студентськими стажуваннями з відповідними ролями та завданнями є актуальним та необхідним.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

2.1 Структура системи

2.1.1 Основні компоненти системи

Архітектура застосунку базується на розподілі на окремі логічні шари та сервіси, задля забезпечення модульності та зручності підтримки.

Основними компонентами системи є:

а) клієнтський шар (Frontend):

1) UI-компоненти (React Components);
2) стан застосунку (React Hooks (useState, useEffect));
3) модулі авторизації;
4) модулі роботи з завданнями, звітами, документами;
5) механізми відправлення запитів до API ;

б) серверний шар (Backend):

1) Controllers (приймають запити від фронтенду);
2) Services (реалізують логіку (генерація документів, обробка файлів, сповіщення));
3) Repositories (робота з базою даних через моделі);
4) Models (описують структуру даних);
5) Middleware (авторизація, перевірка ролей, токени доступу);
6) Laravel Sanctum (захист API та керування токенами);

в) шар даних (Database Layer):

1) таблиці студентів, викладачів та ролей;
2) таблиці завдань, звітів та статусів;
3) таблиці місць стажування;

г) шар роботи з файлами та документами:

1) генерація PDF на основі шаблонів;
2) об’єднання сторінок у PDF;
3) шифрування імен файлів;
4) перевірка коректності завантажених документів;

5) зберігання файлів у файловій системі або хмарі;

д) шар інтеграцій (зовнішні сервіси):

1) Telegram Bot API (надсилання сповіщень студентам);

е) шар безпеки:

1) авторизацію через Laravel Sanctum;
2) валідацію даних;
3) CSRF-захист;
4) шифрування файлів;
5) контроль доступу за ролями;
6) логування дій користувачів.

2.1.2 Взаємодія компонентів системи

Взаємодії компонентів залежить від сценаріїв приведених нижче.

Авторизація та доступ:

– Frontend (React) збирає дані (email/пароль) і через Axios (бібліотека для відправки запитів) відправляє POST-запит на сервер;

– Backend (Laravel) через Middleware перевіряє дані. Sanctum генерує унікальний API-токен;

– токен повертається на фронтенд і зберігається в localStorage, додаючись до заголовків усіх наступних запитів для ідентифікації ролі (Студент/Викладач).

Формування та завантаження звіту:

– користувач (Студент) заповнює форму в React-компоненті та прикріплює файл основної частини звіту;

– React формує об'єкт FormData (формат, який дозволяє спакувати і текстові поля (тема, мета), і сам файл звіту в одну відправку) і відправляє його на API-ендпоінт;

– Laravel Controller приймає запит, проводить валідацію (перевірка типів файлів, обов’язковість текстових полів);

Service Layer (DocumentService) виконує такі дії:

– Database: зберігає текстові дані (тема, об'єкт, мета) у PostgreSQL;

– File System: зберігає файл студента з унікальним хеш-іменем;

– DomPDF: формує PDF-документ на основі Blade-шаблону титульної сторінки та реферату, підставляючи дані з БД;

– FPDI: фізично зшиває згенерований PDF з файлом студента.

Результат (шлях до повного звіту) записується в БД, а статус завдання змінюється на «На перевірці».

Перевірка та сповіщення:

– викладач у своєму інтерфейсі отримує список нових звітів (запит GET до API -> Repositories -> PostgreSQL);

– після перевірки викладач виставляє оцінку. Backend оновлює запис у базі даних;

– Laravel фіксує зміну статусу та ініціює роботу модуля інтеграції;

– Telegram Integration Layer формує текстове повідомлення та через Telegram Bot API відправляє його студенту на основі його telegram_id, збереженого в профілі.

Отримання фінального документа:

– студент натискає кнопку «Завантажити готовий звіт»;

– React звертається до сервера;

– Laravel через Middleware перевіряє, чи має цей студент права доступу саме до цього файлу;

– сервер віддає дані , і браузер студента зберігає фінальний PDF-документ.

2.2 Функціонування системи

2.2.1 Основні процеси системи

Функціонування системи можна розглядати через основні сценарії використання програми:

а) процес створення облікових записів та доступу:

1) адміністратор системи створює облікові записи для користувачів (викладачів та студентів) через панель управління. Під час створення вводиться пароль, email та призначається відповідна роль у базі даних PostgreSQL;

2) студенти та викладачі використовують надані адміністратором вхідні дані для входу в систему;

3) при успішному вході сервер (Laravel) генерує API-токен (Sanctum), який передається на фронтенд і зберігається в localStorage для подальшої авторизації запитів;

б) процес подання та формування звітності:

1) студент вводить дані про практику (тема, об'єкт, мета) та вказує місце практики;

2) студент скачує шаблон документу в якому буде далі працювати;

3) студент завантажує основну текстову частину звіту у форматі PDF;

4) система автоматично генерує титульні сторінки та реферат, об'єднуючи їх із завантаженим файлом у єдиний документ;

в) процес оцінювання:

1) викладач переглядає звіт та залишає коментарі;

2) після успішної перевірки викладач виставляє оцінку, що автоматично змінює статус роботи на «Завершено»;

3) система через Telegram Bot API миттєво надсилає студенту повідомлення про результат перевірки.

2.2.2 Логіка роботи системи

Основні принципи роботи системи:

а) принцип клієнтської валідації та швидкості дій:

1) всі вхідні дані (мета, об'єкт, предмет практики) проходять початкову валідацію на React перед відправкою, що дозволяє користувачу миттєво бачити помилки при заповненні форм, без очікування відповіді від сервера;

2) використання механізмів управління станами дозволяє зберігати введені дані у формі, навіть при випадковому перемиканні між вкладками (через React State та LocalStorage), що запобігає втраті даних при заповненні звіту;

б) принцип серверної обробки та автоматизації:

1) процес формування PDF-файлів та з’єднання документів відбувається на Laravel, завдяки чому ресурси браузера студента звільняються та забезпечується стабільність роботи на слабких пристроях;

2) БД PostgreSQL є центральним сховищем, де зберігаються фінальні версії документів, що виключає розбіжності між тим, що бачить студент, і тим, що перевіряє викладач;

в) принцип шифрування та безпеки даних:

1) паролі користувачів ніколи не зберігаються у відкритому вигляді, бо обробляються алгоритмом Bcrypt;

2) доступ до завантажених PDF-звітів здійснюється через контрольовані сервером посилання (Secure Links). Прямий доступ до папки з файлами закритий, а імена файлів перетворюються на унікальні хеші для захисту від перебору імен;

г) принцип інформування змін статусів:

1) система автоматично відстежує зміни статусів у базі даних (наприклад, перехід зі статусу «На перевірці» до «Оцінено») і миттєво ініціює подію відправки сповіщення через Telegram Bot API коли викладач надсилає оцінку або коментар.

2.3 Вибір мов програмування та інструментарію для розробки програми

2.3.1 Мови програмування

Для розробки програмного забезпечення регулювання студентськими стажуваннями було обрано на фронтенді мову програмування React [7], а на бекенді – PHP [8] і використано фреймворк Laravel [9]. Даний вибір був обумовлений такими ключовими факторами:

– продуктивність та розділення логіки. Мова програмування PHP спеціалізується на веб-розробці, а її сучасні версії забезпечують високу швидкість обробки скриптів та низьке споживання ресурсів сервера. React відповідає за швидкий та динамічний інтерфейс користувача, а Laravel – за обробку даних і логіку. Завдяки цьому досягається чітке розділення фронтенду й бекенду;

– підтримка API. Laravel дозволяє створити стандартизований REST API, задля забезпечення структурованої передачі даних у форматі JSON. A React ефективно взаємодіє з цими ендпоінтами за допомогою асинхронних HTTP-запитів;

– швидкість розробки. Laravel має готову систему маршрутизації та вбудовану автентифікацію, що значно прискорює розробку бекенду;

– популярність. React, PHP і Laravel поширені у веб-розробці і мають велику спільноту. Також вони добре документовані, що значно полегшує вирішення проблем з кодом;

– безпека. Laravel на базі PHP має вбудовані механізми захисту такі як захист від CSRF , хешування паролів та система авторизації (Sanctum);

– зручність роботи з інтерфейсом. Завдяки React можна легко створювати компоненти, легко керувати станами додатку та будувати сучасний UI.

2.3.1.1 Опис переваг та обмежень вибраної мови React

Переваги:

– дозволяє створювати багаторазові UI-компоненти та спрощує підтримку інтерфейсу;

– має високу продуктивність завдяки віртуальному DOM та оптимізованому оновленню елементів;

– не має структури проєкту, що дозволяє адаптувати архітектуру під потреби системи;

– має велику екосистему (бібліотеки для маршрутизації, стану, форм, запитів тощо);

– має активну спільноту, завдяки чому проблеми швидко вирішуються, і можна знайти велику кількість готових рішень.

Обмеження:

– потребує додаткових бібліотек (React Router, Axios, Context), оскільки сам по собі React є лише UI-шаром;

– має високу динамічність екосистеми (часті оновлення можуть ускладнювати підтримку).

В таблиці 2.1 наведено порівняння мов програмування.

Таблиця 2.1 – Порівняння мов програмування

Критерій	React	Angular	Vue
Мова програмування	JavaScript / TypeScript	TypeScript	JavaScript / TypeScript
Гнучкість архітектури	Висока	Низька	Висока
Продуктивність	Висока завдяки Virtual DOM	Висока	Середня
Масштабованість	Висока	Дуже висока	Середня

Продовження таблиці 2.1

Критерій	React	Angular	Vue
Підтримка спільноти	Дуже велика	Велика	Велика
Підходить для великих проєктів	Так	Так	Частково
Кількість бібліотек	Дуже велика	Велика	Середня

2.3.1.2 Опис переваг та обмежень вибраної мови PHP та Laravel

Переваги:

– наявність стабільної екосистеми PHP, яка спрощує розробку серверної логіки;

– використання архітектури Laravel задля забезпечення чіткого розділення відповідальностей;

– потужна ORM Eloquent, що полегшує роботу з базою даних;

– вбудовані механізми безпеки (CSRF-захист, валідація, шифрування);

зручна реалізація токен-орієнтованої автентифікації через Laravel Sanctum;

– розвинена екосистема інструментів (Blade-шаблони, DomPDF);

– висока швидкість розробки завдяки artisan-командам та готовим патернам.

Обмеження:

– PHP поступається продуктивністю мовам, орієнтованим на високонавантажені системи (Go, Node.js, Java);

– Laravel є досить важким фреймворком і потребує більше ресурсів, ніж мінімалістичні рішення;

– для оптимальної продуктивності необхідно застосовувати кешування, черги та оптимізацію запитів;

– складні проєкти потребують акуратності в архітектурі, щоб уникнути надмірної логіки в контролерах.

В таблиці 2.2 наведено порівняння мов програмування.

Таблиця 2.2 – Порівняння мов програмування

Критерій	PHP	JavaScript (Node.js)	Python
Продуктивність	Висока для веб	Дуже висока	Середня
Швидкість розробки	Дуже висока	Висока	Висока
Підтримка PostgreSQL	Повна	Повна	Повна
Інтеграція з React	Дуже проста	Дуже проста	Проста
Спільнота	Найбільша	Дуже велика	Велика
Вбудовані засоби Web	Сесії, Cookies, Роутинг	Потребує багато модулів	Потребує фреймворків

2.3.2 Середовище розробки

Visual Studio Code - це безкоштовний редактор коду, який працює на багатьох операційних системах. Саме його було обрано для розробки програмного забезпечення завдяки його основній перевазі – легкості використання, але в той же час, він має високу ефективність, завдяки чому, він працює на комп’ютерах з середніми характеристиками. Даний редактор підтримує значну кількість мов програмувань, таких як JavaScript, PHP, C# та інші. Крім того, редактор дозволяє додавати фреймворки, інструменти налагодження і загалом має велику розвинену систему розширень.

Також, що сьогодні досить важливо, Visual Studio Code має інтеграцію з Git та іншими контролями версій. До того ж, в нього є сучасні інструменти

задля зручності в роботі з кодами по типу автопідкреслення та автоматичного доповнення.

Безпосередньо, у редакторі присутній дебагінг та вбудований термінал.

2.3.2.1 Обґрунтування вибору та опис його можливостей

Завдяки універсальності та легкості у використанні було обрано Visual Studio Code для розробки сучасного вебзастосунку.

Основні можливості:

– редактор коду (автодоповнення, підсвітка синтаксису, швидка навігація по проєкту, форматування коду);

– дебагінг (покрокове виконання коду, перегляд змінних, стеку викликів та контролю виконання коду);

– інтеграція з системами контролю версій (вбудована підтримка Git для відстеження змін, створення комітів, гілок та роботи з репозиторіями);

– термінал (вбудований командний рядок для виконання команд Node.js, PHP, Composer, npm та інших інструментів без виходу з редактора);

– розширення (підтримка великої кількості плагінів для роботи з React, PHP та Laravel);

– підтримка роботи з API (можливість використання розширень для тестування REST API та HTTP-запитів безпосередньо в середовищі розробки);

– кастомізація середовища (можливість налаштування тем оформлення, гарячих клавіш та робочого простору під потреби розробника);

– підтримка мультиплатформності (робота на Windows, Linux та macOS);

– робота з великими проєктами (ефективне управління структурою файлів і папок у складних вебдодатках).

2.3.3 База даних

Для забезпечення роботи застосунку використано комбіновану систему збереження даних: PostgreSQL [10] для надійного зберігання структурованої інформації, Laravel File Storage [9] для керування завантаженими документами та згенерованими PDF-звітами та LocalStorage [11] для тимчасового зберігання чернеток звітів на стороні клієнта.

2.3.3.1 PostgreSQL

PostgreSQL – це СУБД з відкритим кодом, яка використовує SQL-мову для управління даними.

Основною перевагою даної БД є її надійність та стабільність роботи. Вона підтримує складні SQL-запити та розширюваність (підтримка JSON). Також вона відповідає стандартам ACID.

Недоліком даної БД є те, що вона складна в налаштуванні та потребує багато ресурсів.

В таблиці 2.3 наведено порівняння БД.

Таблиця 2.3 – Порівняння БД

Критерій	PostgreSQL	MySQL	MongoDB
Тип БД	Реляційна	Реляційна	Нереляційна
Цілісність даних	Дуже висока (строге дотримання ACID)	Висока	Середня
Складні запити	Ідеально	Добре	Обмежено
Робота з JSON	Відмінна	Базова	Рідна

Продовження таблиці 2.3

Безпека	Шифрування (на сервері)	Базові механізми, залежить від конфігурації	Шифрування, ролі, але складніша конфігурація
Складність інтеграції	Складна (потрібен сервер) але легка з Laravel	Проста	Середня
Підходить для великих даних	Так	Так	Так

2.3.3.1 Laravel File Storage

Laravel File Storage – це вбудована система, яка дозволяє легко працювати з файлами в Laravel, не прив’язуючись до конкретного місця їх зберігання.

Основною перевагою є її проста інтеграція у фреймворк Laravel та зручність роботи з файлами. Також, вона дозволяє легко завантажувати, зберігати та видаляти файли, а також підтримує різні типи сховищ (локальне та хмарне), що спрощує організацію зберігання файлів у веб-додатку.

Недоліком є те, що вона не призначена для роботи зі структурованими даними та має обмежені можливості пошуку і фільтрації. Крім того, ефективність роботи залежить від файлової системи сервера.

2.3.3.1 LocalStorage

LocalStorage – це вбудоване у веб-браузер сховище, яке дозволяє веб-сайтам зберігати дані (у форматі ключ - значення) безпосередньо на комп’ютері користувача.

Воно легке у використанні та зберігає дані користувача на стороні клієнта в браузері. Також воно має швидкий доступ до даних, бо не робить запити до сервера.

Недоліком є слабкий рівень безпеки, через що в воно не підходить для зберігання конфіденційних даних. Окрім того, воно працює у межах одного браузера і має маленький обсяг пам’яті (5-10 МБ).

2.4 Вимоги до технічних і програмних засобів

2.4.1 Необхідні технічні засобів для розробки та експлуатації

Для розробки та експлуатації застосунку управління студентськими стажуваннями необхідні:

– комп’ютер або ноутбук для розробки та тестування коду;

– інтернет для роботи з API та базою даних.

2.4.2 Вимоги до апаратного та програмного забезпечення

Апаратне забезпечення:

– процесор: Intel Core i5 або еквівалент (необхідно для швидкої компіляції React-компонентів та обробки запитів до бази даних PostgreSQL);

– оперативна пам’ять: рекомендовано – 16 Гб (одночасний запуск сервера Laravel, середовища Node.js, бази даних PostgreSQL та браузера з багатьма вкладками активно споживає ресурси пам'яті);

– накопичувач: SSD-диск рекомендовано від 20 Гб (забезпечує швидкий доступ до бази даних та ефективну роботу Laravel File Storage при завантаженні та читанні PDF-документів);

– мережеве обладнання: Wi-Fi для взаємодії з Telegram Bot API;

Програмне забезпечення:

– операційна система: Windows 10/11, Linux (Ubuntu) або macOS;

– середовище виконання: PHP (версія 8.2.30) та Node.js (версія 22.16.0);

– менеджери пакетів: Composer (для PHP) та npm (для JavaScript (версія 10.9.2));

– фреймворки: Laravel 10/11 та React.js;

– система керування базами даних: PostgreSQL;

– інструменти розробки: Visual Studio Code, Postman (тестування API), pgAdmin (керування БД);

– бібліотеки генерації звітів: DomPDF / Laravel-Snappy;

– браузери: будь-який сучасний браузер із підтримкою стандарту HTML5 та LocalStorage API.

2.5 Висновки за розділом 2

У другому розділі було описано повну архітектуру для вебзастосунку управління студентськими стажуваннями, основні компоненти системи та їхню взаємодію, функціонування системи та її логіку роботи. Також було обґрунтовано вибір мов програмування React та PHP з Laravel за їх сучасність, продуктивність, гнучкість та широкі можливості для розробки веб-додатків. Сучасне середовище розробки – Visual Studio Code, яке забезпечує зручність написання, налагодження та тестування коду й БД – PostgreSQL.

3 РОЗРОБКА ПРОГРАМИ

3.1 Архітектура системи

При створенні архітектури вебзастосунку, логіка була розподілена між клієнтською стороною (фронтенд на React) та серверною стороною (бекенд на PHP з Laravel). Це та розподілена логіка яка використовувалася при написанні коду, робить підтримку та розширення функціональності даного вебзастосунку досить зручним, бо дозволяє покращувати та оновлювати кожну з сторін незалежно один від одного.

Взаємодія між цими ізольованими частинами платформи реалізована через безстатусний інтерфейс REST API. Ключовим інструментом конфігурації цієї взаємодії на стороні бекенду виступає файл маршрутизації, який чітко зіставляє вхідні HTTP-методи (GET, POST) із відповідними методами логіки в контролерах. Для забезпечення безпеки персональних даних студентів та викладачів, усі критичні маршрути (адміністрування користувачів) об'єднані у захищену групу.

Основні компоненти архітектури:

а) view на React Frontend:

1) екрани та компоненти: TeacherDashboard, StudentDashboard, AdminDashboard, Login, Dashboard (відповідають за побудову інтерфейсу та збір даних від користувача);

2) State Management: хуки useState та useEffect, які зберігають результати пошуку та список обраних ID;

б) API layer (Axios):

1) перетворює дії користувача в HTTP-запити (GET, POST) та доставляє JSON-відповіді назад у компоненти;

2) обробляє помилки з’єднання та заголовки автентифікації;

в) controller на PHP та Laravel Backend:

1) маршрути: приймають запит від Axios;

2) логіка: SupervisorController перевіряє валідацію даних (наприклад, чи масив ID не порожній) та викликає методи моделей;

3) форматування: перетворює сирі дані з БД у чистий JSON-формат для відправки на фронтенд;

г) model (модель та БД):

1) моделі: Student, User та інші. Описують зв'язки (наприклад, викладач має багато студентів);

2) БД: зберігає інформацію про студентів, викладачів та їхні зв’язки;

д) services (Сервіси та Middleware):

1) Auth Service: перевіряє, чи має користувач права доступу;

2) Database Service: обробляє SQL-запити до таблиць через міграції.

Схема взаємодії компонентів системи представлена на рисунку 3.1.

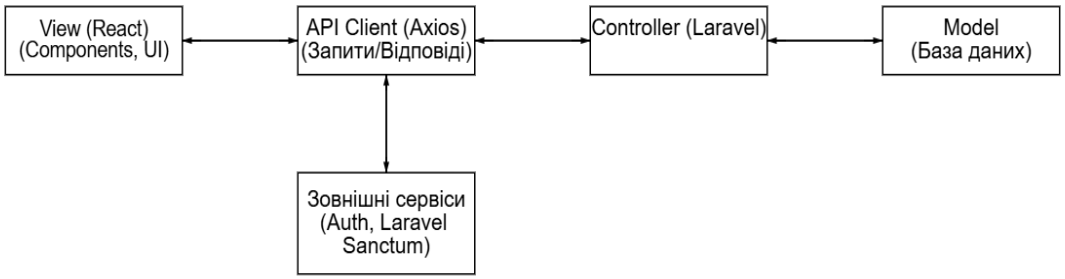


Рисунок 3.1 – Схема взаємодії компонентів системи

3.1.1 Структурна схема системи

Даний вебзастосунок був розроблений за принципом модульної архітектури з чітким розподілом логіки, а саме код системи розділено на незалежні функціональні модулі представлені в схемі на рисунку 3.2.

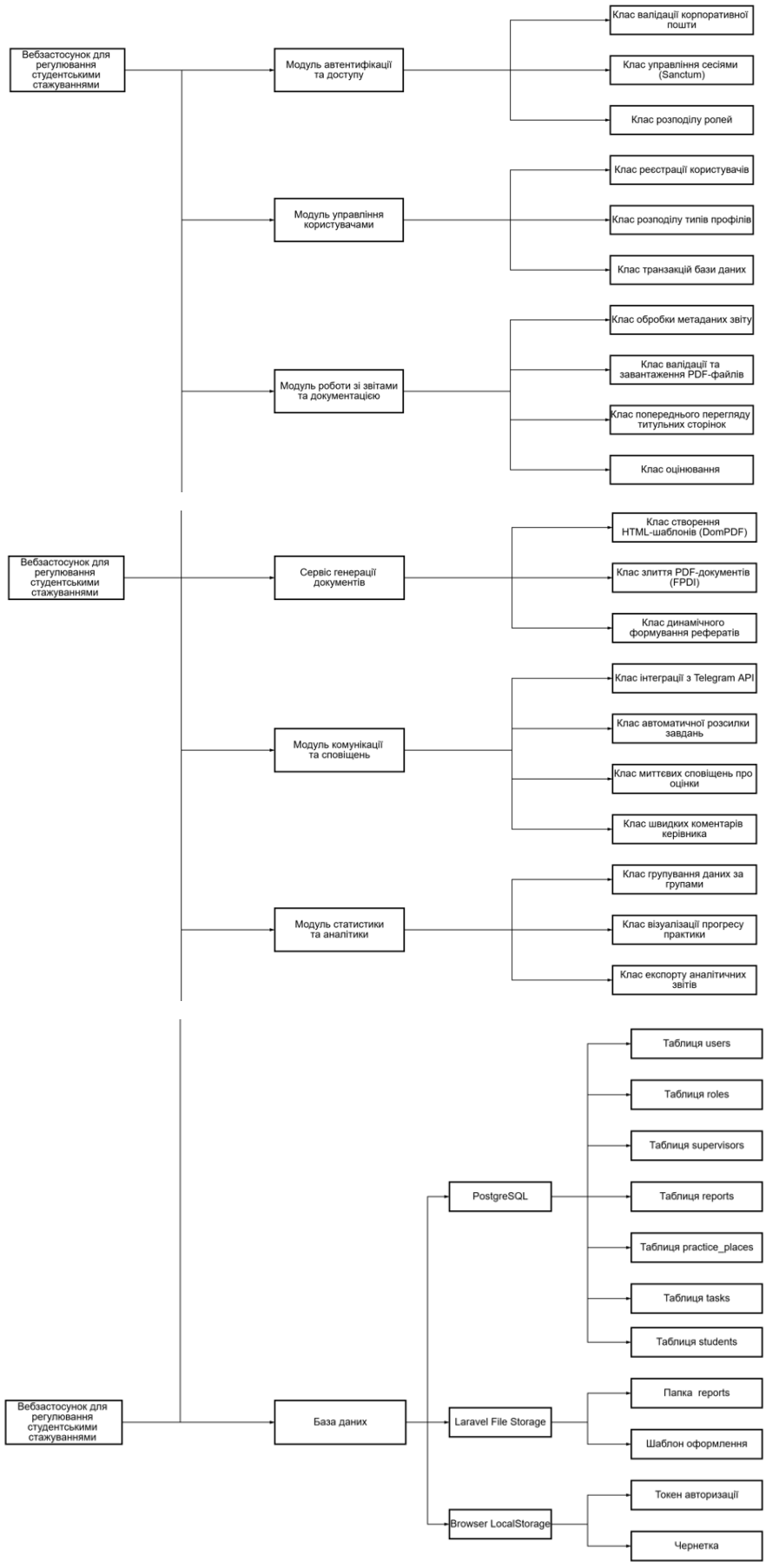


Рисунок 3.2 – Структурна схема системи

3.2 Функціонування системи

3.2.1 Функціональна схема

У вебзастосунку для регулювання студентськими стажуваннями використовується модульна архітектура, завдяки якій, забезпечується правильна обробка даних, автоматична генерація офіційної документації університету та комунікація між учасниками навчального процесу (викладачем та студентом через Telegram). У системі використовуються реляційна база даних (PostgreSQL), захищене файлове сховище на сервері (Laravel File Storage) та локальне сховище браузера (Browser LocalStorage) для забезпечення безпеки та цілісності даних. Розглянемо основні алгоритми, що лежать в основі розробленої програми (рис. 3.3).

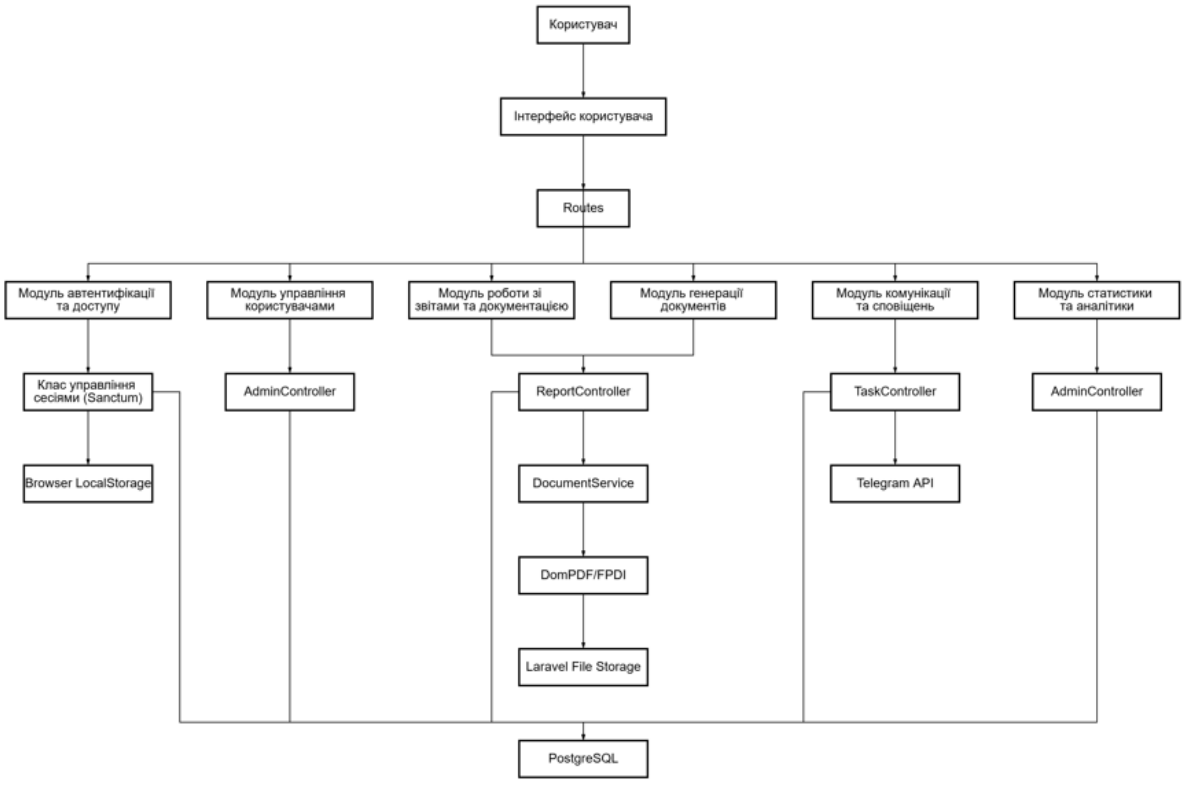


Рисунок 3.3 – Функціональна схема розробленої програми

3.2.2 Опис процесу обміну даними у системі

Процес обміну даними у системі організовано за принципом клієнт-серверної взаємодії з використанням REST API та механізмів локального кешування, що забезпечує цілісність інформації на всіх рівнях сховища. Усе це відбувається автоматично і без перезавантаження сторінки в браузері, що робить систему дуже швидкою для користувача. Фронтенд на React лише збирає дані та відправляє їх, а бекенд на Laravel працює як надійний фільтр, який перевіряє кожну дію на безпеку і не пропускає помилки далі в базу даних.

Основні етапи потоку даних (рис. 3.4):

– введення даних користувачем. Через вебінтерфейс користувач вводить дані (наприклад, заповнює форму звіту або створює нове завдання);

– перевірка сесії та маршрутизація. Перед відправкою запиту система перевіряє наявність токена доступу в LocalStorage. Далі запит через систему маршрутів (Routes) потрапляє до відповідного контролера на бекенді;

– валідація та обробка в Middleware. Пакет Laravel Sanctum проводить автентифікацію запиту, звіряючи дані з PostgreSQL. Після успішної перевірки викликається контролер або сервіс (наприклад, DocumentService);

– розподілене збереження даних. Залежно від типу, дані зберігаються паралельно: метадані та записи про операції фіксуються у базі даних PostgreSQL, а згенеровані або завантажені документи (PDF) архівуються у Laravel File Storage;

– синхронізація та оновлення інтерфейсу. Після підтвердження успішного запису сервер повертає статус виконання, на основі якого інтерфейс оновлює стан LocalStorage (наприклад, очищує чернетки) та візуально відображає зміни які побачить користувач.

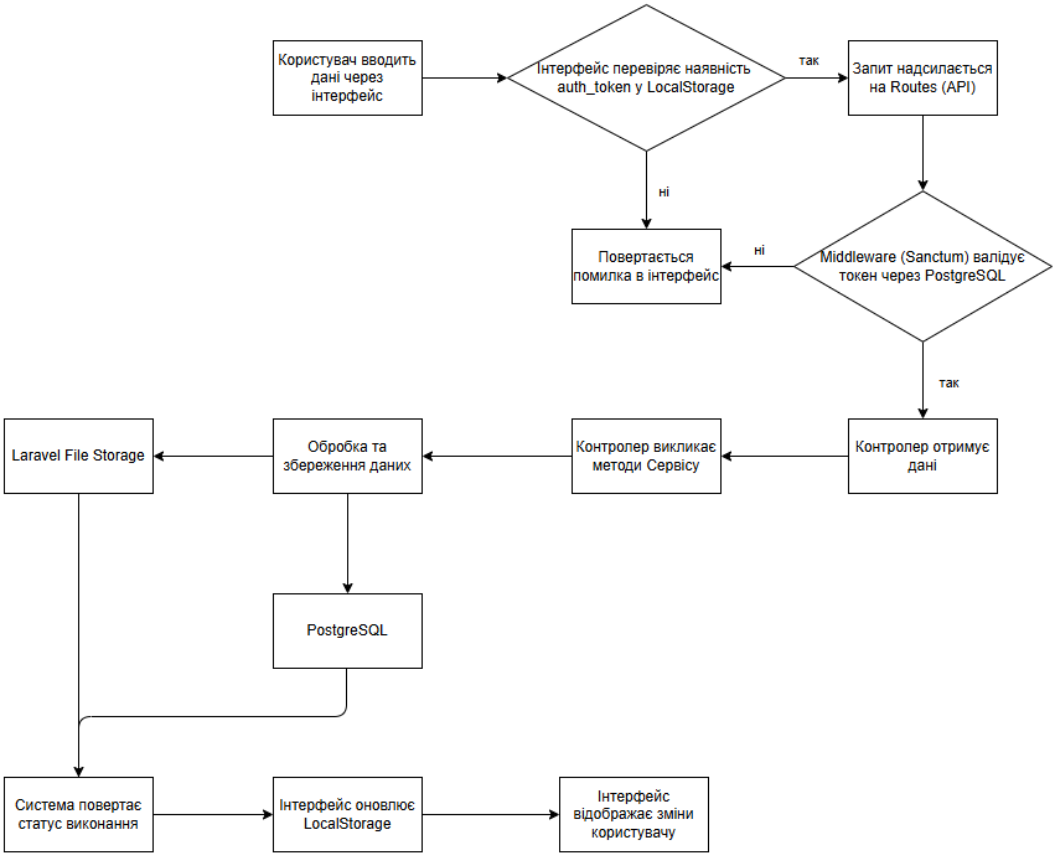


Рисунок 3.4 – Схема процесу обміну даними у системі

3.2.3 Сценарії використання

Сценарії використання вебзастосунка:

а) сценарій «Адміністрування та управління доступом»:

1) адміністратор авторизується в системі через API;

2) адміністратор створює облікові записи викладачів та студентів у модулі управління користувачами;

3) адміністратор призначає ролі та групує студентів за кафедрами;

4) система генерує токени доступу (Sanctum) та зберігає записи в PostgreSQL;

б) сценарій «Створення та подання звіту студентом»:

1) студент заповнює текстові поля практики, які автоматично зберігаються в LocalStorage як чернетка;

2) студент завантажує файл звіту у форматі PDF через вебінтерфейс;

3) система зберігає файл у Laravel File Storage та реєструє певні дані в базі даних;

4) студент перевіряє попередньо сформований документ і відправляє його на перевірку викладачу;

в) сценарій «Перевірка та оцінювання викладачем робіт»:

1) викладач додає студентів до своєї групи для кураторства через відповідне поле;

2) викладач бачить в таблиці список надісланих звітів та завантажує PDF для перегляду;

3) викладач приймає звіт і виставляє оцінку або відхиляє його з коментарем про доопрацювання;

4) система оновлює статус запису в PostgreSQL та надсилає відповідне сповіщення;

г) сценарій «Комунікація та сповіщення через Telegram»:

1) викладач надсилає завдання студентам (може бути особисте повідомлення, або всім закріпленим за ним студентам) через інтерфейс TaskController;

2) система взаємодіє з Telegram API для передачі повідомлення;

3) студент миттєво отримує сповіщення про нове завдання або зміну статусу звіту в Telegram-боті;

д) сценарій «Аналітика та моніторинг успішності»:

1) адміністратор відкриває вкладку статистики;

2) система автоматично робить запит до методу getDetailedStatistics у AdminController;

3) контролер звертається до PostgreSQL, витягує дані про кількість студентів, розраховує середній бал та відсоток проходження практики для кожної групи окремо;

4) система формує відповідь за поточний рік, що включає загальнокафедральні показники та детальну статистику у кожної з груп;

5) адміністратор бачить на екрані актуальний прогрес груп;

6) адміністратор натискає на відповідну кнопку та генерує файл зі статистикою.

Схема розподілу прав доступу зображена на рисунку 3.5.

Рисунок 3.5 – Схема розподілу прав доступу

3.3 Опис модулів системи

3.3.1 Модуль автентифікації та доступу

3.3.1.1 Алгоритм автентифікації та доступу

Алгоритм автентифікації забезпечує багаторівневу перевірку: спочатку на стороні клієнта, а потім на сервері. В разі успішної автентифікації система генерує токен та ініціює механізм авторизації, який направляє користувача до відповідного кабінету. Процес автентифікації відбувається за такою схемою (рис. 3.6):

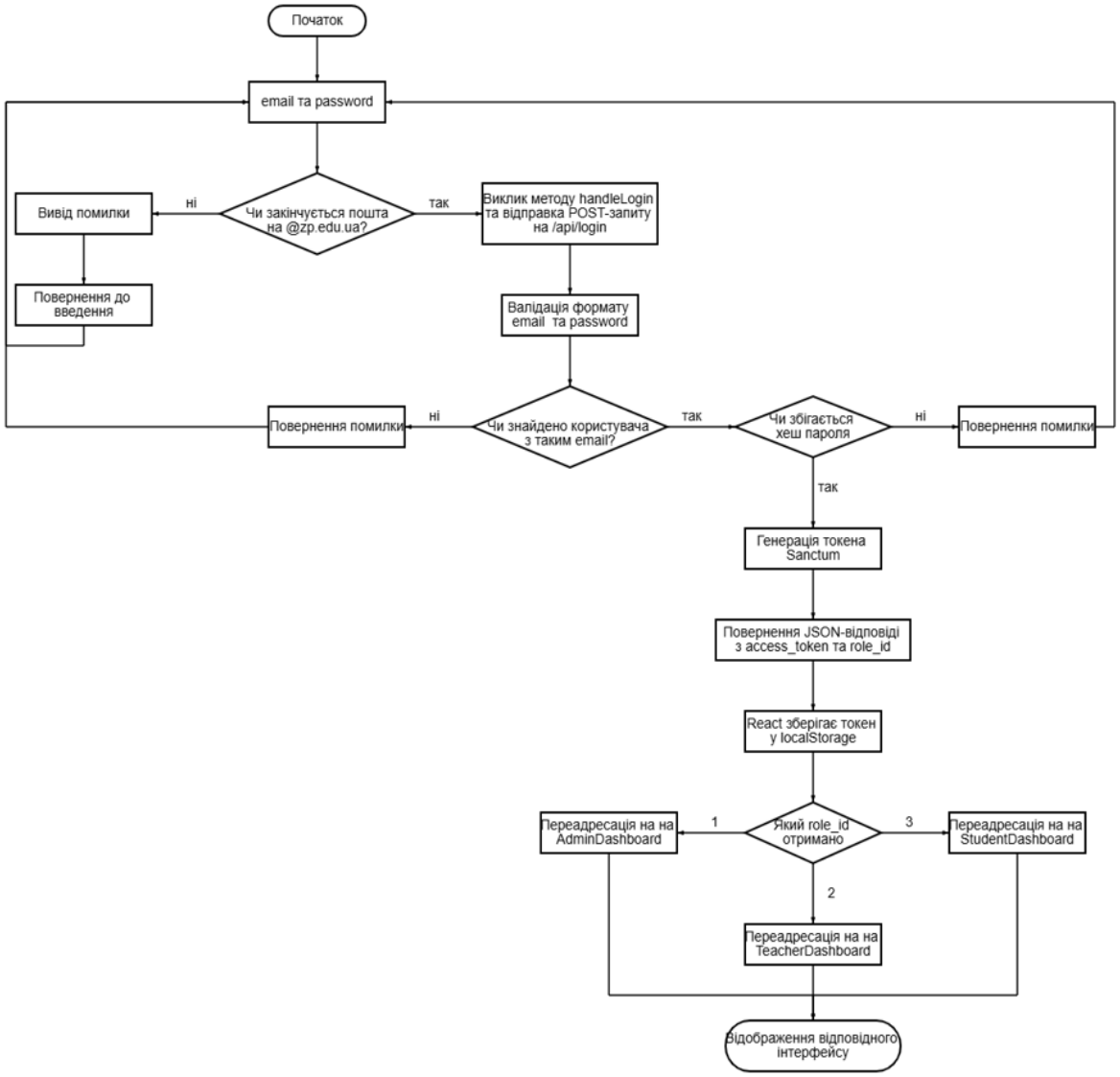


Рисунок 3.6 – Алгоритм автентифікації та доступу користувача

3.3.1.2 Функціональність модуля

До функцій модуля автентифікації та доступу користувача відносяться:

– валідація даних на бекенді (наприклад, обов’язкове використання корпоративної пошти @zp.edu.ua);

– перевірка збіжності введених користувачем даних та хешованих даних в БД;

– створення унікальних токенів доступу для кожного користувача;

– розрізнення трьох видів користувачів та їх перенаправлення до відповідних кабінетів;

– збереження стану сесії в локальному сховищі.

3.3.1.3 Реалізація та використані технології

Модуль реалізовано за допомогою таких технологій:

– Laravel Sanctum (для управління API-токенами);

– Bcrypt hashing (для безпечного шифрування та зберігання паролів у базі даних);

– React Router Dom (для організації захищених маршрутів на стороні клієнта та перенаправлення згідно з ролями);

– Axios (для виконання асинхронних HTTP-запитів до API сервера);

– Browser LocalStorage (для збереження токенів доступу).

Основні класи та файли модуля:

– AuthController.php (серверний контролер, що містить логіку входу (login) та виходу (logout) із системи);

– User.php (модель Eloquent, що описує структуру користувача та його зв'язок із ролями);

– Login.jsx (компонент React, що реалізує інтерфейс форми входу та первинну валідацію корпоративної пошти);

– Dashboard.jsx (компонент для розподілу прав доступу та відображення відповідного кабінету користувача (студента, викладача або адміністратора).

Ключові фрагменти коду, такі як валідація корпоративного домену @zp.edu.ua та генерація токена, наведено в Додатку Б.1.

Екран входу в систему реалізовано як функціональний компонент React з використанням хуків useState для управління станом форми та useNavigate для перенаправлення користувача після успішної автентифікації (рис. 3.7). Реалізацію серверної частини наведено в Додатку Б.2.

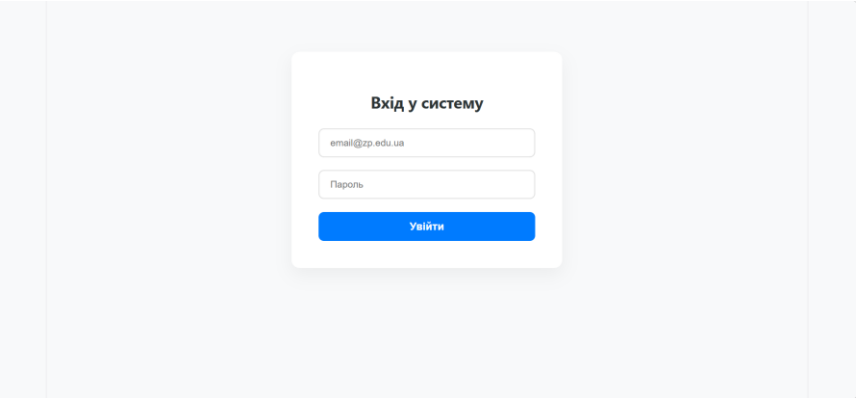


Рисунок 3.7 – Екран входу в систему

3.3.2 Модуль управління користувачами

3.3.2.1 Алгоритм управління користувачами

Алгоритм описує шлях від заповнення форми в React до транзакційного збереження даних у PostgreSQL та відправки повідомлення в Telegram. Процес реєстрування нового користувача відбувається за такою схемою (рис. 3.8):

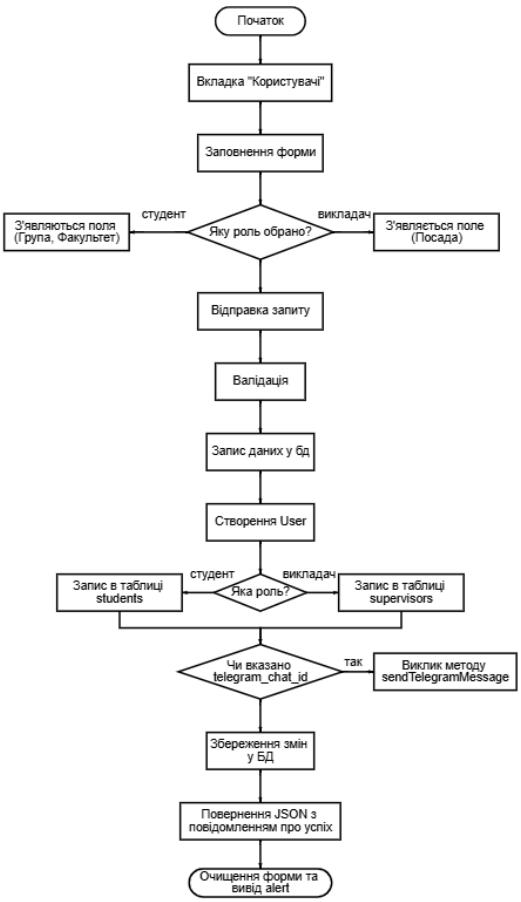


Рисунок 3.8 – Алгоритм управління користувачами

3.3.2.2 Функціональність модуля

До функцій модуля управління користувачами відносяться:

– реєстрація нових користувачів (можливість створення акаунтів для викладачів та студентів системи);

– динамічне формування профілів (автоматичне створення записів у пов’язаних таблицях (students або supervisors) залежно від обраної ролі);

– транзакційне збереження даних (гарантує цілісність БД, якщо при створенні профілю щось піде не так, то і створення користувача (User) також буде скасовано);

– інтеграція з месенджером (автоматична відправка логіну та паролю користувачу в Telegram за наявності chat_id);

– валідація на бекенді (перевірка унікальності електронної пошти та коректності заповнення обов'язкових полів).

3.3.2.3 Реалізація та використані технології

Модуль реалізовано за допомогою таких технологій:

– Laravel DB Transaction (для забезпечення цілісності даних);

– Eloquent ORM (для управління зв’язками між моделями користувачів);

– Telegram Bot API (для реалізації системи миттєвих сповіщень про створення акаунту);

– React hooks useState (для динамічної зміни полів форми вводу залежно від обраної ролі користувача);

Основні класи та файли модуля:

– AdminController.php (storeUser для обробки логіки реєстрації та взаємодії з Telegram);

– AdminDashboard.jsx (клієнтський компонент для відображення форми);

– User.php, Student.php, Supervisor.php (моделі даних для представлення сутностей у системі).

Фрагменти серверної логіки транзакційного запису даних і Telegram-сповіщень наведено в Додатку Б.3.

Адмін-панель реалізовано на React із використанням useState для динамічної зміни полів форми залежно від ролі користувача (рис. 3.9) наведено в Додатку Б.4.

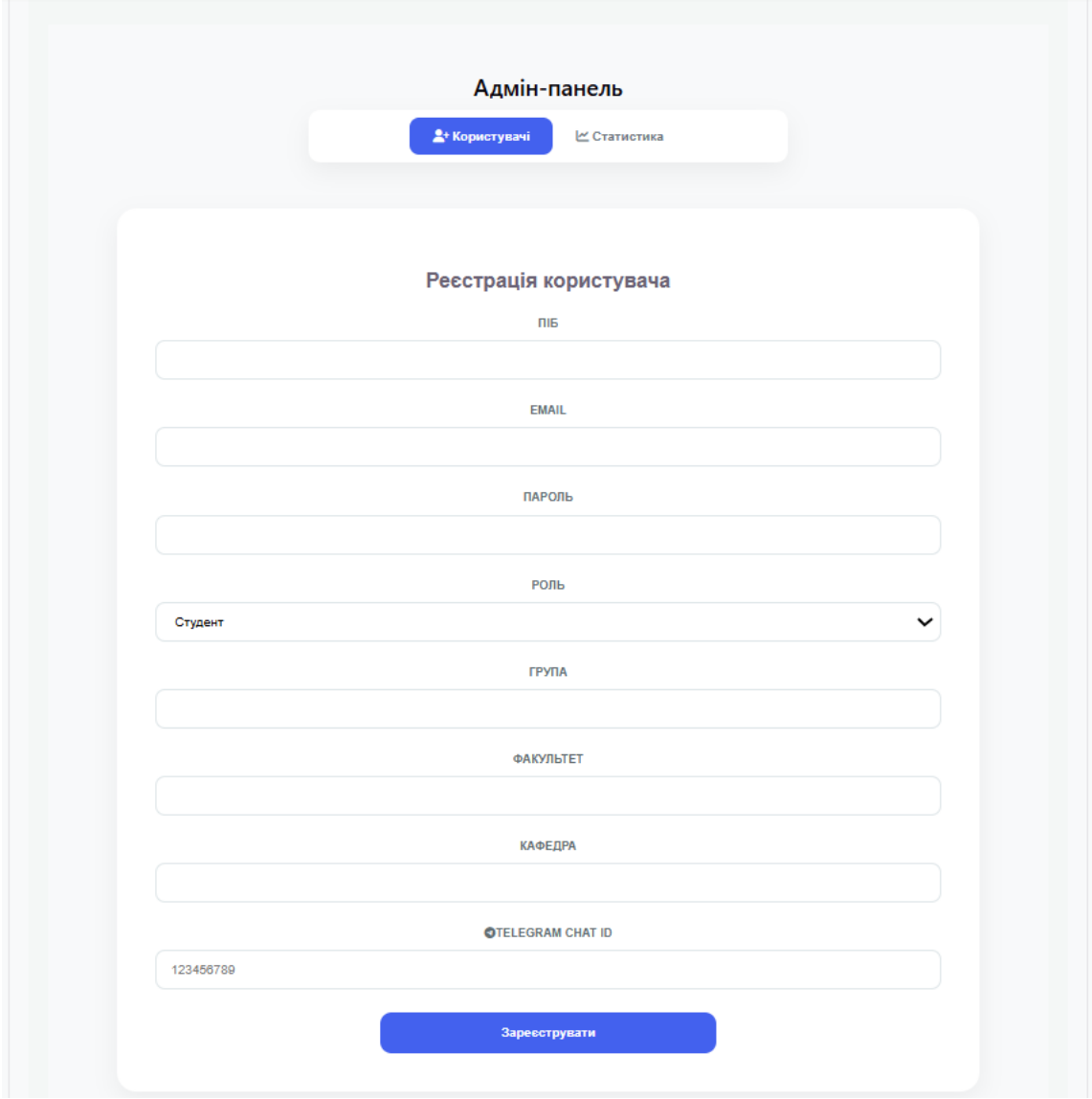


Рисунок 3.9 – Інтерфейс кабінету адміністратора для додавання нових користувачів

3.3.3 Модуль роботи зі звітами та документацією

3.3.3.1 Алгоритм формування та подання звіту

Процес формування звіту про проходження практики реалізовано у вигляді послідовного алгоритму, який забезпечує автоматизацію операцій, валідацію даних та миттєве сповіщення керівника про надходження нового звіту. Алгоритм складається з наступних ключових етапів (див. рис. 3.10):

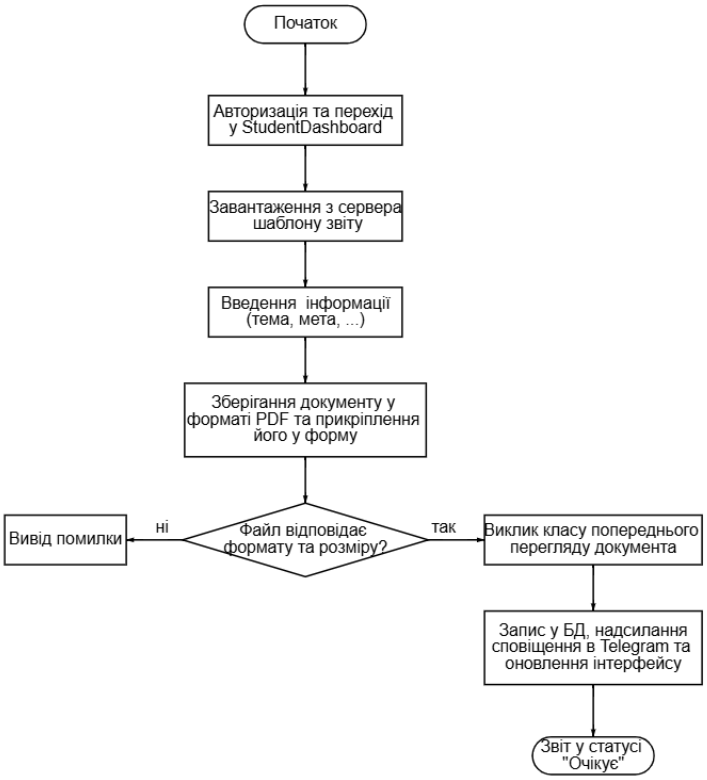


Рисунок 3.10 – Алгоритм формування та подання звіту

3.3.3.2 Функціональність модуля

До функцій модуля роботи зі звітами та документацією відносяться:

– автоматизована генерація документації (формування титульних сторінок та рефератів на основі введених студентом метаданих);

– забезпечення стандартів оформлення (можливість скачування актуальних шаблонів у форматі .docx);

– багаторівнева валідація файлів (перевірка формату PDF, розміру документа та повноти необхідної інформації);

– інтерактивне оцінювання (можливість для викладача виставляти бали та надавати текстові рецензії безпосередньо в системі);

– система зворотнього зв’язку (миттєве сповіщення студентів про результати перевірки звіту через Telegram).

3.3.3.3 Реалізація та використані технології

Модуль реалізовано за допомогою таких технологій:

– Laravel File Storage (для безпечного зберігання та управління доступом до файлів звітів);

– TCPDF / Dompdf (для динамічного створення титульних сторінок у форматі PDF);

– Telegram Bot API (для миттєвих сповіщень викладача про нові звіти та студента про оцінку);

– Axios та FormData (для асинхронної передачі файлів та метаданих без перезавантаження сторінки).

Основні класи та файли модуля:

– ReportController.php (обробка логіки завантаження, валідації та оцінювання звітів);

– DocumentService.php (клас для генерації титульних сторінок та об’єднання PDF-файлів);

– StudentDashboard.jsx та TeacherDashboard.jsx (клієнтські компоненти для взаємодії користувачів зі звітами);

– Report.php (модель даних для управління записами про звіти, їх статуси та бали).

Фрагменти логіки обробки завантаження та інтеграції з Telegram наведено в Додатку Б.5. Візуальний інтерфейс подання звіту з динамічною валідацією реалізовано на React (рис. 3.11) наведено в Додатку Б6.

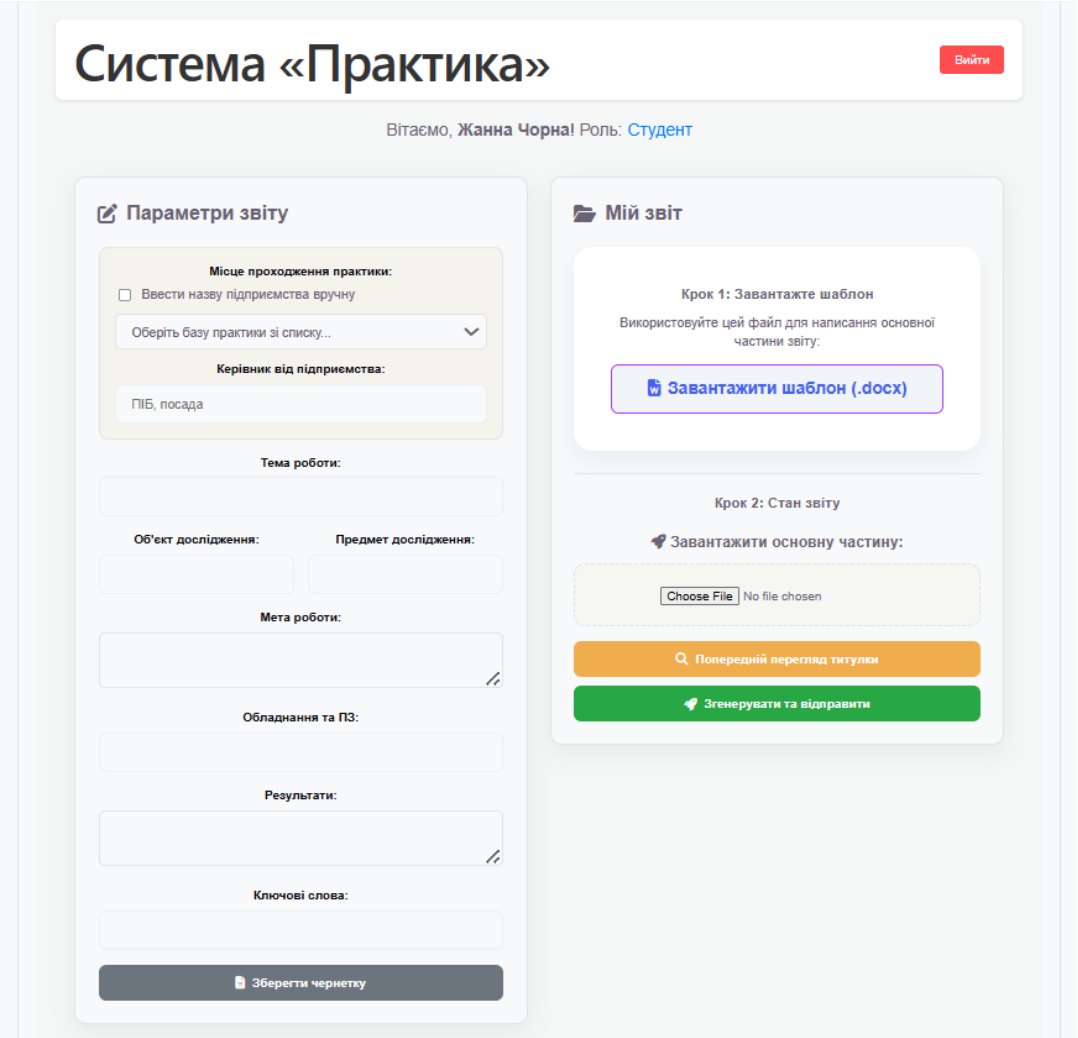


Рисунок 3.11 – Інтерфейс кабінету студента для створення звітів

3.3.4 Модуль генерації документів

3.3.4.1 Алгоритм генерації документів

Алгоритм роботи модуля базується на динамічному поєднанні введених користувачем даних із готовими структурами документів. Процес включає перетворення HTML-коду у PDF-формат та автоматичне злиття кількох файлів у єдиний фінальний документ. Алгоритм складається з наступних ключових етапів (див. рис. 3.12):

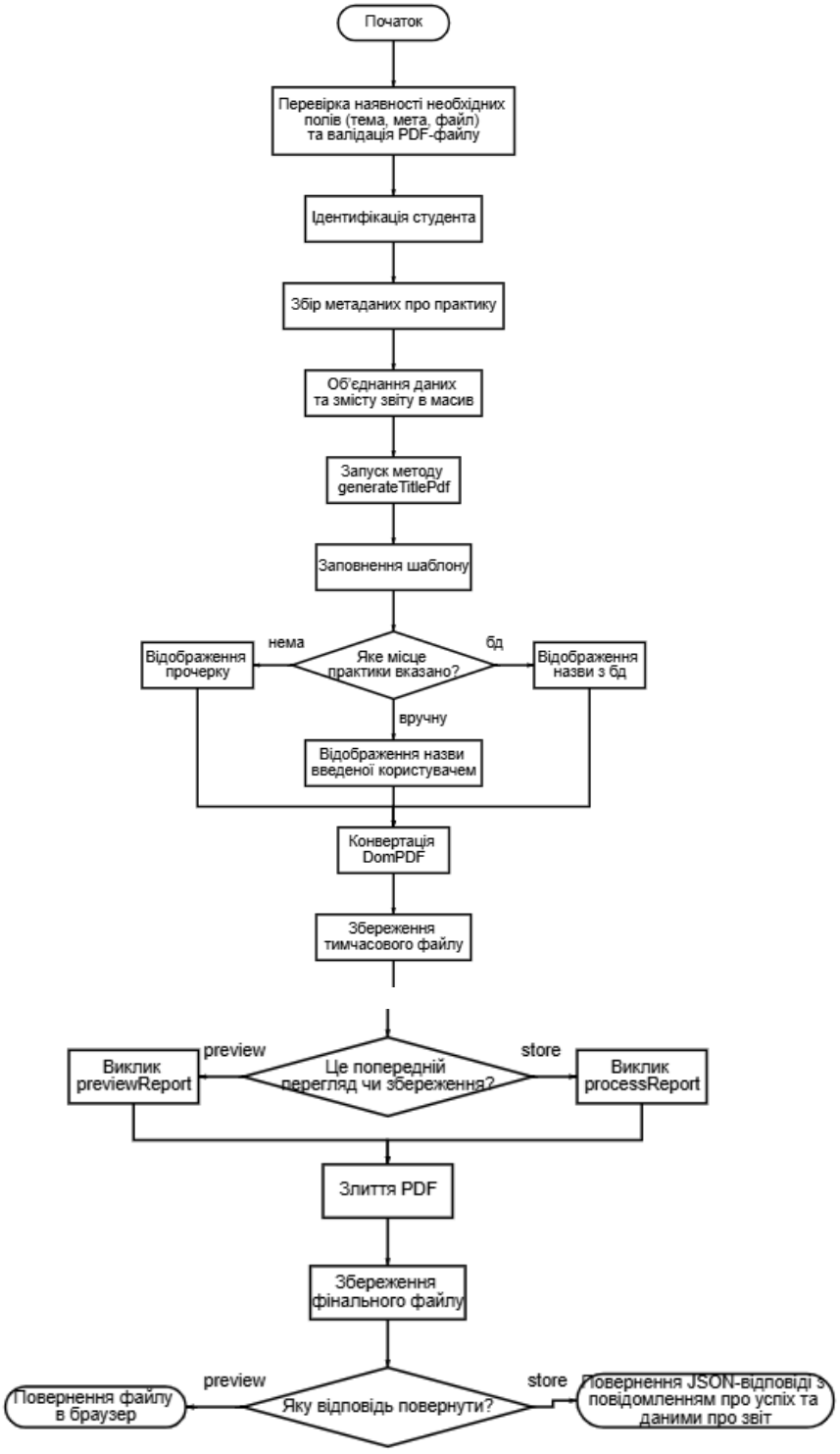


Рисунок 3.12 – Алгоритм генерації документів

3.3.4.2 Функціональність модуля

До функцій модуля генерації документів відносяться:

– автоматичне формування титульного аркуша (заповнення даних про студента, групу та керівників на основі профілю в системі);

– динамічна обробка місця практики (офіційна назва підприємства або введений вручну тексту);

– генерація інтерактивного реферату (автоматичне включення обсягу звіту, апаратних засобів та результатів у текстовий шаблон);

– попередній перегляд (Preview) (можливість перевірити коректність оформлення звіту без його збереження в базі даних).

3.3.4.3 Реалізація та використані технології

Модуль реалізовано за допомогою таких технологій:

– DomPDF: бібліотека для конвертації HTML/CSS у формат PDF, що забезпечує точне позиціонування елементів титульної сторінки;

– FPDI: інструмент для імпорту існуючих PDF-файлів та їх об'єднання зі згенерованими сторінками;

– Blade Templating: двигун шаблонів Laravel для підстановки змінних у HTML-структуру документів.

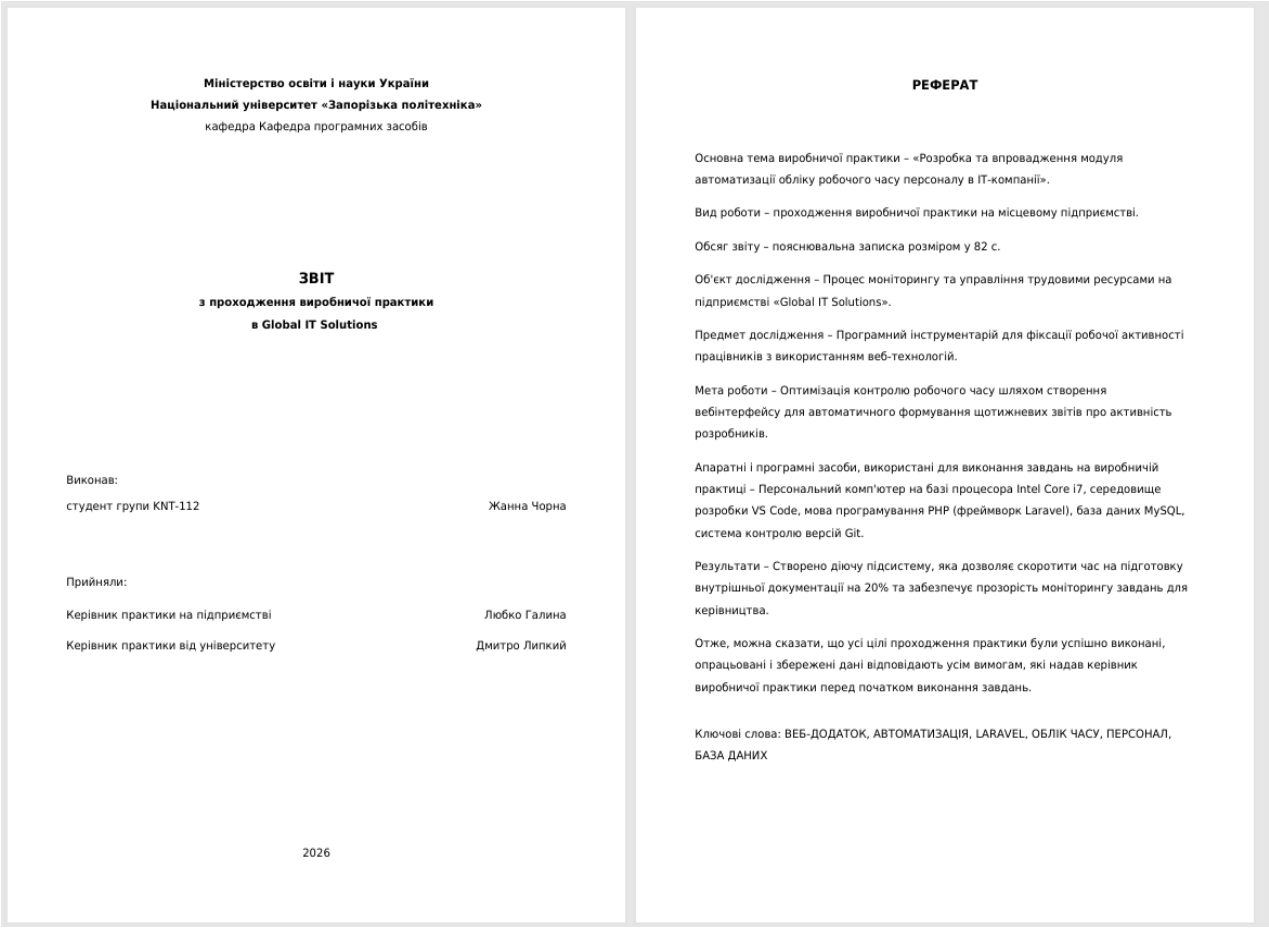
Основні класи та файли модуля:

– ReportController.php: містить методи store та preview, що відповідають за логіку обробки вхідних даних та виклик сервісів генерації;

– DocumentService.php: клас для безпосередньої роботи з PDF-бібліотеками та злиттям файлів;

– report_template.blade.php (HTML-код): шаблон документа, що використовує CSS для формування титульної сторінки та реферату.

Фрагмент логіки генерації титульної сторінки та реферату наведено в Додатку Б.5 та Б.7. Результат автоматичної генерації титульної сторінки та реферату у форматі PDF (рис. 3.13).



Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
кафедра Кафедра програмних засобів

ЗВІТ
з проходження виробничої практики
в Global IT Solutions

Виконав:
студент групи KNT-112 Жанна Чорна

Прийняли:
Керівник практики на підприємстві Любко Галина
Керівник практики від університету Дмитро Липкий

2026

РЕФЕРАТ

Основна тема виробничої практики – «Розробка та впровадження модуля автоматизації обліку робочого часу персоналу в IT-компанії».

Вид роботи – проходження виробничої практики на місцевому підприємстві.

Обсяг звіту – пояснювальна записка розміром у 82 с.

Об'єкт дослідження – Процес моніторингу та управління трудовими ресурсами на підприємстві «Global IT Solutions».

Предмет дослідження – Програмний інструментарій для фіксації робочої активності працівників з використанням веб-технологій.

Мета роботи – Оптимізація контролю робочого часу шляхом створення вебінтерфейсу для автоматичного формування щотижневих звітів про активність розробників.

Апаратні і програмні засоби, використані для виконання завдань на виробничій практиці – Персональний комп'ютер на базі процесора Intel Core i7, середовище розробки VS Code, мова програмування PHP (фреймворк Laravel), база даних MySQL, система контролю версій Git.

Результати – Створено діючу підсистему, яка дозволяє скоротити час на підготовку внутрішньої документації на 20% та забезпечує прозорість моніторингу завдань для керівництва.

Отже, можна сказати, що усі цілі проходження практики були успішно виконані, опрацьовані і збережені дані відповідають усім вимогам, які надав керівник виробничої практики перед початком виконання завдань.

Ключові слова: ВЕБ-ДОДАТОК, АВТОМАТИЗАЦІЯ, LARAVEL, ОБЛІК ЧАСУ, ПЕРСОНАЛ, БАЗА ДАНИХ

Рисунок 3.13 – Результат автоматичної генерації титульної сторінки та реферату у форматі PDF

3.3.5 Модуль комунікації та сповіщень

3.3.5.1 Алгоритм комунікації та сповіщень

Алгоритм роботи модуля забезпечує автоматизоване надсилання сповіщень користувачам через месенджер Telegram при настанні ключових подій у системі. Логіку процесу представлено на рис. 3.14:

Рисунок 3.14 – Алгоритм комунікації та сповіщень

3.3.5.2 Функціональність модуля

До функцій модуля комунікацій та сповіщень відносяться:

– інтеграція з Telegram API (забезпечення технічного зв'язку для передачі даних);

– автоматична розсилка повідомлень (інформування викладачів про нові звіти та студентів про оцінки);

– миттєва передача логінів/паролів (безпечне надсилання облікових даних новим користувачам);

– швидкі коментарі керівника (передача рецензій на звіт безпосередньо в месенджер).

3.3.5.3 Реалізація та використані технології

Модуль реалізовано за допомогою таких технологій:

– Telegram Bot API (метод sendMessage);

– Laravel HTTP Client: для виконання POST-запитів до API.

Основні класи та файли модуля:

– Controller.php (Abstract): містить базовий метод sendTelegramMessage, що є спільним для всіх контролерів системи;

– ReportController.php: ініціює сповіщення при зміні статусу звіту (updateGrade);

– User.php: модель, що містить ідентифікатор зв'язку telegram_chat_id.

Фрагменти коду, що відповідають за роботу з Telegram API, наведено в Додатку Б.8. Приклади сповіщень зображені на рисунку нижче (рис. 3.15).

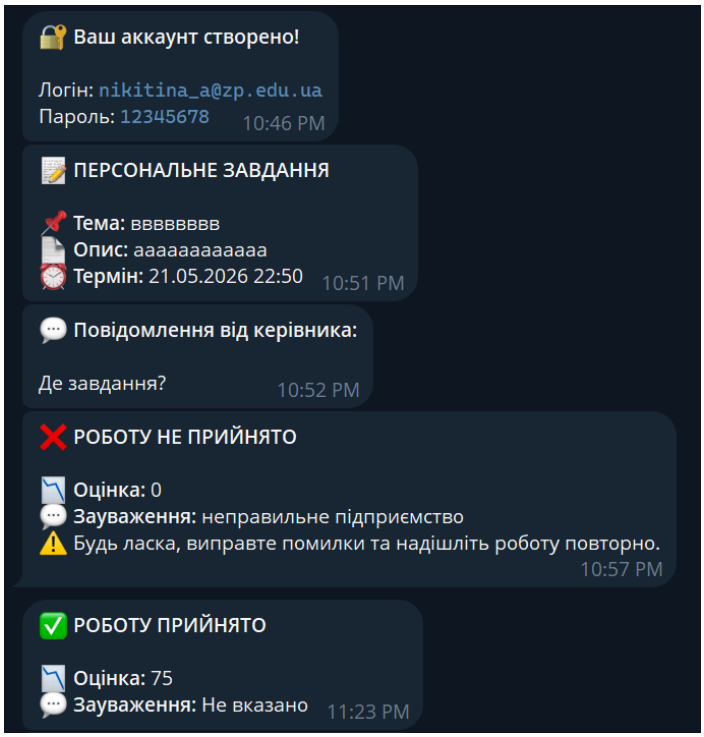


Рисунок 3.15 – Приклади повідомлень студента

3.3.6 Модуль статистики та аналітики

3.3.6.1 Алгоритм отримання статистики та аналітики

Алгоритм роботи модуля забезпечує збір даних про успішність студентів у реальному часі та можливість формування звітів у форматі PDF (рис. 3.16).

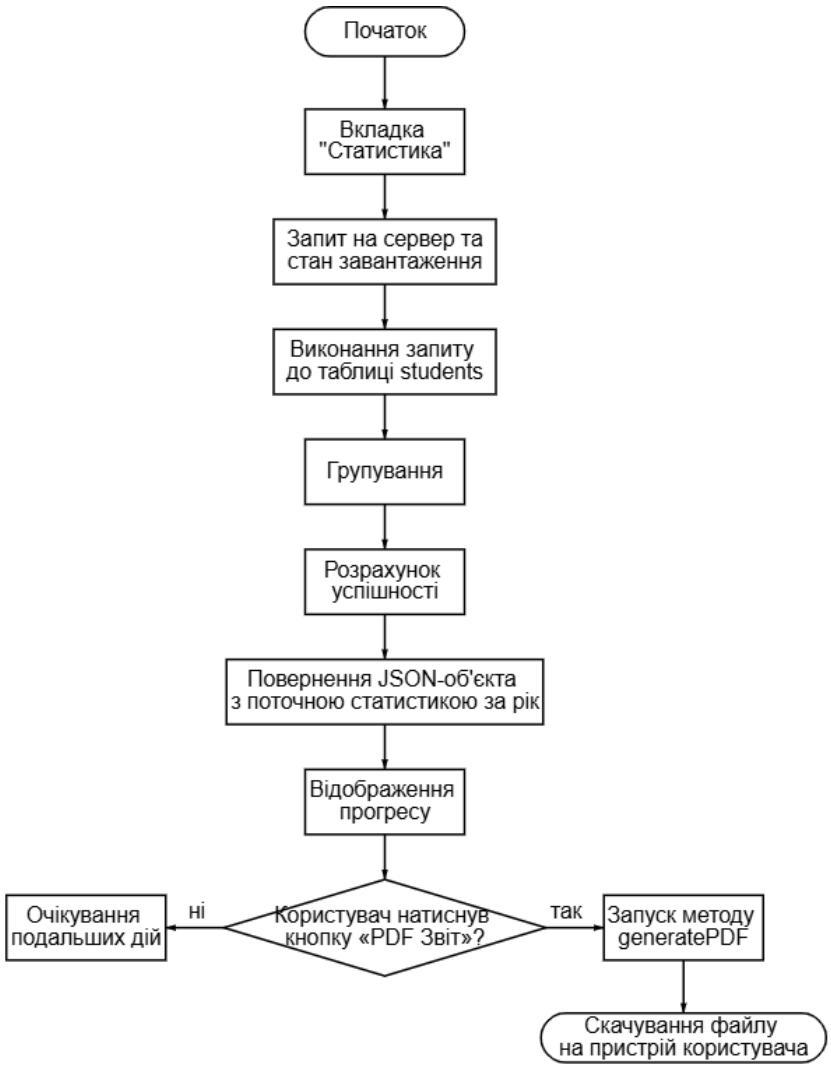


Рисунок 3.16 – Алгоритм отримання статистики та аналітики

3.3.6.2 Функціональність модуля

До функцій модуля статистики та аналітики відносяться:

– клас групування даних за групами (автоматичне сортування студентів та їх результатів за групами);

– клас візуалізації прогресу практики (відображення ключових метрик успішності та середнього балу в інтерфейсі адміністратора);

– клас експорту аналітичних звітів (формування офіційного документа «Department Academic Performance Report»).

3.3.6.3 Реалізація та використані технології

Модуль реалізовано за допомогою таких технологій:

– jsPDF та autoTable: клієнтські бібліотеки для генерації складних PDF-документів з таблицями безпосередньо в браузері;

– Eloquent selectRaw: для виконання складних математичних розрахунків (середнє значення, підрахунок за умовою) на рівні бази даних для підвищення швидкодії;

– Axios та useEffect: для автоматичного оновлення аналітичних даних при кожному відкритті вкладки статистики.

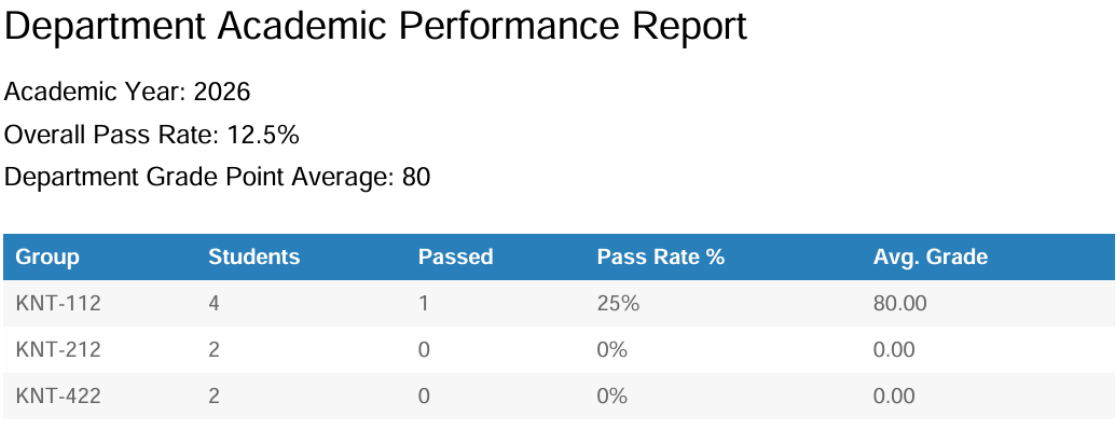
Основні класи та файли модуля:

– AdminDashboard.js (React): містить логіку формування таблиць та метод generatePDF;

– AdminController.php (Laravel): реалізує метод getDetailedStatistics, що готує дані про успішність кафедри;

– Student.php (Model): модель, через яку здійснюється накопичення звітів.

Програмна реалізація методів агрегації даних про успішність кафедр та груп міститься в Додатку Б.3 тв Б.4. Приклад згенерованого звіту статистики зображено на рис. 3.17.



Department Academic Performance Report

Academic Year: 2026

Overall Pass Rate: 12.5%

Department Grade Point Average: 80

Group	Students	Passed	Pass Rate %	Avg. Grade
KNT-112	4	1	25%	80.00
KNT-212	2	0	0%	0.00
KNT-422	2	0	0%	0.00

Рисунок 3.17 – Приклад згенерованого звіту статистики

3.4 Проєктування бази даних

3.4.1 Структура бази даних

База даних вебзастосунку для управління студентськими стажування складається з реляційної БД PostgreSQL, файлового сховища – Laravel File Storage та локального сховища – Browser LocalStorage. PostgreSQL використовує табличну модель для зберігання структурованої інформації для цілісності зв'язків. Laravel File Storage зберігає PDF-документи та шаблони, а Browser LocalStorage підтримує сесії користувачів та зберігає чернетки звітів.

Основними сутностями реляційної БД є: користувачі, ролі, студенти, керівники, звіти, завдання та місця практики. Ці сутності відображають весь функціонал системи, наприклад, управління навчальним процесом, автоматизація створення документів та моніторинг успішності студентів.

Структура бази даних детально описана в таблиці 3.1, яка наводить перелік сутностей, їхні атрибути, типи даних і описи.

Таблиця 3.1 – Опис концептуальної моделі бази даних

№	Сутність	Опис	Атрибут	Тип даних	Опис
1	2	3	4	5	6
1	user	Користувачі	id	bigserial	Унікальний ідентифікатор користувача (PK)
			name	character varying	Повне ім'я (ПІБ) користувача
			email	character varying	Електронна пошта (унікальна)
			password	character varying	Хешований пароль для автентифікації
			role_id	bigint	Зовнішній ключ для зв'язку з роллю
			telegram_chat_id	character varying	ID чату Telegram для сповіщень
			remember_token	character varying	Токен для функції запам’ятовування користувача
			created_at	timestamp	Дата та час створення акаунта
			updated_at	timestamp	Дата та час останнього оновлення даних
2	roles	Ролі	id	bigserial	Унікальний ідентифікатор ролі (PK)
			name	character varying	Назва ролі
			updated_at	timestamp	Дата та час останнього оновлення даних

Продовження таблиці 3.1

1	2	3	4	5	6
2	roles	Ролі	created_at	timestamp	Дата та час створення акаунта
			updated_at	timestamp	Дата та час останнього оновлення даних
3	students	Студенти	id	bigserial	Унікальний ідентифікатор запису студента (PK)
			user_id	bigint	Зовнішній ключ для зв'язку з таблицею users
			group	character varying	Назва академічної групи
			faculty	character varying	Назва факультету
			department	character varying	Назва кафедри
			practice_place_id	bigint	ID офіційного місця практики з бази підприємств
			supervisor_id	bigint	ID викладача-керівника практики
			custom_practice_place_name	character varying	Назва підприємства, введена студентом вручну
			created_at	timestamp	Дата та час створення акаунта
			updated_at	timestamp	Дата та час останнього оновлення даних

Продовження таблиці 3.1

1	2	3	4	5	6
4	supervisors	Викладачі	id	bigserial	Унікальний ідентифікатор керівника (PK)
			user_id	bigint	Зовнішній ключ для зв'язку з таблицею users
			position	character varying	Науковий ступінь або посада
			department	character varying	Назва кафедри, до якої належить викладач
			created_at	timestamp	Дата та час створення акаунта
			updated_at	timestamp	Дата та час останнього оновлення даних
5	practice_ places	Місце практики	id	bigserial	Унікальний ідентифікатор підприємства (PK)
			name	character varying	Офіційна назва компанії або установи
			address	character varying	Адреса підприємства
			contact_person	character varying	ПІБ особи від підприємства
			created_at	timestamp	Дата та час створення акаунта
			updated_at	timestamp	Дата та час останнього оновлення даних
			name	character varying	Офіційна назва компанії або установи
			address	character varying	Адреса підприємства

Продовження таблиці 3.1

1	2	3	4	5	6
5	practice_ places	Місце практики	contact_person	character varying	ПІБ особи від підприємства
			created_at	timestamp	Дата та час створення акаунта
			updated_at	timestamp	Дата та час останнього оновлення даних
6	tasks	Завдання	id	bigserial	Унікальний ідентифікатор завдання (PK)
			supervisor_id	bigint	Зовнішній ключ для зв'язку з викладачем, що створив завдання
			title	character varying	Короткий заголовок завдання
			description	text	Детальний опис завдання
			deadline	timestamp	Кінцевий термін виконання
			created_at	timestamp	Дата та час створення акаунта
			updated_at	timestamp	Дата та час останнього оновлення даних
			created_at	timestamp	Дата та час створення акаунта
			updated_at	timestamp	Дата та час останнього оновлення даних

Продовження таблиці 3.1

1	2	3	4	5	6
7	reports	Звіти	id	bigserial	Унікальний ідентифікатор звіту (PK)
			student_id	bigint	ID студента, який подав звіт
			task_id	bigint	ID завдання, до якого відноситься звіт
			original_file_path	character varying	Шлях до фізичного PDF-файлу у сховищі
			year	integer	Рік проходження практики
			status	enum	Статус перевірки
			grade	integer	Оцінка за звіт
			comment	text	Зауваження викладача
			topic	character varying	Тема звіту
			object	text	Об'єкт дослідження практики
			subject	text	Предмет дослідження
			goal	text	Мета проходження практики
			hardware_software	text	Перелік використаного обладнання та ПЗ
			results	text	Отримані результати та висновки
			keywords	character varying	Ключові слова для автоматичної анотації

Продовження таблиці 3.1

1	2	3	4	5	6
7	reports	Звіти	external_supervisor	character varying	ПІБ керівника від підприємства
			practice_place_id	bigint	Зв'язок з базою практики
			created_at	timestamp	Дата та час створення акаунта
			updated_at	timestamp	Дата та час останнього оновлення даних

3.4.2 Взаємозв'язки між таблицями

Взаємозв'язки між таблицями реалізовано через зовнішні ключі:

- Users – Roles (багато до одного);
- Students – Users (один до одного);
- Supervisors – Users (один до одного);
- Students – Practice_places (багато до одного);
- Students – Supervisors (багато до одного);
- Tasks – Supervisors (багато до одного);
- Reports – Students (багато до одного);
- Reports – Tasks (багато до одного);
- Reports – Practice_places (багато до одного).

Схему взаємозв'язків між таблицями зображено на рисунку 3.18.

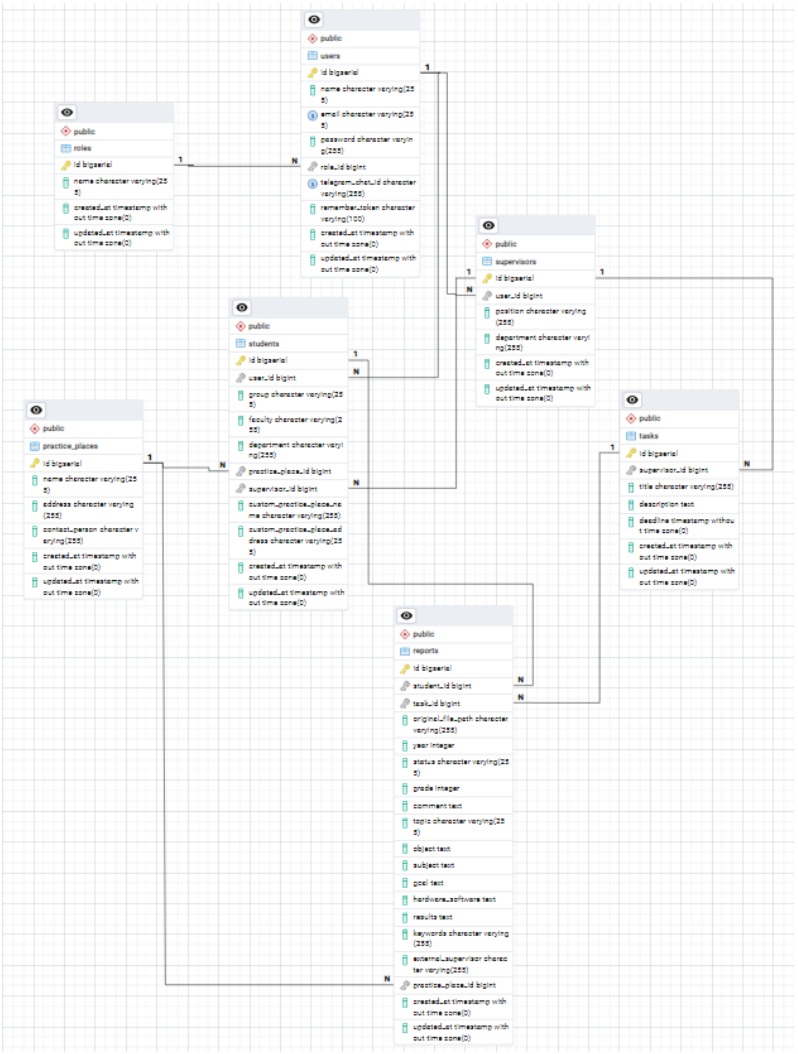


Рисунок 3.18 – Схема взаємозв'язків між таблицями

3.4.3 Індексація та оптимізація

Для забезпечення швидкої роботи вебзастосунку та мінімізації часу на відповідь при роботі з великими масивами даних, в PostgreSQL реалізовано стратегію індексування ключових полів. Оптимізація базується на таких елементах:

– індексація зовнішніх ключів: всі поля, що забезпечують зв'язки між таблицями (наприклад, user_id у таблицях students та supervisors, student_id та task_id у таблиці reports), автоматично індексуються завдяки методу foreignId()->constrained() у міграціях Laravel. Це прискорює операції об’єднання таблиць (JOIN) при формуванні повних профілів користувачів;

– унікальні індекси: для полів email та telegram_chat_id у таблиці users застосовано унікальні індекси, для гарантії цілісності даних та можливості миттєво знаходити адресата при надсиланні сповіщень;

– індексація статусів та дат: поле status в таблиці reports підлягає індексуванню для швидкої фільтрації звітів за станом «pending».

3.5 Висновки за розділом 3

В даному розділі було детально описано процес розробки вебзастосунку для керування студентськими стажуваннями.

Розроблено та реалізовано шість основних модулів системи.

Модуль автентифікації та доступу – для безпечного входу до системи та розмежування прав доступу користувачів відповідно їхнім ролям. Модуль управління користувачами – для адміністрування облікових записів. Модуль роботи зі звітами та документацією – відповідає за завантаження, валідацію, зберігання та перевірку студентських звітів про проходження практики. Модуль генерації документів – автоматизує процес формування титульних сторінок та рефератів у форматі PDF на основі шаблонів та внесених у систему метаданих. Модуль комунікації та сповіщень – реалізує оперативну передачу системних повідомлень та нагадувань користувачам через інтеграцію з Telegram Bot API. Модуль статистики та аналітики – забезпечує візуалізацію прогресу практики та розрахунок показників успішності студентів із можливістю експорту звітів.

Спроєктовано структуру бази даних, яка складається з реляційної БД PostgreSQL, файлового сховища – Laravel File Storage та локального сховища – Browser LocalStorage, які всі взаємодіють між собою. Визначено таблиці, їх поля та взаємозв'язки, а також індекси для оптимізації запитів.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Опис застосування програми

Вебзастосунок для керування студентськими стажуваннями призначений для автоматизації процесів взаємодії між студентами та викладачами під час проходження студентських практик. Програма дає змогу створювати базу підприємств, призначати студентам завдання та стежити за термінами подання звітів. Вона охоплює основні етапи проходження практики: від швидкого отримання студентами індивідуальних завдань, до автоматичного формування рефератів і титульних сторінок на основі введених даних.

Система також дозволяє гнучко обирати місце практики: студент може скористатися вже наявним списком партнерських організацій, або додати нове підприємство вручну. Важливою частиною роботи застосунку є вбудовані сповіщення через Telegram-бота, які забезпечують оперативну комунікацію та нагадують про важливі дедлайни.

4.1.1 Огляд планової цільової аудиторії

Актуальність розробки системи для керування студентськими стажуваннями з автоматизацією звітів з практики підтверджується масштабами сучасної системи вищої освіти України. Згідно з даними Державної служби статистики України, загальна кількість здобувачів вищої освіти за 2025 рік близько мільйона осіб [12]. А через те, що проходження практики є обов'язковим елементом навчального процесу для кожного студента, це створює значне навантаження на адміністративні ресурси та викладацький склад закладів вищої освіти.

4.1.1.1 Розрахунок активної аудиторії

Хоча загальна кількість студентів дорівнює мільйону, проходження практики студентами передбачається переважно на останніх курсах, а саме 3-4 курси бакалаврату та магістратура. Тож, практика становить близько 40-50% від загального складу студентів, тому кількість активних користувачів модуля подання звітів становить приблизно 400 000 – 500 000 осіб щороку.

Саме через це система має забезпечувати обробку близько пів мільйона документів (звітів, щоденників та рефератів) протягом травня–червня. Це потребує архітектури, яка може швидко обробляти великі обсяги даних і забезпечувати стабільний доступ до PDF-файлів.

Великі статистичні показники свідчать про наявність великих ЗВО. У таких закладах автоматизація є необхідною, щоб уникнути перевантаження та проблем із ручною обробкою звітів.

Також, з огляду на загальну аудиторію потенційних користувачів, використання Telegram-бота як модуля сповіщень є найбільш раціональним рішенням. Він забезпечує швидку та стабільну доставку інформації про дедлайни та зміни статусів звітів для великої кількості студентів одночасно, не перевантажуючи сервер застосунку.

4.1.1.2 Розподіл за учасниками навчального процесу

З урахуванням реальної кількості практикантів, цільова аудиторія системи в межах одного навчального року розподіляється так:

– студенти-практиканти (~ 450 000 осіб): Основні користувачі, що формують запити на генерацію документів та завантажують файли звітів у систему;

– керівники практики (викладачі): Забезпечують супровід та перевірку робіт для вищезгаданої кількості студентів. Використання системи дозволяє одному керівнику ефективно адмініструвати великі групи студентів одночасно, що дуже важко при паперовому документообігу;

– адміністрація ВНЗ: Використовує аналітичний модуль для контролю за ходом практики у масштабах всього закладу.

Такий масштабний обсяг цільової аудиторії вимагає від програмного забезпечення високої продуктивності, особливо в періоди пікових навантажень, які зазвичай припадають на кінець семестрів. Впровадження клієнт-серверної архітектури на базі React та Laravel дозволяє ефективно розподіляти обчислювальні ресурси.

4.2 Інструкція по роботі з програмою

4.2.1 Запуск вебзастосунку

Для початку роботи з вебзастосунком для курування студентськими стажуваннями для пристрою користувача є такі вимоги:

– наявність сучасного веббраузера (Google Chrome, Mozilla Firefox, Safari або Microsoft Edge останніх версій);

– встановлений месенджер Telegram для отримання сповіщень;

– підключення до мережі Інтернет;

– підтримка пристроєм перегляду PDF-файлів (для перевірки згенерованих документів).

Для активації модуля комунікації через Telegram-бот (рис. 4.1 – 4.2) користувачу необхідно:

– відкрити Telegram та знайдіть бота за його назвою (@StudentInternshipBot);

– натиснути кнопку «Старт».

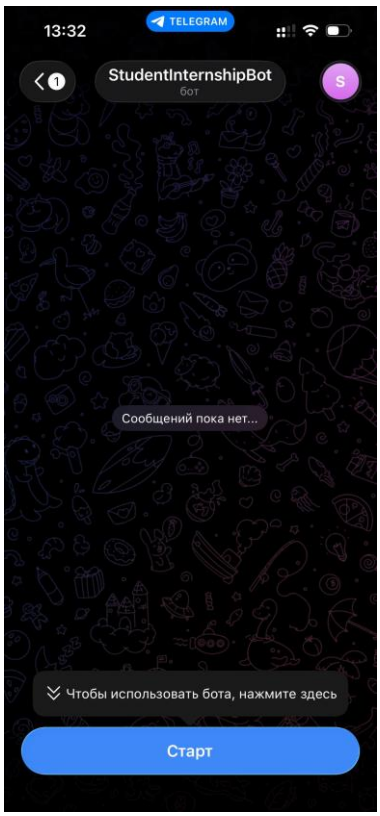


Рисунок 4.1 – Приєднання до Telegram-боту

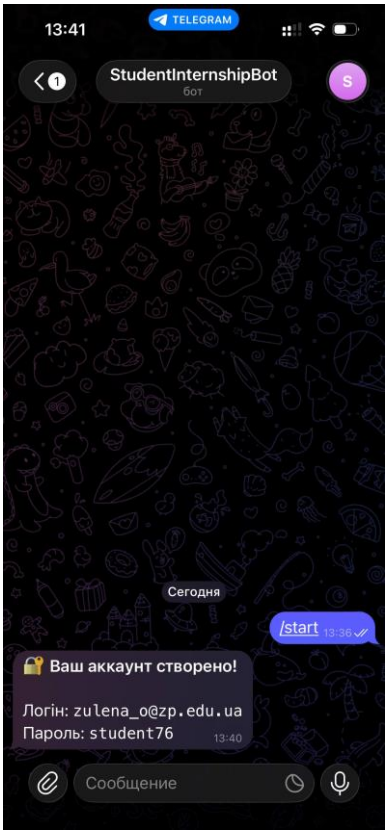


Рисунок 4.2 – Отримання корпоративної електронної пошти та пароля

Для доступу через веб-інтерфейс (рис. 4.3) користувачу необхідно:

– перейти за офіційною URL-адресою системи (https://practice-reports.univ.edu.ua);

– пройти процедуру автентифікації, використовуючи надані адміністратором облікові дані.

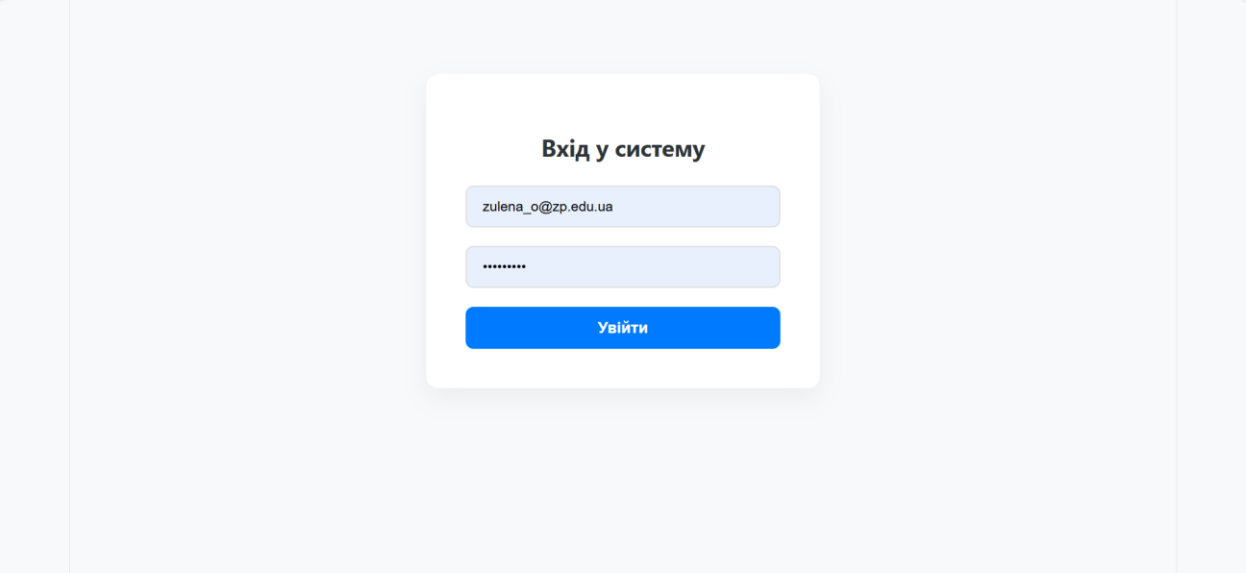


Рисунок 4.3 – Екран сторінки автентифікації куди користувач вводить надані дані

Далі потрібно дозволити необхідні налаштування:

– дозвіл на відкриття спливаючих вікон (для перегляду сформованих звітів у браузері);

– дозвіл на надсилання сповіщень у Telegram для оперативного отримання статусів перевірки.

Схему процесу доступу та активації системи зображено на рисунку 4.4.

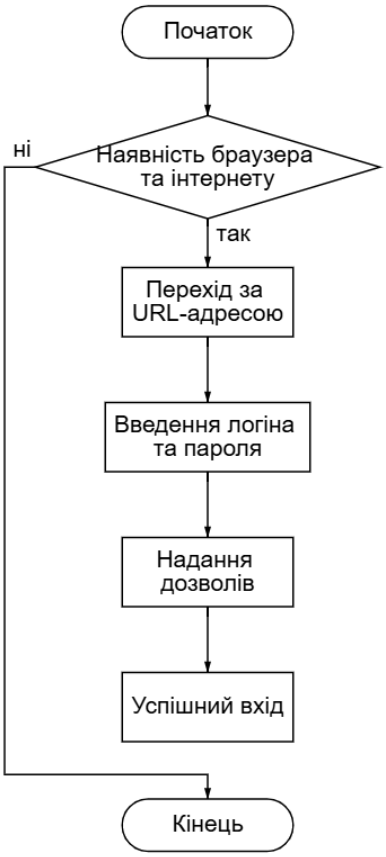


Рисунок 4.4 – Схема процесу відкриття застосунку

4.2.2 Методика роботи з програмою

Основні дії користувача при роботі з програмою (рис. 4.5):

а) робота з індивідуальними завданнями:

1) перегляд призначених керівником завдань та термінів їх виконання;

2) вибір підприємства з наявної бази або внесення даних про нове місце практики;

3) завантаження додаткових матеріалів;

б) формування та подання звітності:

1) заповнення метаданих звіту (тема, об'єкт, мета, використане ПЗ тощо);

2) завантаження фінального PDF-файлу зі звітом;

3) автоматична генерація супровідних документів (рефератів, титульних сторінок) на основі введених даних;

в) моніторинг та отримання результатів:

1) відстеження статусу перевірки звіту («pending», «approved», «rejected»);

2) перегляд оцінок та коментарів викладача;

3) отримання миттєвих сповіщень про результати перевірки через Telegram-бота.

Для ефективної роботи вебзастосунку студентам рекомендується ретельно заповнювати всі текстові поля в модулі подання звіту, оскільки ці дані автоматично підставляються у шаблони документів, що мінімізує помилки при оформленні документів. Викладачам варто залишати коментарі до відхилених звітів, щоб студенти могли вчасно внести правки та дотриматися дедлайнів.

Рисунок 4.5 – Схема основних дій користувача

4.3 Приклад роботи користувача з програмою

Розглянемо типовий сценарій використання програми на прикладі навчального процесу кафедри, де процес розпочинається з організації робочого простору.

Процес розпочинається зі входу адміністратора в систему. Оскільки безпека даних є пріоритетом, адмін створює облікові записи в таблиці (рис. 4.6).

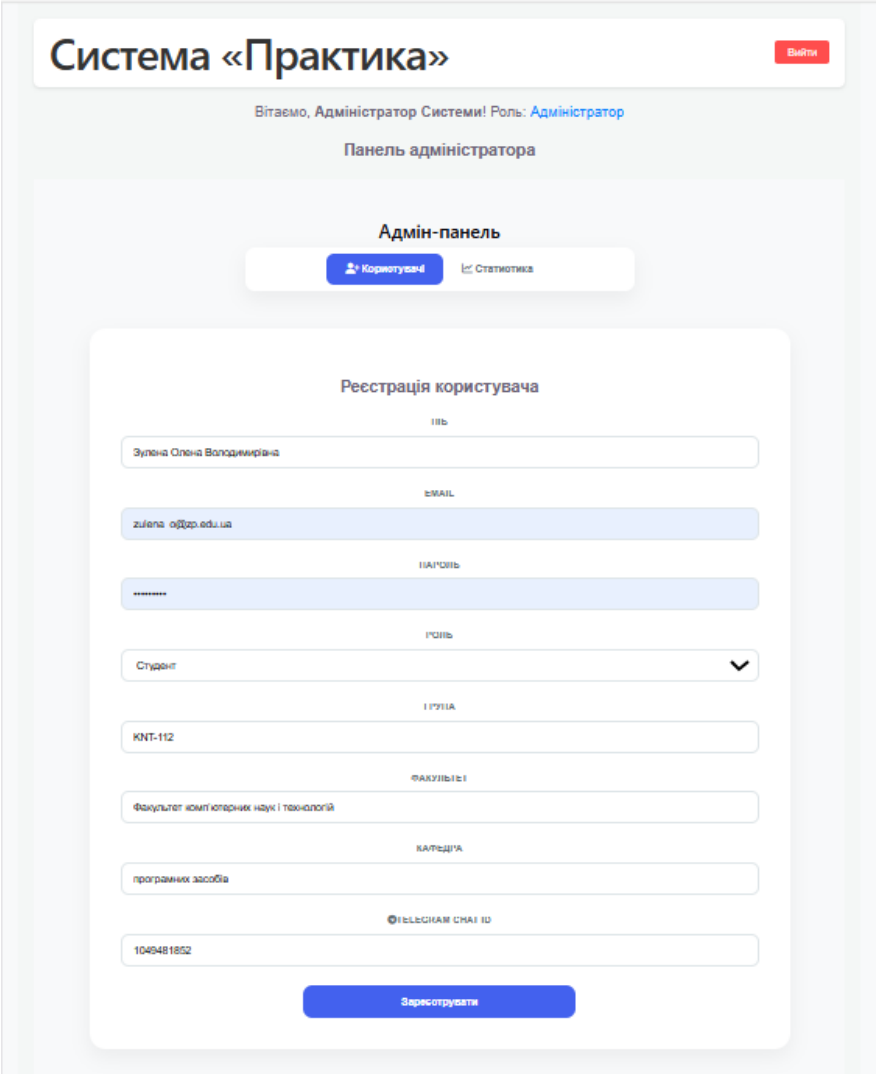


Рисунок 4.6 – Створення облікових записів адміністратором

Після створення акаунтів до роботи стає керівник практики. Його першочерговим завданням є формування своєї навчальної групи. У відповідному модулі він обирає зі списку зареєстрованих студентів тих, чиєю практикою він керуватиме (рис. 4.7).

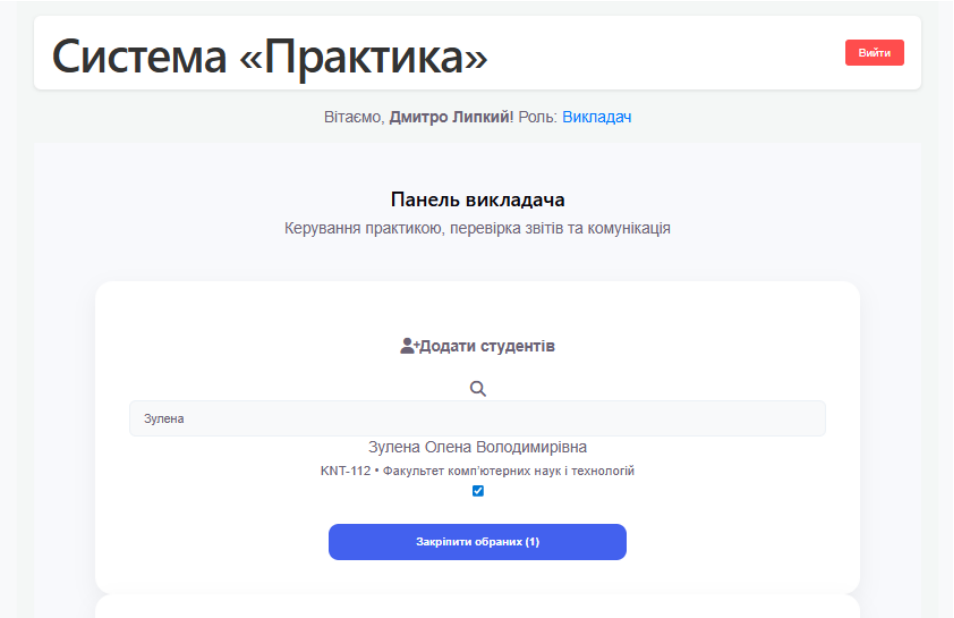


Рисунок 4.7 – Закріплення викладачем студентів

Після цього, він формує перелік індивідуальних завдань для своєї групи із дедлайнами та описом вимог (рис. 4.8), після чого, вони автоматично стають доступними закріпленим за ним студентам (рис. 4.9).

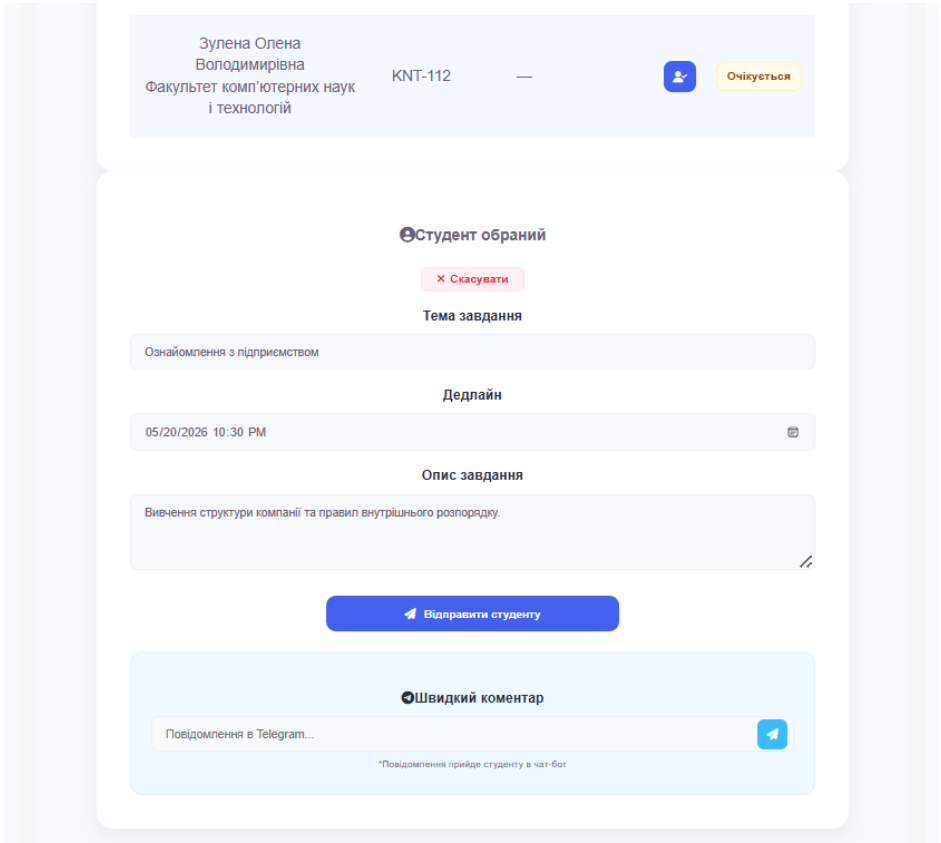


Рисунок 4.8 – Створення викладачем індивідуального завдання для студента

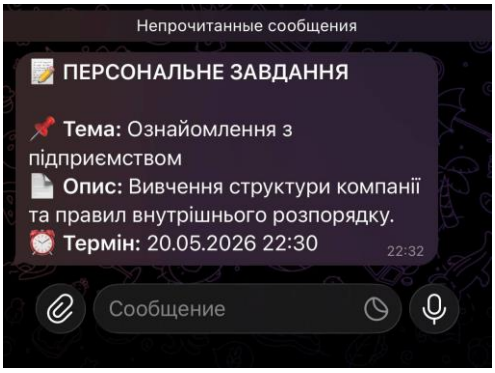


Рисунок 4.9 – Сповіщення для студента після створення викладачем завдання

Далі відбувається етап активної роботи студента. Після входу в особистий кабінет, студент завантажує шаблон пояснювальної записки (рис. 4.10) та працює з нею після чого завантажує PDF 1.4 або нижче, а далі обирає базу практики з представлених (рис. 4.10) або вносить дані про нове підприємство (рис. 4.11).

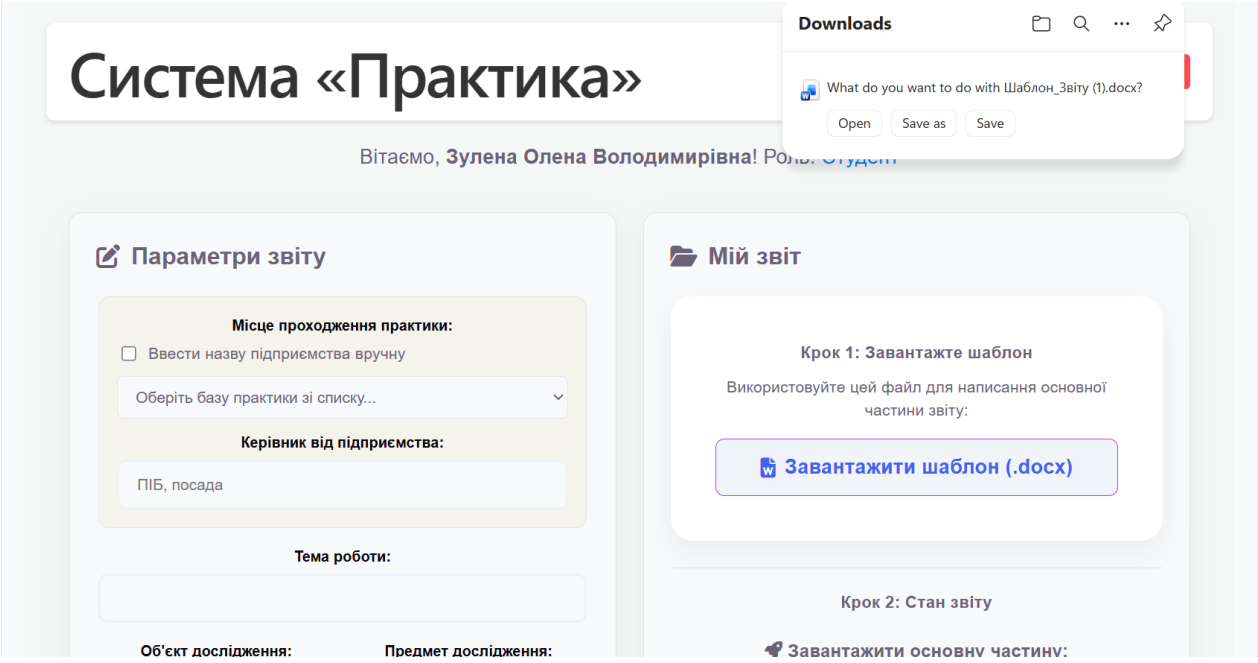


Рисунок 4.10 – Завантаження студентом шаблону пз та вибір бази практики із запропонованих

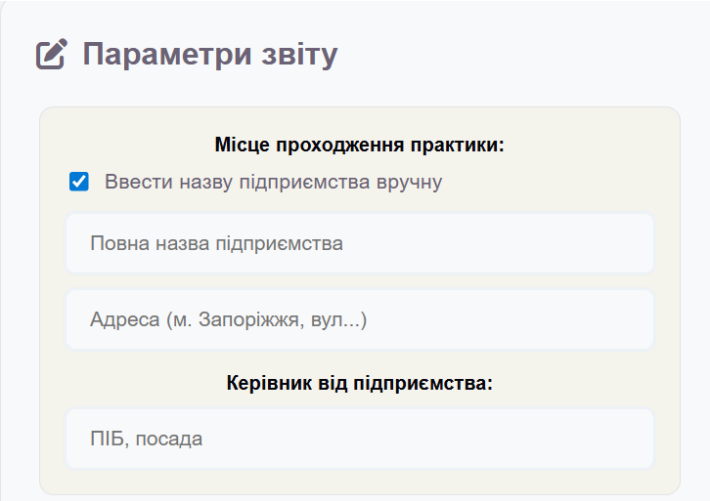


Рисунок 4.11 – Введення студентом бази практики

У процесі виконання завдання студент заповнює метадані майбутнього звіту та зберігає цю чернетку (рис. 4.12).

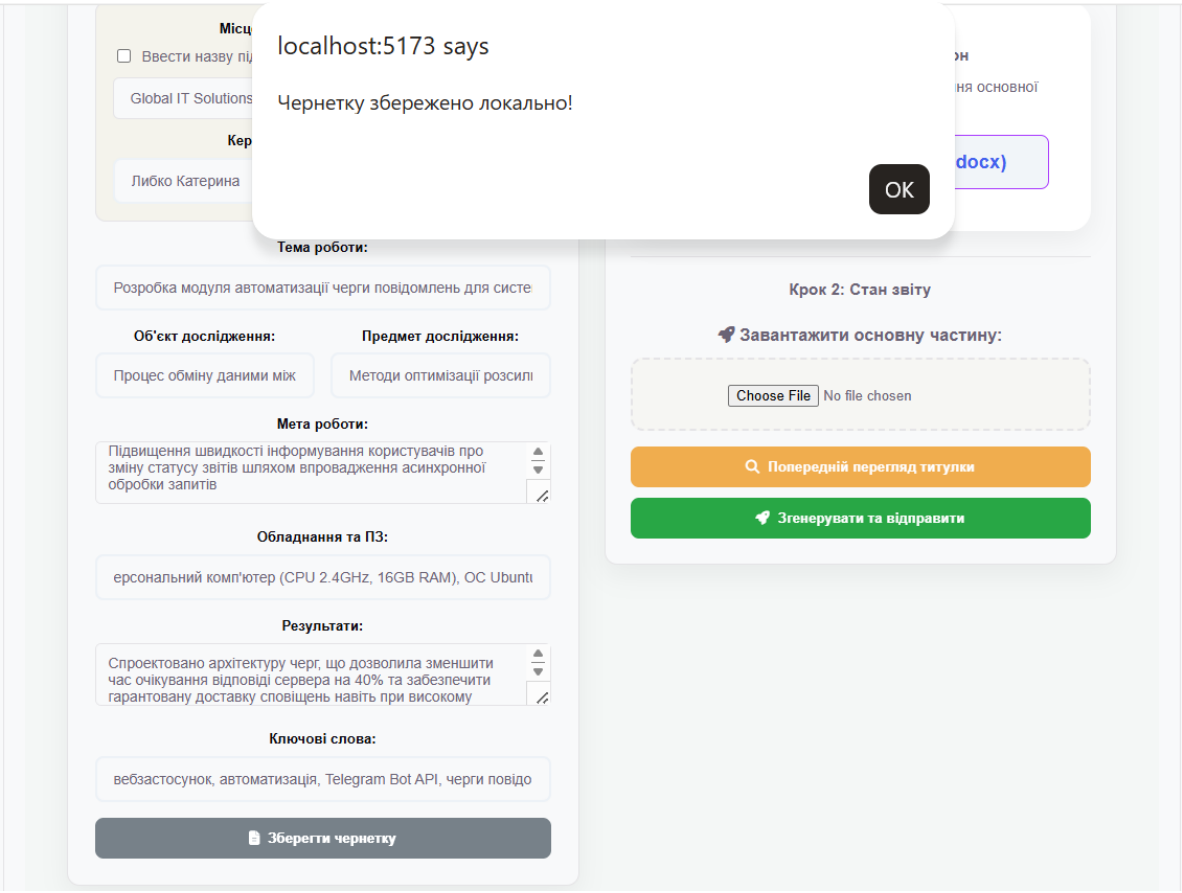
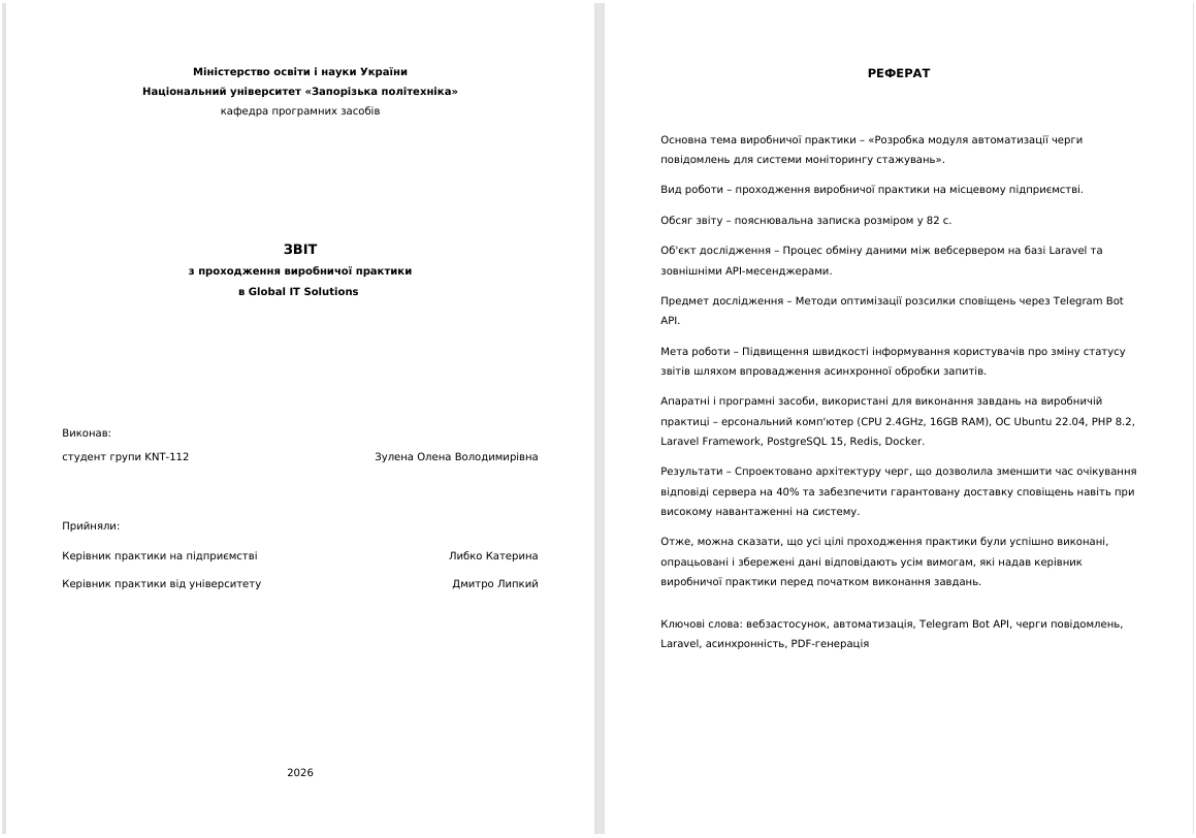


Рисунок 4.12 – Заповнення та зберігання методаних в чернетці студентом

На основі введених даних система дозволяє автоматично згенерувати титульну сторінку та реферат у форматі PDF після натискання кнопки для попереднього перегляду згенерованого документа (рис. 4.13). Після чого, якщо студента все влаштовує він відправляє цей документ викладачу на оцінювання (рис. 4.14).



Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
кафедра програмних засобів

ЗВІТ
з проходження виробничої практики
в Global IT Solutions

Виконав:
студент групи КNT-112 Зулена Олена Володимирівна

Прийняли:
Керівник практики на підприємстві Либко Катерина
Керівник практики від університету Дмитро Липкий

2026

РЕФЕРАТ

Основна тема виробничої практики – «Розробка модуля автоматизації черги повідомлень для системи моніторингу стажувань».

Вид роботи – проходження виробничої практики на місцевому підприємстві.

Обсяг звіту – пояснювальна записка розміром у 82 с.

Об'єкт дослідження – Процес обміну даними між вебсервером на базі Laravel та зовнішніми API-месенджерами.

Предмет дослідження – Методи оптимізації розсилки сповіщень через Telegram Bot API.

Мета роботи – Підвищення швидкості інформування користувачів про зміну статусу звітів шляхом впровадження асинхронної обробки запитів.

Апаратні і програмні засоби, використані для виконання завдань на виробничій практиці – ерсональний комп'ютер (CPU 2.4GHz, 16GB RAM), ОС Ubuntu 22.04, PHP 8.2, Laravel Framework, PostgreSQL 15, Redis, Docker.

Результати – Спроектовано архітектуру черг, що дозволила зменшити час очікування відповіді сервера на 40% та забезпечити гарантовану доставку сповіщень навіть при високому навантаженні на систему.

Отже, можна сказати, що усі цілі проходження практики були успішно виконані, опрацьовані і збережені дані відповідають усім вимогам, які надав керівник виробничої практики перед початком виконання завдань.

Ключові слова: вебзастосунок, автоматизація, Telegram Bot API, черги повідомлень, Laravel, асинхронність, PDF-генерація

Рисунок 4.13 – Попередній перегляд студентом згенерованого документа

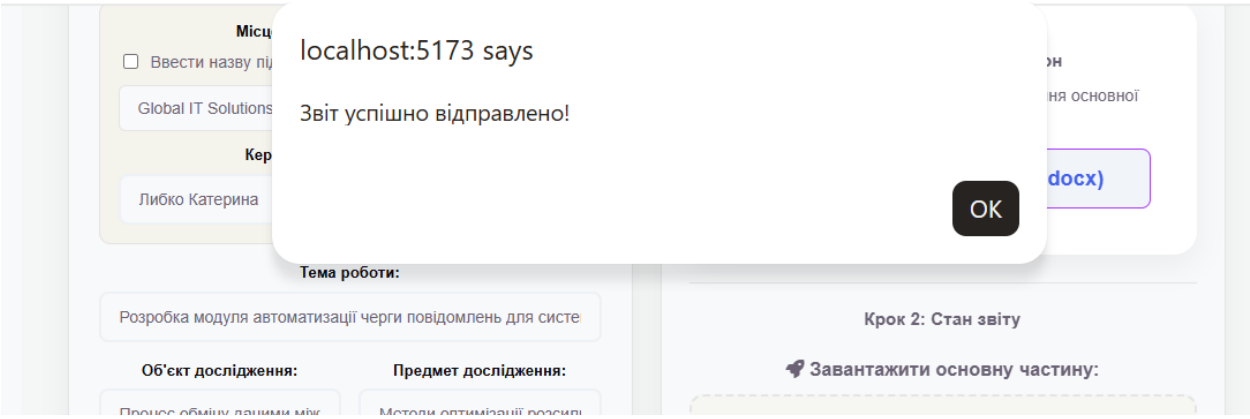


Рисунок 4.14 – Відправка студентом згенерованого звіту викладачу

Після завантаження, керівник отримує повідомлення (рис. 4.15) та перевіряє роботу в інтерфейсі викладача.

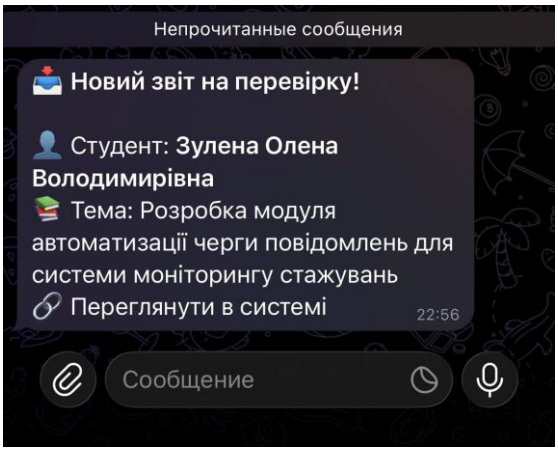


Рисунок 4.15 – Повідомленні керівника після завантаження студентом роботи

Він має можливість змінити статус звіту на «схвалено» із виставленням оцінки (рис. 4.16) або «відхилено» із додаванням коментаря щодо необхідних правок (рис. 4.17).

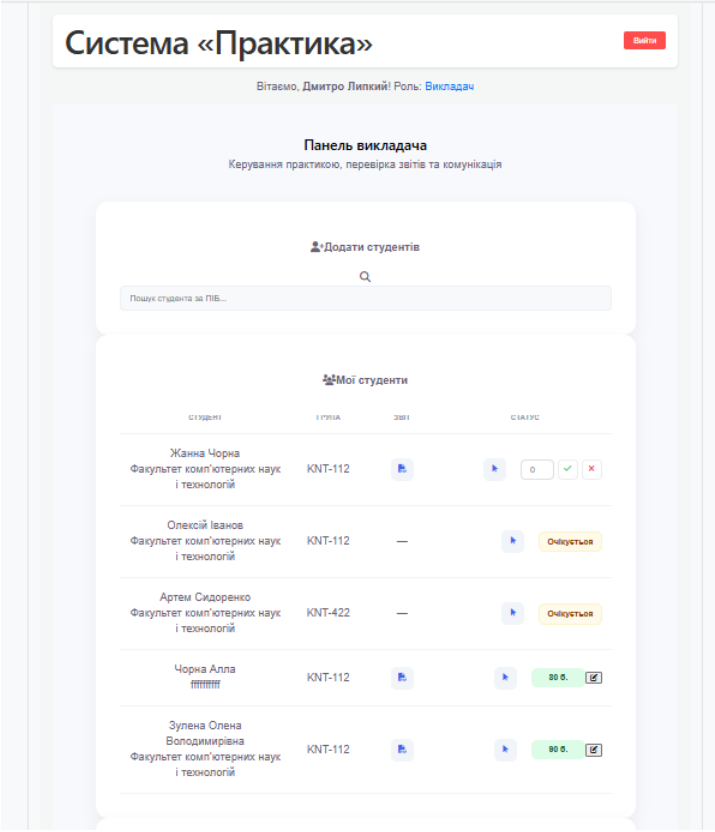


Рисунок 4.16 – Прийняття роботи студента викладачем

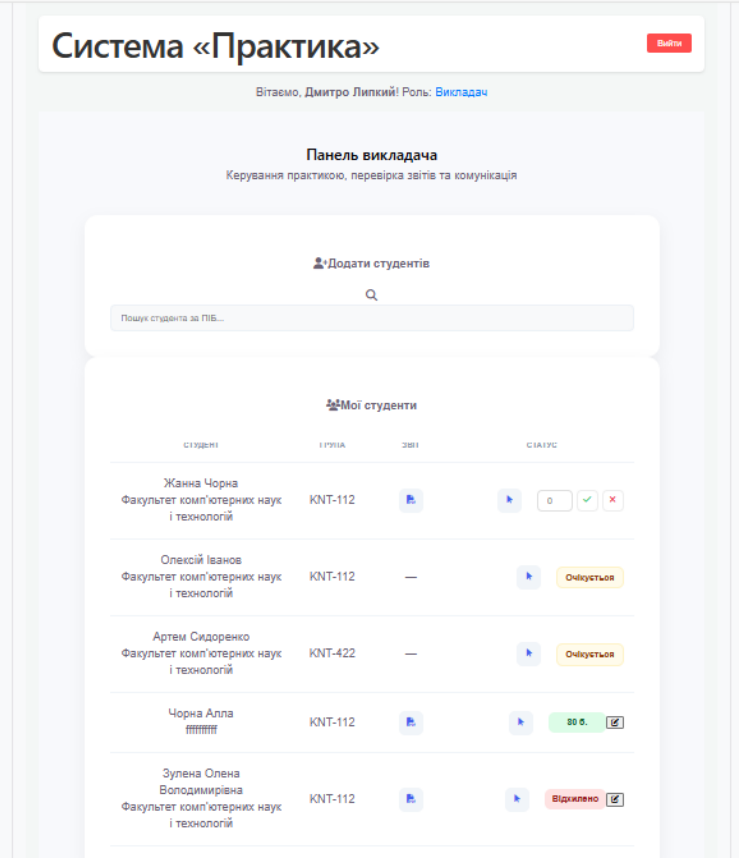


Рисунок 4.17 – Відхилення роботи студента викладачем

Весь процес супроводжується миттєвими сповіщеннями через Telegram-бота, що дозволяє студенту швидко дізнаватися про результат перевірки, не перебуваючи на сайті (рис. 4.18).

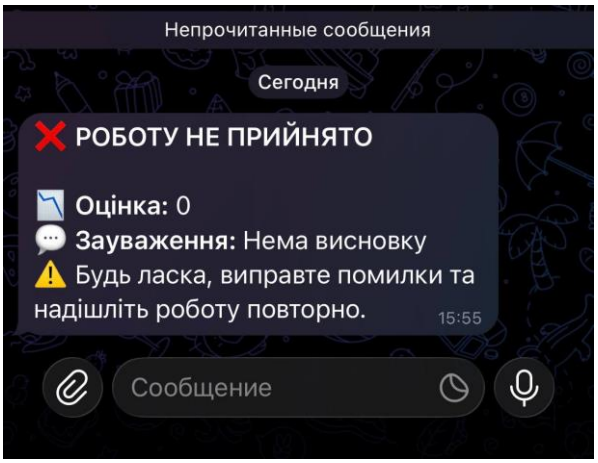


Рисунок 4.18 – Сповіщення студента щодо результату перевірки роботи

Якщо роботу студента схвалено, то в своєму кабінеті в нього будуть заблоковані всі поля для вводу даних, лише буде видно оцінку та можливість завантажити його звіт (рис. 4.19).

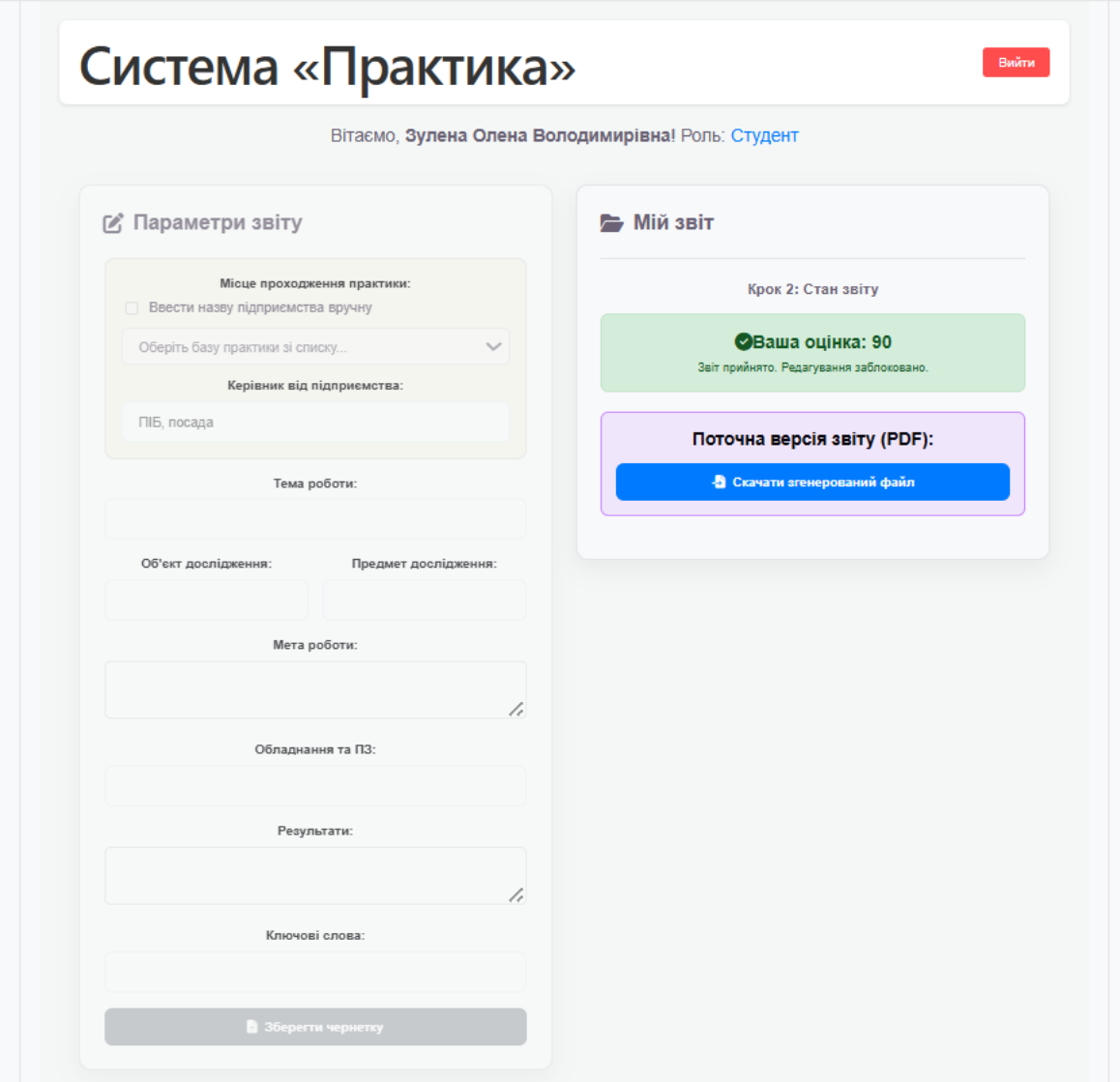


Рисунок 4.19 – Кабінет студента після прийняття викладачем звіта

А якщо ж звіт було відхилено, то студент побачить червоне повідомлення про те що звіт відхилили з коментарем викладача та можливістю переробити звіт (рис. 4.20).

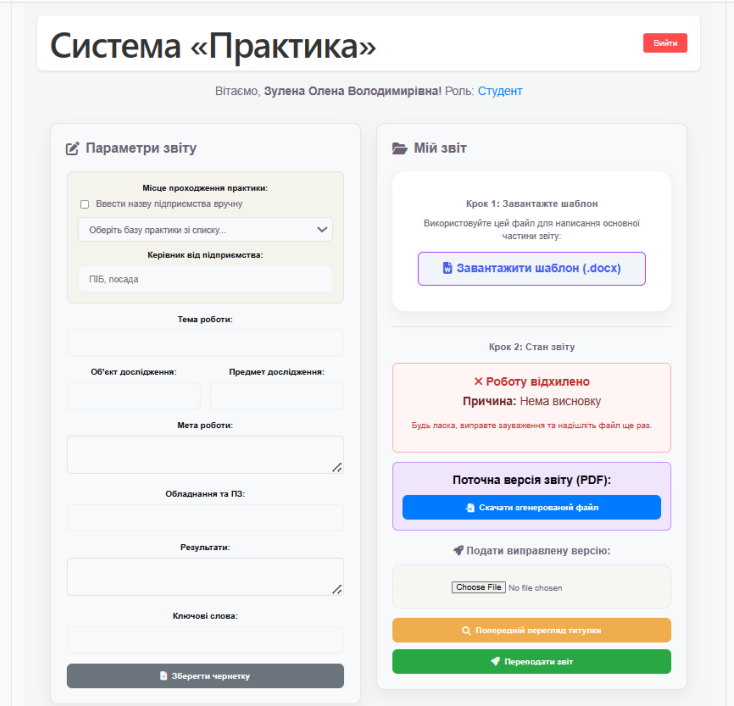


Рисунок 4.20 – Кабінет студента після відхилення викладачем звіта

Викладач, в свою чергу, зможить надсилати окремим студентам повідомлення, наприклад, щоб дати додаткові інструкції або нагадати що скоро підійде час здачі робіт (рис. 4.21).

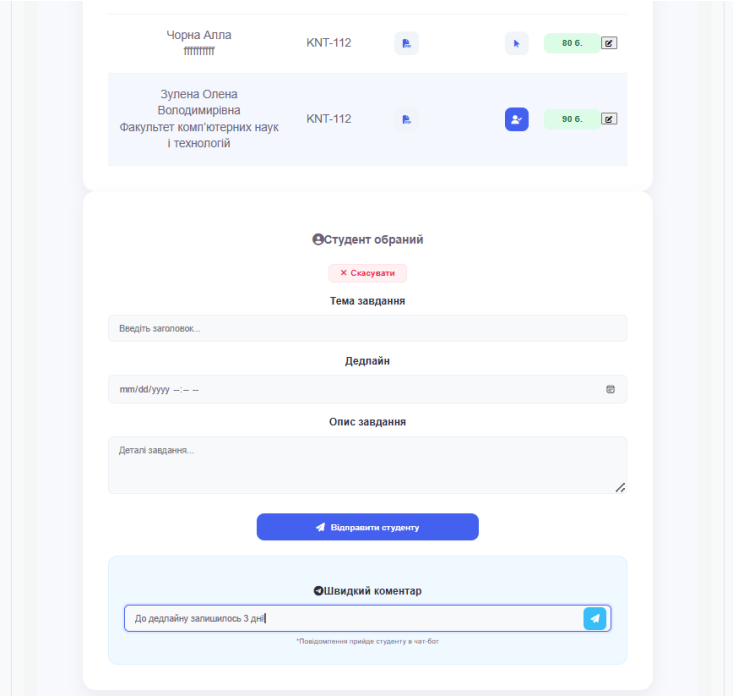


Рисунок 4.21 – Можливість викладача надсилати особисті повідомлення студентам в кабінеті викладача

Адміністратор також зможе у вкладці «Статистика» попередньо переглянути успіхи кожної з груп (рис. 4.22), а також матиме можливість сформувати PDF звіт з статистикою кафедри (рис. 4.23).

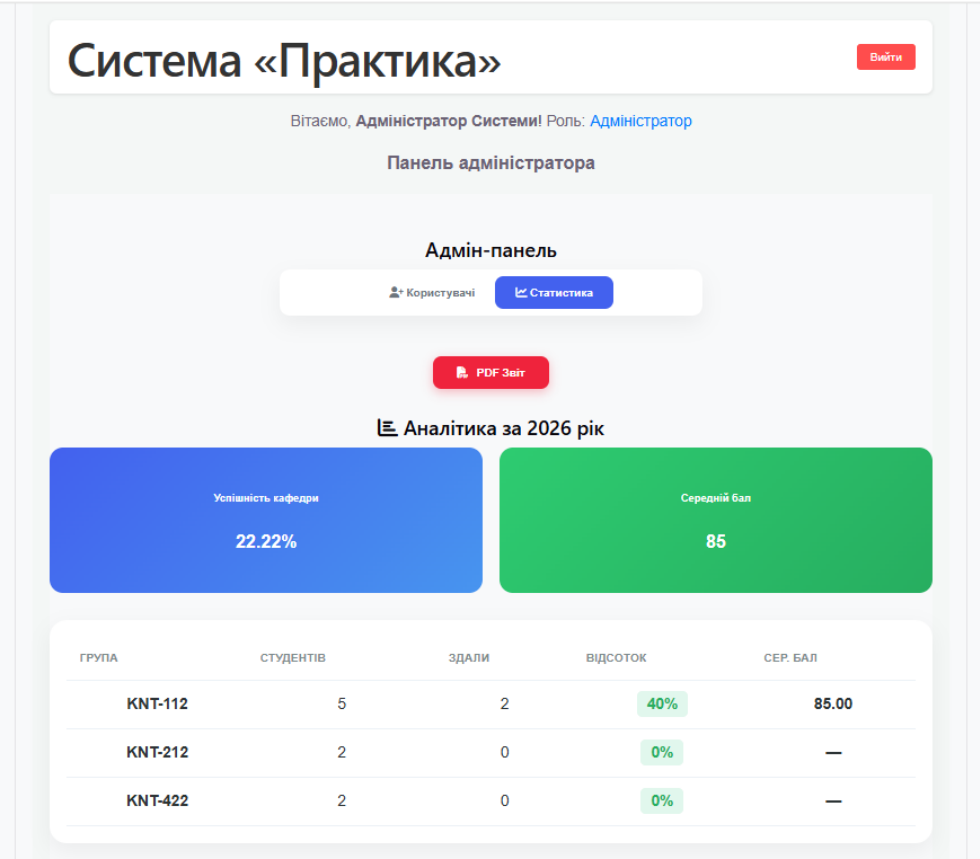
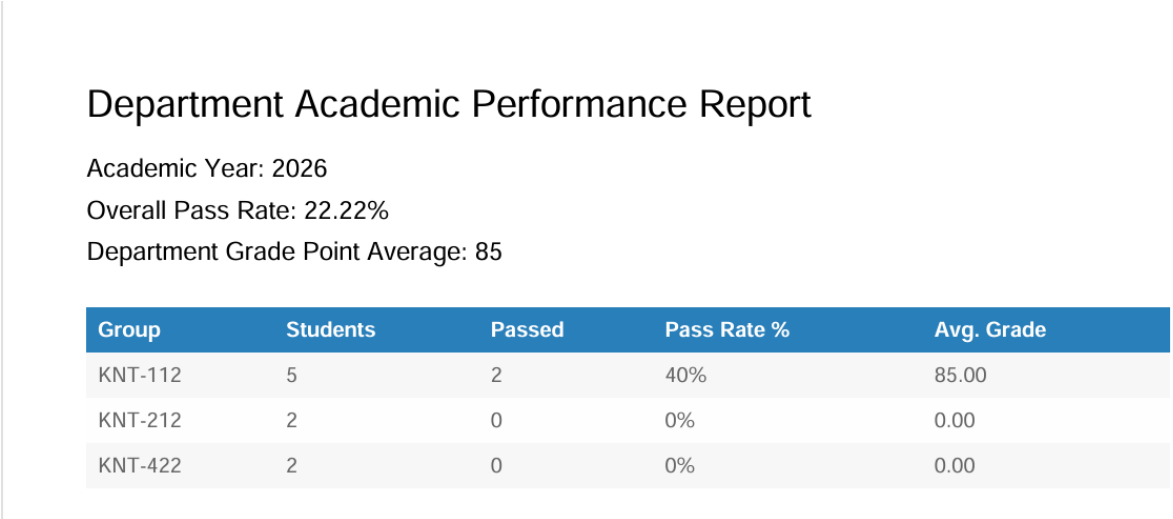


Рисунок 4.22 – Кабінет адміністратора вкладка статистики



Department Academic Performance Report

Academic Year: 2026

Overall Pass Rate: 22.22%

Department Grade Point Average: 85

Group	Students	Passed	Pass Rate %	Avg. Grade
KNT-112	5	2	40%	85.00
KNT-212	2	0	0%	0.00
KNT-422	2	0	0%	0.00

Рисунок 4.23 – Згенерований викладачем PDF файл статистики кафедри

4.4 Тестування роботи програми

4.4.1 Функціональне тестування

Методи тестування:

а) модульне тестування (Unit Testing):

1) тестування окремих компонентів (контролерів та моделей);
2) перевірка бізнес-логіки (обробка статусів та прав доступу);
3) тестування сервісів (генерація документів та робота з чергами);

б) інтеграційне тестування:

1) взаємодія між фронтендом та бекендом;
2) цілісність даних при збереженні у базі даних PostgreSQL;
3) робота із зовнішнім Telegram Bot API;

в) UI тестування:

1) перевірка адаптивності графічного інтерфейсу;
2) тестування навігаційних маршрутів користувача;
3) перевірка динамічного відображення даних.

Результати функціонального тестування наведені в таблиці 4.1.

Таблиця 4.1 – Результати функціонального тестування

Функціонал	Тест-кейси	Результат	Виправлені помилки
Автентифікація	6	Успішно	1
Управління ролями	4	Успішно	2
Робота зі звітами	10	Успішно	3
Генерація PDF	7	Успішно	2
Telegram-сповіщення	5	Успішно	1
Модуль статистики	3	Успішно	1

4.4.2 Аналіз функціонального тестування

Детальний аналіз результатів функціонального тестування показав високу надійність основних компонентів системи та коректність роботи інтеграцій:

Результати модульного тестування (рис. 4.24):

– загальна кількість тестів: 38;
– успішно пройдено: 32 (~90%);
– виявлено помилок: 15;
– виправлено помилок: 11.

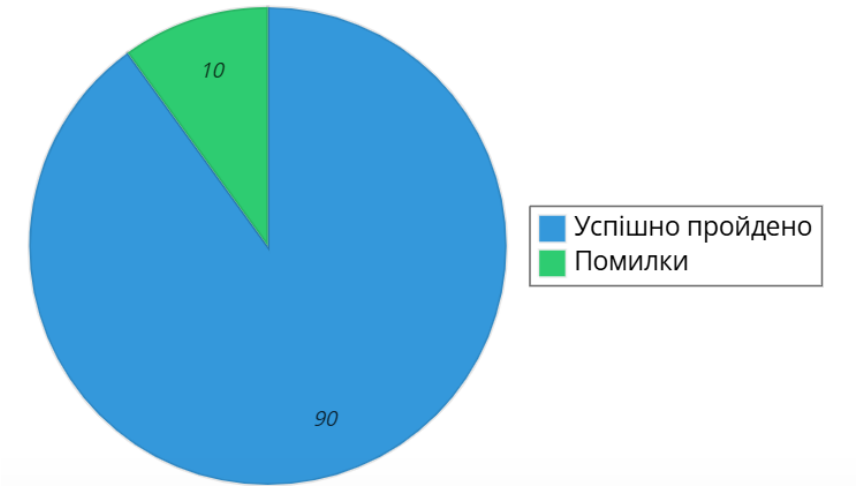


Рисунок 4.24 – Статистика модульного тестування

Основні виправлені помилки:

– помилка виведення даних у PDF;
– збій у завантаженні даних;
– логічна помилка сповіщень;
– валідація файлів (некоректна обробка при об'єднанні PDF-файлів).

Аналіз ключових виявлених помилок наведено в таблиці 4.2.

Таблиця 4.2 – Аналіз виявлених помилок

Тип помилки	Пріоритет	Час виявлення	Час виправлення	Статус
Формування PDF	Високий	15.03.2026	15.03.2026	Виправлено
Зв'язки БД	Високий	27.03.2026	29.03.2026	Виправлено
Сповіщення Telegram	Середній	08.04.2026	10.04.2026	Виправлено
Валідація файлів	Високий	16.04.2025	23.04.2026	Виправлено

4.4.3 Тестування продуктивності

Методи тестування продуктивності:

а) навантажувальне тестування:

1) перевірка працездатності системи при одночасному доступі декількох користувачів;

2) аналіз затримки між зміною статусу звіту в системі та отриманням сповіщення через Telegram Bot API;

3) завантаження та об'єднання багатосторінкових PDF-документів (понад 20 сторінок);

б) моніторинг системних ресурсів:

1) контроль споживання оперативної пам'яті;

2) оцінка ефективності обробки JavaScript-коду на стороні клієнта при відображенні великих списків звітів.

Результати використання оперативної пам'яті при генерації PDF представлено на рисунку 4.25.

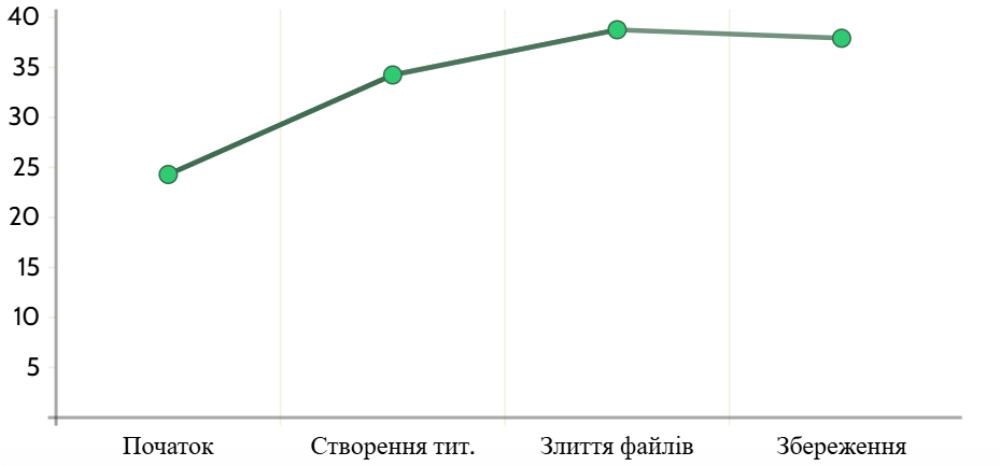


Рисунок 4.25 – Графік використання пам'яті під час генерації PDF

4.4.4 Аналіз продуктивності

Тестування продуктивності для оцінки швидкості рендерингу та реакції інтерфейсу проводилось у різних браузерах (рис. 4.26).

Час повного завантаження інтерфейсу (LCP):

– Google Chrome: 0.66 секунди;
– Mozilla Firefox: 0.80 секунди;
– Microsoft Edge: 0.46 секунди.

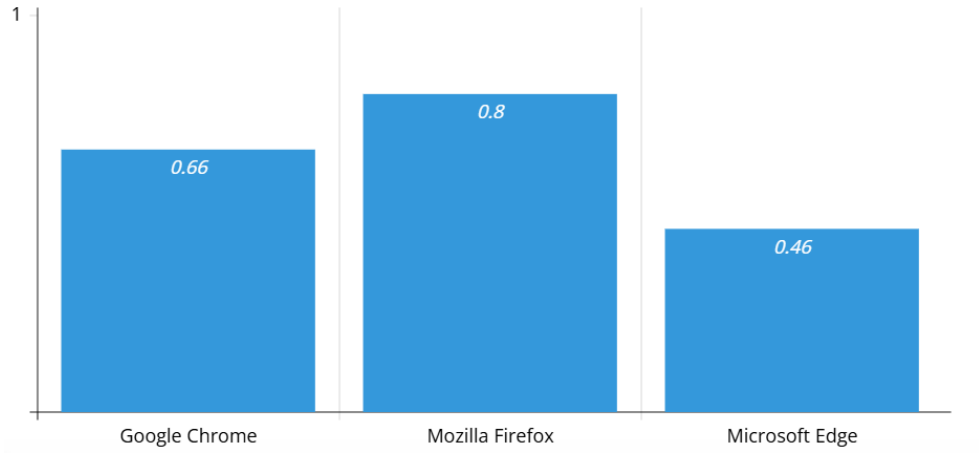


Рисунок 4.26 – Графік часу завантаження сторінки у різних браузерах

Використання системних ресурсів пристрою користувача при роботі з інтерфейсом наведено в таблиці 4.3.

Таблиця 4.3 – Порівняльний аналіз використання ресурсів під час генерації звіту

Браузер	Пам'ять (RAM)	Час виконання (POST)	Швидкість JS / INP
Microsoft Edge	12.4 MB	3.52 с	48 ms
Google Chrome	15.2 MB	3.67 с	56 ms
Mozilla Firefox	8.86 MB	3.80 с	14.5 ms (JS)

4.4.5 Тестування безпеки

Методи тестування безпеки:

а) тестування механізмів автентифікації та сесій:

1) валідація сесійних токенів;
2) захист від brute-force атак;
3) безпечне зберігання сесій;

б) тестування криптографічного захисту та цілісності:

1) хешування паролів;
2) захист від CSRF;

в) тестування контролю доступу:

1) розмежування прав доступу;
2) тестування ролей користувачів.

Результати тестування безпеки підсумовано в таблиці 4.4.

Таблиця 4.4 – Результати тестування безпеки

Категорія тесту	Кількість тестів	Виявлені недоліки	Усунені недоліки
Автентифікація та токени	8	2	2
Криптографічний захист	2	0	0
Права доступу та ролі	6	1	1

4.4.6 Аналіз безпеки

Було проведено комплексний аналіз безпеки розробленого вебзастосунку. Перевірено стійкість алгоритму хешування BCrypt, який Laravel використовує для збереження паролів. І підтверджено, що навіть у разі прямого доступу до БД, автентифікаційні дані залишаються захищеними. Далі було проведено тестування життєвого циклу токенів доступу: при виході із системи токен анулюється на сервері, що запобігає повторному використанню сесії. Також було перевірено захищеність запитів до БД. Завдяки використанню Eloquent ORM, усі вхідні дані автоматично екрануються, що виключає можливість SQL-ін'єкцій через форми введення даних.

Результати сканування безпеки та перевірки стійкості системи наведено в таблиці 4.5.

Таблиця 4.5 – Результати сканування безпеки

Метод перевірки	Знайдено вразливостей	Виправлено
Статичний аналіз	1	1
Динамічний аналіз	1	1
Тестування на проникнення	1	1

4.4.7 Аналіз користувацького досвіду

Було проведено опитування користувачів (від авторизації до отримання оцінки за звіт). Результати оцінки інтерфейсу (рис. 4.27) показали високий рівень зручності:

- дуже задоволені: 25%;
- задоволені: 50%;
- нейтрально: 25%.

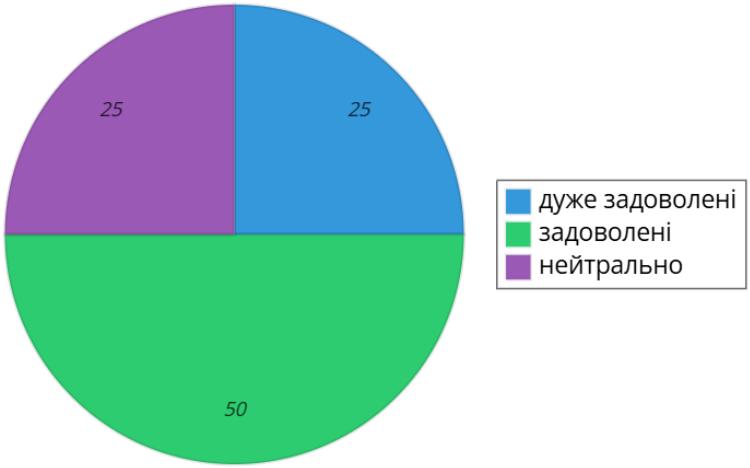


Рисунок 4.27 – Результати оцінювання користувацького інтерфейсу

Найбільш використовувані функції та оцінка їхньої реалізації наведені в таблиці 4.6.

Таблиця 4.6 – Популярність та оцінка функціональних можливостей

Функція	Частота використання	Оцінка користувачів
Генерація та завантаження звіту	98%	9.8/10
Перегляд статистики та оцінок	75%	8/10
Telegram-сповіщення про статус	40%	9.5/10
Пошук місць практики	60%	9/10

4.5 Рекомендації щодо використання

На основі проведеного тестування та аналізу відгуків сформовано ключові рекомендації для ефективної роботи із застосунком:

– перед поданням звіту необхідно використати функцію Preview, щоб перевірити коректність заповнення полів у згенерованому PDF-файлі;

– для миттєвого отримання результатів перевірки та оцінок рекомендується прив'язати обліковий запис до бота Telegram;

– необхідно завантажувати файли виключно у форматі PDF розміром до 10 МБ;

– при заповненні даних про практику обов'язково треба обрати підприємство зі списку «Місця практики» або ввести вручну для автоматичного заповнення адрес та реквізитів;

– після закінчення сесії необхідно завжди використовувати кнопку Logout, щоб анулювати токен доступу та запобігти несанкціонованому використанню профіля.

4.6 Плани з розвитку

Подальший розвиток вебзастосунку передбачає удосконалення роботи Telegram-бота. У майбутніх версіях планується додати розумні сповіщення, які завчасно будуть нагадувати студентам про дедлайни, надсилатимуть графіки консультацій та повідомлятимуть про необхідність завантажити проміжні звіти. Також планується впровадження електронного цифрового підпису. Це дозволить студентам і викладачам підписувати звіти та договори онлайн, без потреби друкувати документи чи передавати їх особисто. Такий підхід зробить документообіг значно зручнішим для всіх учасників процесу.

ВИСНОВКИ

Було розроблено та протестовано вебзастосунок для регулювання студентськими стажуваннями, використовуючи фреймворк Laravel та React. Аналіз предметної області підтвердив, що цифровізація освіти є необхідною для забезпечення якості вищої освіти в умовах сучасних викликів.

Спираючись на концепцію етапів цифровізації освіти, розроблений застосунок відповідає моделі цифрових перетворень [13]. Він не просто замінює паперові носії, а створює новий інструментарій для взаємодії через API та системи сповіщень. Відсутність спеціалізованих інструментів, що раніше створювала розрив у моніторингу, була усунута впровадженням єдиної платформи, яка використовує PostgreSQL для надійного зберігання даних та Laravel Sanctum для безпечної автентифікації користувачів.

Дослідження підкреслює, що практика є ключовою ланкою, яка пов’язує теорію з реальними навичками [14]. У застосунку це реалізовано через модуль генерації PDF-звітів, де аналіз студентських даних дозволяє вдосконалювати програми практики. Під час тестування продуктивності було встановлено, що використання при запуску застосунка у Firefox час виконання JS-логіки дорівнює 14.5 мс, а ось Chrom-браузер показує відгук інтерфейсу (INP) який дорівнює 48–56 мс, що підтверджує високу ефективність React-компонентів.

Окрім цього, створений вебзастосунок вирішує критичну проблему моніторингу підготовки фахівців [15]. Перевірка на безпеку допомогла знайти й прибрати вразливості, а також налагодити, права доступу. Додаток став кращим, коли було підключено Telegram Bot API, що дозволило досягти задоволеності користувачів у питанні оперативності сповіщень. Під час тестування було виявлено, що переважна більшість користувачів задоволені інтерфейсом програми. Особливо їм припали до душі функції попереднього перегляду та автоматичного заповнення даних про організації.

Тож, розроблений вебзастосунок поєднує потужність Laravel на бекенді та гнучкість React на фронтенді, та є важливим інструментом для цифровізації освітньої діяльності. Його впровадження оптимізує роботу навчального процесу, забезпечує прозорість оцінювання та сприяє підготовці конкурентоспроможних фахівців.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Information Technologies and Learning Tools Journal Article. Institute of Information Technologies and Learning Tools of NAES of Ukraine. URL: https://journal.iitta.gov.ua/index.php/itlt/article/view/6314 (date of access: 15.05.2026).

2. Інструкція з діловодства в міністерствах та інших органах виконавчої влади. Верховна Рада України. URL: https://zakon.rada.gov.ua/laws/show/z0035-93#Text (дата звернення: 15.05.2026).

3. Technology in Education – Global Education Monitoring Report. UNESCO. URL: https://www.unesco.org/gem-report/en/publication/technology (date of access: 15.05.2026).

4. About Moodle. Moodle. URL: https://docs.moodle.org/501/en/About_Moodle (date of access: 15.05.2026).

5. Classroom Help Center. Google. URL: https://support.google.com/edu/classroom#topic=10298088 (date of access: 15.05.2026).

6. Internship Management System. GitHub Repository. URL: https://github.com/tncrayt/internship-management-system (date of access: 15.05.2026).

7. Your First Component. React. URL: https://react.dev/learn (date of access: 15.05.2026).

8. Manual. PHP. URL: https://www.php.net/manual/en/index.php (date of access: 15.05.2026).

9. Documentation (Version 12.x). Laravel. URL: https://laravel.com/docs/12.x (date of access: 15.05.2026).

10. Documentation. PostgreSQL. URL: https://www.postgresql.org/docs/current (date of access: 15.05.2026).

11. Window.localStorage API. Mozilla Developer Network. URL: https://developer.mozilla.org/ru/docs/Web/API/Window/localStorage (date of access: 15.05.2026).

12. Офіційний портал статистичних даних. Державна служба статистики України. URL: https://stat.gov.ua/uk/explorer (дата звернення: 15.05.2026).

13. Цифровізація освітнього процесу в умовах дистанційного навчання в закладах вищої освіти. Педагогічні науки. URL: https://pednauk.cusu.edu.ua/index.php/pednauk/article/view/1174/1102 (дата звернення: 15.05.2026).

14. Practice as a link between theory and real skills. Springer Link. URL: https://link.springer.com/article/10.1007/s10639-025-13751-x (date of access: 15.05.2026).

15. Importance of monitoring systems in student training. Springer Link. URL: https://link.springer.com/article/10.1186/s12909-024-06607-4 (date of access: 15.05.2026).

ДОДАТОК А

Технічне завдання

Вступ

Вебзастосунок для регулювання студентськими стажуваннями призначений для автоматизації документообігу та контролю навчального процесу у закладах вищої освіти. Він розроблений для використання студентами, викладачами та адміністрацією кафедр, які потребують зручного інструменту для подання звітів, генерації документації та моніторингу успішності. Область застосування охоплює організацію та супровід виробничих і навчальних практик, а об’єктом використання є персональні комп’ютери та мобільні пристрої з доступом до мережі Інтернет.

А.1 Підстави до розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу, затверджене відповідним наказом Національного університету «Запорізька Політехніка» № 139 від 7 квітня 2026 року. Найменування теми розробки – «Розроблення та дослідження програмного забезпечення для управління студентськими стажуваннями».

А.2 Призначення розробки

Вебзастосунок призначений для автоматизації процесів збору метаданих звітів, автоматичного формування супровідних документів (рефератів, титульних сторінок) у форматі PDF та забезпечення швидкого зв'язку між учасниками через Telegram-бота. Його призначення полягає:

– для студентів: спрощенні підготовки звітної документації та отриманні миттєвих результатів перевірки;

– для керівників: полегшенні перевірки великої кількості робіт та централізованому зберіганні результатів практики;

– для адміністрації: прозорому управлінні базами практик, реєстрами користувачів та легким формуванням статистичних звітів.

А.3 Вимоги до програми чи програмного виробу

А.3.1 Вимоги до фунціоналних характеристик

Вебзастосунок для керування студентськими стажуваннями забезпечує повний цифровий супровід студентських стажувань: від постановки завдання, до генерації статистичних даних.

Спочатку, користувачі входять у систему через захищену форму з email та паролем. Залежно від ролі: студент, викладач або адміністратор, вони отримують персоналізований доступ із чітким розмежуванням прав до функціональних модулів та API. Також, викладачі можуть створювати індивідуальні завдання, встановлювати дедлайни та відстежувати прогрес групи в реальному часі. Студенти, в свою чергу, обирають місце практики з реєстру, переглядають вимоги до робіт і завантажують готові звіти. Далі система самостійно зчитує введені користувачем метадані та вставляє їх у шаблони титульних сторінок і рефератів. І згенеровані сторінки автоматично об’єднуються з PDF-файлом звіту, попередньо завантаженим студентом. Перед збереженням дані перевіряються на коректність, а імена файлів на сервері шифруються, задля гарантії безпеки та унеможливлення плутанини. Також, завдяки інтеграції з Telegram Bot API користувачі миттєво отримують сповіщення: схвалено чи відхилено звіт, додано новий коментар від викладача або виставлено оцінку. Це забезпечує оперативний зворотний зв’язок без потреби постійно оновлювати вебсторінку. На останок, адміністратори та керівництво кафедри можуть створювати статистичні PDF-звіти щодо загальної успішності практики, кількості зданих робіт та динаміки виконання завдань.

А.3.2 Вимоги до надійності

Для забезпечення надійного функціонування веб-додатка одночасно необхідно виконувати кілька технічних умов. Система працює на стабільних та перевірених технологіях – фреймворку Laravel та базі даних PostgreSQL. Це

гарантує, що дані будуть цілісними та надійно збереженими. Персональні дані шифруються, щоб ніхто сторонній не міг отримати до них доступ. Також, якщо було випадково введено некоректні дані або завантажено файл неправильного формату, система повідомить про помилку. Крім цього, передбачено захист від втрати даних, якщо раптом обірветься інтернет-з'єднання.

А.3.3 Умови експлуатації

Вебзастосунок має стабільно працювати на пристроях користувачів у стандартних умовах експлуатації офісної та побутової техніки (температура від +10°C до +35°C, вологість до 80%). Через те, що це хмарна система, важливо мати стабільне інтернет-з'єднання на стороні клієнта. Будь-який користувач повинен володіти базовими навичками роботи з персональним комп'ютером або смартфоном, вміти користуватися веббраузером та месенджером Telegram. Спеціальної технічної підготовки для роботи з інтерфейсом не треба завдяки інтуїтивно зрозумілій навігації, а адміністрування серверної частини може виконуватися системним адміністратором кафедри.

А.3.4 Вимоги до складу параметрів технічних засобів

Для стабільної роботи з вебзастосунком для регулювання студентськими стажуваннями потрібен ноутбук, планшет або смартфон, який має відповідати наступним мінімальним характеристикам:

– процесор: від 1.2 Ггц;
– оперативна пам’ять: від 2 Гб;
– вільне місце на диску: від 200 Мб;
– екран: роздільна здатність від 360x640 пікселів;
– операційна система: Windows 10/11, macOS, Linux, Android 7.0+ або iOS 13+;
– браузер: будь-який сучасний браузер (з підтримкою HTML5);

– месенджер: встановлений Telegram;
– інтернет-з’єднання: швидкість від 512 Кбіт/с.

Доступ до системи надається шляхом передачі URL-адреси та облікових даних (логін/пароль) авторизованим користувачам закладу освіти. Вимоги до фізичного маркування чи пакування відсутні, оскільки продукт є цифровим.

А.3.5 Вимоги до інформаційної й програмної сумісності

Вебзастосунок має бути повністю сумісним із сучасними браузерами (Chrome, Firefox, Safari, Edge) та коректно працювати на ОС Windows, macOS, Linux, iOS та Android. Інформаційна взаємодія між фронтендом на React та бекендом на Laravel забезпечується через REST API з використанням формату JSON. Програмна сумісність із сервером потребує наявності PHP 8.2+ та бази даних PostgreSQL для надійного збереження метаданих. Система інтегрується із Telegram Bot API для розсилки повідомлень та використовує спеціалізовані бібліотеки для коректної генерації та злиття PDF-документів. Архітектура бази даних відповідає стандартам SQL, що дозволяє легко експортувати статистику для зовнішньої аналітики.

А.3.6 Вимоги до маркування й упакування

Доступ до системи надається шляхом передачі URL-адреси та облікових даних (логін/пароль) авторизованим користувачам закладу освіти. Вимоги до фізичного маркування чи пакування відсутні, оскільки продукт є цифровим.

А.3.7 Вимоги до транспортування й збереження

Оскільки застосунок розповсюджується в цифровому вигляді, вимоги до транспортування та збереження не застосовуються.

А.4 Вимоги до програмної документації

Програмна документація до застосунку включає пояснювальну записку та технічне завдання. Ці документи містять інформацію про розробку, функціонал і вимоги до програми.

А.5 Техніко–економічні показники

Впровадження вебзастосунку має значний потенціал для підвищення ефективності освітнього процесу та комплексної оптимізації роботи кафедри. Завдяки автоматичній генерації PDF-файлів студенти зможуть зменшити час на підготовку документації у 3-4 рази. Збір робіт та автоматизація статистики суттєво зменшують навантаження на керівників, повністю виключаючи необхідність опрацювання паперових звітів. Інтеграція з Telegram-ботом забезпечує миттєву комунікацію між учасниками, прискорюючи інформування про статуси робіт. Економічна доцільність проєкту підтверджується використанням безкоштовних технологій з відкритим кодом, що дозволяє закладу освіти отримати потужний інструмент автоматизації без значних капіталовкладень.

А.6 Стадії й етапи розробки

Розробка включає наступні етапи:

– постановка завдання (1 тиждень);
– аналіз предметної області (1–2 тижні);
– розробка архітектури (1 тиждень);
– розробка програми (2 тижні);
– тестування (1 тиждень);
– оформлення документації (2 тижні).

А.7 Порядок контролю та приймання

Процес контролю якості передбачає поетапну перевірку всіх модулів системи. По-перше, модульне тестування підтвердило коректність обробки даних на сервері та генерації PDF. По-друге, інтеграційні тести забезпечили стабільний зв’язок між React, Laravel та Telegram API. Окремо було проведено перевірку інтерфейсу для гарантії зручності користувачів. Крім того, працездатність застосунку було підтверджено у різних браузерах. А також, після усунення виявлених проблем система повністю відповідає технічним вимогам і готова до впровадження.

ДОДАТОК Б

Текст програми

Б.1 Фрагмент файлу Login.jsx

```
const Login = () => {
    const [email, setEmail] = useState('');
    const [password, setPassword] = useState('');
    const [error, setError] = useState('');
    const navigate = useNavigate();
    const handleLogin = async (e) => {
        e.preventDefault();
        setError('');
        if (!email.endsWith('@zp.edu.ua')) {
              setError('Будь ласка, використовуйте корпоративну пошту
@zp.edu.ua');
            return;
        }
        try {
                                        const     response     =     await
axios.post('http://127.0.0.1:8000/api/login', {
                email,
                password
            });
            localStorage.setItem('token', response.data.access_token);
            localStorage.setItem('user_role', response.data.user.role);
            navigate('/dashboard');
        } catch (err) {
           setError(err.response?.data?.message || 'Помилка при вході');
    }};

    return (
        <div className="login-container">
            <div className="login-card">
                <h2>Вхід у систему</h2>
                <form onSubmit={handleLogin}>
                    <div className="login-form-group">
                                <input  type="email"  className="login-
input"placeholder="email@zp.edu.ua"       value={email}onChange={(e)       =>
setEmail(e.target.value)} required />
                    </div>
                    <div className="login-form-group">
                              <input  type="password"  className="login-
input"placeholder="Пароль"          value={password}onChange={(e)          =>
setPassword(e.target.value)} required />
```

```
                        </div>
                        {error && <p className="login-error">{error}</p>}
                        <button type="submit" className="login-button">
                            Увійти
                        </button>
        </form></div></div>);};
        export default Login;
```

Б.2 Фрагмент файлу AuthController.php

```
        class AuthController extends Controller{
            public function login(Request $request){
                        $request->validate(['email'  =>  ['required',  'email',
'regex:/^[a-zA-Z0-9._%+-]+@zp\.edu\.ua$/'], 'password' => 'required',], [
                        'email.regex' => 'Використовуйте лише корпоративну пошту
@zp.edu.ua']);
                    $user = User::with('role')->where('email', $request->email)-
>first();
                     if (! $user || ! Hash::check($request->password, $user-
>password)) {
                        return response()->json(['message' => 'Невірні дані для
входу'], 401);}
                    $token = $user->createToken('auth_token')->plainTextToken;
                    return response()->json(['access_token' => $token, 'token_type'
=> 'Bearer', 'user' => ['id' => $user->id, 'name' => $user->name, 'email' =>
$user->email, 'role' => $user->role->name,]]);
            }
            public function logout(Request $request)
            {
                $request->user()->currentAccessToken()->delete();
                return response()->json(['message' => 'Вихід успішний']);
            }
        }
```

Б.3 Фрагмент файлу AdminController.php

```
        class AdminController extends Controller{
            public function storeUser(Request $request){
                        $validated    =    $request->validate(['name'    =>
'required|string|max:255',  'email'  =>  'required|email|unique:users,email',
```

```
'password' => 'required|string|min:8', 'role_id' => 'required|in:2,3', 'group'
=>        'required_if:role_id,3|nullable|string',        'faculty'        =>
'required_if:role_id,3|nullable|string',  'department'  =>  'required|string',
'position' => 'required_if:role_id,2|nullable|string', 'telegram_chat_id' =>
'nullable|string|unique:users,telegram_chat_id',]);
              try {
                       return  DB::transaction(function  ()  use  ($validated,
$request){
                      $telegram_id = $request->telegram_chat_id;

                      if (empty(trim($telegram_id))) {
                          $telegram_id = null;}
                         $user = User::create(['name' => $validated['name'],
'email'         =>         $validated['email'],         'password'         =>
Hash::make($validated['password']),   'role_id'   =>   $validated['role_id'],
'telegram_chat_id' => $telegram_id,]);
                      if ($user->telegram_chat_id) {
                          $msg = "🔐 <b>Ваш аккаунт створено!</b>\n\n"
                              . "Логін: <code>{$user->email}</code>\n"
                              . "Пароль: <code>{$request->password}</code>";
                          $this->sendTelegramMessage($user->telegram_chat_id,
$msg); }
                      if ($validated['role_id'] == 3) {
                          Student::create(['user_id' => $user->id, 'group' =>
$validated['group'],  'faculty'  =>  $validated['faculty'],  'department'  =>
$validated['department'],]);
                      } elseif ($validated['role_id'] == 2) {
                        Supervisor::create(['user_id' => $user->id, 'position'
=> $validated['position'], 'department' => $validated['department'],]); }
                       return response()->json(['message' => 'Користувача та
профіль успішно створено', 'user' => $user], 201); });
              } catch (\Exception $e) {
                       Log::error("Помилка  створення  користувача:  "  .  $e-
>getMessage());
                  return response()->json(['message' => 'Сталася помилка при
збереженні даних', 'error' => $e->getMessage()], 500); }}
          public function getDetailedStatistics(Request $request) {
              $currentYear = now()->year;
               $groupStats = \App\Models\Student::with(['user', 'reports']) -
>select('group')     ->selectRaw('count(*)     as     total_students')     -
>selectRaw('count(CASE WHEN id IN (select student_id from reports where grade
IS NOT NULL) THEN 1 END) as passed_count') ->selectRaw('avg((select grade from
```

```
reports where student_id = students.id limit 1)) as average_grade') -
>groupBy('group') ->get()->map(function($item) {
                        $item->pass_percentage = $item->total_students > 0
                          ? round(($item->passed_count / $item->total_students)
* 100, 2): 0; return $item; });
               $departmentStats = [
                   'total_students' => $groupStats->sum('total_students'),
                   'total_passed' => $groupStats->sum('passed_count'),
                                'avg_grade_total'    =>    round($groupStats-
>avg('average_grade'), 2),
                   'pass_rate_total' => $groupStats->sum('total_students') > 0
                      ? round(($groupStats->sum('passed_count') / $groupStats-
>sum('total_students')) * 100, 2): 0];
                return response()->json(['year' => $currentYear, 'groups' =>
$groupStats, 'department' => $departmentStats]);
         }}
```

Б.4 Фрагмент файлу AdminDashboard.jsx

```
       const AdminDashboard = () => {
           const [activeTab, setActiveTab] = useState('users');
           const [stats, setStats] = useState(null);
           const [loadingStats, setLoadingStats] = useState(false);
           const [formData, setFormData] = useState({name: '', email: '',: '',
role_id:  '3',  group:  '',  faculty:  '',  department:  '',  position:  '',
telegram_chat_id: ''});
           const generatePDF = () => {
               if (!stats || !stats.groups) {
                   alert("No data available for report");
                   return; }
               const doc = new jsPDF();
               doc.setFontSize(18);
               doc.text("Department Academic Performance Report", 14, 20);
               doc.setFontSize(12);
               doc.text(`Academic Year: ${stats.year}`, 14, 30);
                                    doc.text(`Overall      Pass      Rate:
${stats.department?.pass_rate_total}%`, 14, 37);
                           doc.text(`Department    Grade    Point    Average:
${stats.department?.avg_grade_total}`, 14, 44);
               const tableColumn = ["Group", "Students", "Passed", "Pass Rate
%", "Avg. Grade"];
```

```
                    const tableRows = stats.groups.map(g => [g.group ||
"N/A",g.total_students,        g.passed_count,        `${g.pass_percentage}%`,
g.average_grade ? Number(g.average_grade).toFixed(2) : "0.00"]);
                autoTable(doc, {
                    head: [tableColumn],
                    body: tableRows,
                    startY: 52,
                    theme: 'striped', });
                doc.save(`Report_${stats.year}.pdf`);};
            useEffect(() => {
                if (activeTab === 'stats') {
                    setLoadingStats(true);
                     axios.get('http://localhost:8000/api/admin/statistics/deta
iled')
                        .then(res => {
                            setStats(res.data);
                            setLoadingStats(false); })
                        .catch(err => {
                              console.error("Помилка завантаження статистики:",
err);
                            setLoadingStats(false); });
              }}, [activeTab]);
            const resetForm = () => {
                setFormData({
                    name: '', email: '', password: '', role_id: '3',
                       group: '', faculty: '', department: '', position: '',
telegram_chat_id: ''
             });};
            const handleSubmit = async (e) => {
                e.preventDefault();
                try {
                      await axios.post('http://localhost:8000/api/admin/users',
formData);
                    alert('Користувача успішно створено!');
                    resetForm();
                } catch (error) {
                    console.error(error);
                   alert('Помилка: ' + (error.response?.data?.message || 'Сервер
недоступний'));
            }};
            return (
            <div className="admin-layout">
                <div className="container py-4">
```

```
                    <header className="d-flex justify-content-between align-
items-center mb-4">
                        <h2 className="fw-bold m-0">Адмін-панель</h2>
                    </header>
                    <ul className="nav-pills-custom">
                        <li className="nav-item">
                            <button className={`nav-btn ${activeTab === 'users'
? 'active' : ''}`} onClick={() => setActiveTab('users')}>
                                <i className="fas fa-user-plus me-2"></i>
Користувачі
                            </button>
                        </li>
                        <li className="nav-item">
                            <button className={`nav-btn ${activeTab === 'stats'
? 'active' : ''}`} onClick={() => setActiveTab('stats')}>
                                <i className="fas fa-chart-line me-2"></i>
Статистика
                            </button>
                        </li>
                    </ul>
                    <div className="tab-content">
                        {activeTab === 'users' && (
                            <div className="app-card animate__animated
animate__fadeIn">
                                <h3>Реєстрація користувача</h3>
                                <form onSubmit={handleSubmit}>
                                    <div className="row">
                                        <div className="col-md-6 f-group">
                                            <label className="f-label">ПІБ</label>
                                                <input type="text" className="f-
control"       required       value={formData.name}       onChange={e       =>
setFormData({...formData, name: e.target.value})} />
                                        </div>
                                        <div className="col-md-6 f-group">
                                                <label  className="f-
label">Email</label>
                                            <input type="email" className="f-
control"       required       value={formData.email}       onChange={e       =>
setFormData({...formData, email: e.target.value})} />
                                        </div>
                                        <div className="col-md-6 f-group">
                                                <label  className="f-
label">Пароль</label>
```

```
                                        <input type="password" className="f-
control"      required      value={formData.password}      onChange={e      =>
setFormData({...formData, password: e.target.value})} />
                                    </div>
                                    <div className="col-md-6 f-group">
                                    <label className="f-label">Роль</label>
                                            <select  className="f-select"
value={formData.role_id}  onChange={e  =>  setFormData({...formData,  role_id:
e.target.value})}>
                                     <option value="3">Студент</option>
                                    <option value="2">Викладач</option>
                                    </select>
                                    </div>
                                    {formData.role_id === '3' ? (
                                        <>
                                                <div className="col-md-4 f-
group">
                                                        <label  className="f-
label">Група</label>
                                                <input className="f-control"
value={formData.group}   onChange={e   =>   setFormData({...formData,   group:
e.target.value})} />
                                            </div>
                                                <div className="col-md-4 f-
group">
                                                        <label  className="f-
label">Факультет</label>
                                                <input className="f-control"
value={formData.faculty}  onChange={e  =>  setFormData({...formData,  faculty:
e.target.value})} />
                                            </div>
                                                <div className="col-md-4 f-
group">
                                                        <label  className="f-
label">Кафедра</label>
                                                <input className="f-control"
value={formData.department}    onChange={e    =>    setFormData({...formData,
department: e.target.value})} />
                                            </div>
                                        </>) : (
                                        <>
                                                <div className="col-md-6 f-
group">
```

```
                                                    <label className="f-
label">Посада</label>
                                                <input className="f-control"
value={formData.position} onChange={e => setFormData({...formData, position:
e.target.value})} />
                                            </div>
                                                <div className="col-md-6 f-
group">
                                                    <label className="f-
label">Кафедра</label>
                                                <input className="f-control"
value={formData.department}    onChange={e    =>    setFormData({...formData,
department: e.target.value})} />
                                    </div></>)}
                                        <div className="col-md-12 f-group">
                                            <label className="f-label text-
primary">
                                              <i className="fab fa-telegram
me-2"></i>Telegram Chat ID
                                        </label>
                                          <input type="text" className="f-
control" placeholder="123456789" value={formData.telegram_chat_id} onChange={e
=> setFormData({...formData, telegram_chat_id: e.target.value})} />
                                    </div></div>
                                        <button type="submit" className="btn-
main">Зареєструвати</button>
                            </form></div>)}
                        {activeTab === 'stats' && (
                            <div className="animate__animated animate__fadeIn">
                                {loadingStats ? (
                                    <div className="text-center py-5">
                                        <div className="spinner-border text-
primary" role="status"></div>
                                            <p  className="mt-2">Опрацювання
даних...</p>
                                    </div>) : stats ? (<>
                                        <div className="pdf-wrapper">
                                            <button  className="btn-pdf"
onClick={generatePDF}>
                                             <i  className="fas fa-file-
pdf"></i> PDF Звіт
                                        </button>
                                        </div>
```

```
                                                <h2 className="mb-4 fw-bold"><i
className="fas fa-chart-bar"></i> Аналітика за {stats.year} рік</h2>
                                            <div className="stats-grid mb-5">
                                              <div className="stat-card bg-primary-
grad shadow-sm">
                                                    <h6>Успішність кафедри</h6>
                                                     <h3>{stats.department?.pass_ra
te_total}%</h3>
                                              </div>
                                             <div className="stat-card bg-success-
grad shadow-sm">
                                                    <h6>Середній бал</h6>
                                                     <h3>{stats.department?.avg_gra
de_total}</h3>
                                            </div></div>
                                            <div className="table-box">
                                                <table className="app-table">
                                                    <thead>
                                                        <tr>
                                                             <th>Група</th><th>Студ
ентів</th><th>Здали</th><th>Відсоток</th><th>Сер. бал</th></tr>
                                                    </thead>
                                                    <tbody>
                                                        {stats.groups?.map(group =>
(
                                                            <tr key={group.group}>
                                                                 <td><strong>{group
.group}</strong></td><td>{group.total_students}</td><td>{group.passed_count}<
/td><td><span                                                 className="badge-
pass">{group.pass_percentage}%</span></td><td><strong>{group.average_grade   ?
Number(group.average_grade).toFixed(2) : '—'}</strong></td>
                                                            </tr>
                                                        ))}
                                                    </tbody></table></div></>) : (
                                          <div className="alert alert-info rounded-
pill px-4">Дані відсутні.</div>
                                    )} </div>)}</div></div>div>);};
       export default AdminDashboard;
```

Б.5 Фрагмент файлу ReportController.php

```
class ReportController extends Controller{
    protected $documentService;
    public function __construct(DocumentService $documentService) {
        $this->documentService = $documentService; }
    public function store(Request $request) {
        $messages = ['report_file.mimes' => 'Будь ласка, завантажте файл
у форматі PDF.', 'report_file.max' => 'Файл занадто великий (макс. 10 МБ).',];
                                $request->validate(['report_file'      =>
'required|mimes:pdf|max:10240', 'task_id' => 'required|integer', 'topic' =>
'required|string|max:255',  'object'  =>  'nullable|string',  'subject'  =>
'nullable|string',  'goal'  =>  'nullable|string',  'hardware_software'  =>
'nullable|string',   'results'   =>   'nullable|string',   'keywords'   =>
'nullable|string',      'external_supervisor'      =>      'nullable|string',
'custom_practice_place_name'          =>          'nullable|string|max:255',
'custom_practice_place_address'         =>         'nullable|string|max:255',
'practice_place_id' => 'nullable|integer', $messages);
            $user = $request->user();
            $student = $user->student;
             $student->update($request->only(['custom_practice_place_name',
'custom_practice_place_address', 'practice_place_id'
            ]));
            if ($request->custom_practice_place_name) {
                $student->update(['practice_place_id' => null]);
            }
            $student->load('practicePlace');
            $student->refresh();
            if (!$student) {
               return response()->json(['message' => 'Тільки студенти можуть
завантажувати звіти'], 403);
            }
             $student->update($request->only(['custom_practice_place_name',
'custom_practice_place_address', 'practice_place_id'
            ]));
            if ($request->custom_practice_place_name) {
                $student->update(['practice_place_id' => null]);
            }
            $student->refresh();
              $report  =  Report::firstOrNew(['student_id'  =>  $student->id,
'task_id' => $request->task_id,
            ]);
```

```
            if ($report->exists && $report->status === 'approved') {
                return response()->json(['message' => 'Звіт вже оцінено.'],
403);
            }
            $report->fill($request->only([
                'topic', 'object', 'subject', 'goal',
                            'hardware_software',   'results',   'keywords',
'external_supervisor'
            ]));
            if ($request->hasFile('report_file')) {
                try {
                                    $finalPath   =   $this->documentService-
>processReport($report, $request->file('report_file'));

                     if ($report->exists && $report->original_file_path &&
Storage::exists($report->original_file_path)) {
                        Storage::delete($report->original_file_path);
                    }
                    $report->original_file_path = $finalPath;
                    $report->status = 'pending';
                    $report->year = date('Y');
                    $report->save();
                    try {
                        $teacher = $report->task->supervisor->user;

                        if ($teacher && $teacher->telegram_chat_id) {
                            $msg = "📥 <b>Новий звіт на перевірку!</b>\n\n"
                                . "👤 Студент: <b>{$user->name}</b>\n"
                                . "📚 Тема: {$report->topic}\n"
                                                                . "📎 <a
href='".env('APP_URL')."/admin/reports/{$report->id}'>Переглянути в
системі</a>";
                                        $this->sendTelegramMessage($teacher-
>telegram_chat_id, $msg);
                        }
                    } catch (\Exception $e) {
                            Log::error("Помилка Telegram (store): " . $e-
>getMessage());
                    }
                    $message = $report->wasRecentlyCreated ? 'Звіт успішно
створено' : 'Звіт успішно оновлено';
```

```
                    return response()->json(['message' => $message, 'report'
=> $report], 201);
                } catch (\Exception $e) {
                    return response()->json(['message' => 'Помилка при
генерації документа: ' . $e->getMessage()], 500);
                }
            }
            return response()->json(['message' => 'Файл не знайдено'], 400);
        }

        public function preview(Request $request)
        {
                                $request->validate(['report_file'      =>
'required|mimes:pdf|max:10240',  'topic'  =>  'required|string',  'object'  =>
'nullable|string', 'subject' => 'nullable|string', 'goal' => 'nullable|string',
'hardware_software'  =>  'nullable|string',  'results'  =>  'nullable|string',
'keywords' => 'nullable|string', 'external_supervisor' => 'nullable|string',
'custom_practice_place_name'          =>          'nullable|string|max:255',
'custom_practice_place_address'         =>         'nullable|string|max:255',
'practice_place_id' => 'nullable|integer',]);
            $user = $request->user();
            $student = $user->student;
                    $student->custom_practice_place_name   =   $request-
>custom_practice_place_name;
                $student->custom_practice_place_address   =   $request-
>custom_practice_place_address;
            $student->practice_place_id = $request->practice_place_id;

                    if   ($request->practice_place_id   &&   !$request-
>custom_practice_place_name) {
                $student->load('practicePlace');
            }
            if ($request->custom_practice_place_name) {
                $student->practice_place_id = null;
                $student->setRelation('practicePlace', null);
            }
                    $student->custom_practice_place_name   =   $request-
>custom_practice_place_name;
                $student->custom_practice_place_address   =   $request-
>custom_practice_place_address;
            if ($request->custom_practice_place_name) {
                $student->practice_place_id = null;
            }
```

```
                $tempReport = new Report($request->all());
                $tempReport->setRelation('student', $student);
                try {
                                    $pdfContent    =    $this->documentService-
>previewReport($tempReport, $request->file('report_file'));
                    return response($pdfContent)
                        ->header('Content-Type', 'application/pdf')
                                     ->header('Content-Disposition',   'inline;
filename="preview.pdf"');
                } catch (\Exception $e) {
                    return response()->json(['message' => 'Помилка прев\'ю: ' .
$e->getMessage()], 500);
                }
            }
            public function download(Request $request, $id)
            {
                if ($user->role === 'student' && $report->student_id !== $user-
>student->id) {abort(403, 'У вас немає прав для завантаження цього файлу'); }

                 if (!Storage::exists($report->original_file_path)) {abort(404,
'Файл не знайдено на сервері'); }
                return Storage::download(
                    $report->original_file_path,
                    "Звіт_практика_{$report->student->user->name}.pdf"
                );
            }
            public function updateGrade(Request $request, $id)
            {
               $request->validate(['grade' => 'required|integer|min:0|max:100',
'status' => 'required|in:approved,rejected', 'comment' => 'nullable|string']);
                $report = Report::findOrFail($id);
               $report->update($request->only(['grade', 'status', 'comment']));
                try {
                    $studentUser = $report->student->user;
                    if ($studentUser && $studentUser->telegram_chat_id) {
                           $statusText = $report->status === 'approved' ? "✅
Схвалено" : "❌ Потребує доопрацювання";
                        $msg = "🔔 <b>Ваш звіт перевірено!</b>\n\n"
                            . "📊 Оцінка: <b>{$report->grade}</b>\n"
                            . "📝 Статус: {$statusText}\n"
```

```
                            . ($report->comment ? "💬 Коментар: <i>{$report->comment}</i>" : "");
                                        $this->sendTelegramMessage($studentUser->telegram_chat_id, $msg);
                    }
                    } catch (\Exception $e) {Log::error("Помилка Telegram (updateGrade): " . $e->getMessage());}
                return response()->json(['message' => 'Оцінку виставлено. Звіт заблоковано для змін.']);
            }
        }
```

Б.6 Фрагмент файлу StudentDashboard

```
        return (
                <div className="dashboard-grid">
                        <section className="panel-section" style={{ opacity: isApproved ? 0.7 : 1 }}>
                        <h3><i className="fas fa-edit"></i> Параметри звіту</h3>
                        <fieldset disabled={isApproved} style={{ border: 'none', padding: 0 }}>
                            <div className="field-group">
                                <div className="input-card">
                                            <label className="label-text">Місце проходження практики:</label>
                                    <label style={{ display: 'flex', alignItems: 'center', gap: '8px', marginBottom: '10px', cursor: 'pointer', fontSize: '14px' }}>
                                                <input type="checkbox" checked={isManualInput} onChange={() => setIsManualInput(!isManualInput)} />
                                                <span>Ввести назву підприємства вручну</span>
                                    </label>
                                    {isManualInput ? (
                                        <div className="field-group" style={{ gap: '8px' }}>
                                                <input className="custom-input" type="text" placeholder="Повна назва підприємства" value={formData.custom_practice_place_name || ''} onChange={e => updateForm('custom_practice_place_name', e.target.value)} />
```

```
                                        <input className="custom-input"
type="text"      placeholder="Адреса      (м.      Запоріжжя,      вул...)"
value={formData.custom_practice_place_address    ||    ''}    onChange={e    =>
updateForm('custom_practice_place_address', e.target.value)} />
                                    </div>) : (
                                        <select className="custom-select"
value={formData.practice_place_id      ||      ''}      onChange={e      =>
updateForm('practice_place_id', e.target.value)}>
                                            <option value="">Оберіть базу
практики зі списку...</option>
                                        {companies.map(c => <option key={c.id}
value={c.id}>{c.name}</option>)}
                                    </select>)}
                                <div style={{ marginTop: '12px' }}>
                                    <label className="label-text">Керівник
від підприємства:</label>
                                <input className="custom-input" type="text"
placeholder="ПІБ,   посада"   value={formData.external_supervisor   ||   ''}
onChange={e => updateForm('external_supervisor', e.target.value)} />
                                </div></div><div>
                                        <label  className="label-text">Тема
роботи:</label>
                                <input className="custom-input" type="text"
value={formData.topic    ||    ''}    onChange={e    =>    updateForm('topic',
e.target.value)} />
                            </div>
                                    <div  style={{  display:  'grid',
gridTemplateColumns: '1fr 1fr', gap: '15px' }}>
                                <div>
                                    <label className="label-text">Об'єкт
дослідження:</label>
                                <input className="custom-input" type="text"
value={formData.object    ||    ''}    onChange={e    =>    updateForm('object',
e.target.value)} />
                                </div><div>
                                    <label className="label-text">Предмет
дослідження:</label>
                                <input className="custom-input" type="text"
value={formData.subject    ||    ''}    onChange={e    =>    updateForm('subject',
e.target.value)} />
                                </div></div><div>
                                        <label  className="label-text">Мета
роботи:</label>
```

```
                                        <textarea className="custom-textarea"
style={{ minHeight: '60px' }} value={formData.goal || ''} onChange={e =>
updateForm('goal', e.target.value)} />
                                </div><div>
                                    <label className="label-text">Обладнання та
ПЗ:</label>
                                    <input className="custom-input" type="text"
value={formData.hardware_software || ''} onChange={e =>
updateForm('hardware_software', e.target.value)} />
                                </div><div>
                                                <label  className="label-
text">Результати:</label>
                                        <textarea className="custom-textarea"
style={{ minHeight: '60px' }} value={formData.results || ''} onChange={e =>
updateForm('results', e.target.value)} />
                                </div><div>
                                        <label  className="label-text">Ключові
слова:</label>
                                    <input className="custom-input" type="text"
value={formData.keywords || ''} onChange={e => updateForm('keywords',
e.target.value)} />
                                </div>
                                <button onClick={saveDraft} className="btn-base
btn-secondary"><i className="fas fa-file-alt"></i> Зберегти чернетку</button>
                            </div>
                        </fieldset>
                    </section>

                    <section className="panel-section">
                       <h3><i className="fas fa-folder-open"></i> Мій звіт</h3>
                        {!isLocked && (
                              <div  className="app-card" style={{ marginBottom:
'24px' }}>
                                  <h5 style={{ margin: '0 0 10px 0' }}>Крок 1:
Завантажте шаблон</h5>
                                <p style={{ fontSize: '14px', margin: '0 0 15px
0' }}>Використовуйте цей файл для написання основної частини звіту:</p>
                                                <a  href="/template_student.docx"
download="Шаблон_Звіту.docx" className="btn-base btn-outline">
                                                 <i  className="fas  fa-file-word"></i>
Завантажити шаблон (.docx)
                                </a></div>)}
```

```
                        <hr style={{ border: 'none', borderTop: '1px solid var(-
--border)', margin: '20px 0' }} />
                         <div className="upload-container">
                              <h5 style={{ margin: '0 0 15px 0' }}>Крок 2: Стан
звіту</h5>
                             {isApproved && (
                                   <div className="status-badge status-approved
animate__animated animate__fadeIn">
                                     <h4 style={{ margin: 0 }}>
                                          <i className="fas fa-check-circle me-
2"></i>
                                         Ваша оцінка: {currentReport.grade}
                                     </h4>
                                      <p style={{ fontSize: '12px', margin: '5px
0 0' }}>Звіт прийнято. Редагування заблоковано.</p>
                                </div>)}
                             {isRejected && (
                                  <div className="status-badge status-rejected"
style={{ backgroundColor: '#fff5f5', border: '1px solid #feb2b2', padding:
'15px', borderRadius: '8px', marginBottom: '15px' }}>
                                       <h4 style={{ color: '#c53030', margin: 0
}}><i className="fas fa-xmark"></i> Роботу відхилено</h4>
                                       <p style={{ color: '#742a2a', marginTop:
'5px' }}>
                                                          <strong>Причина:</strong>
{currentReport.comment || "Не вказано"}
                                     </p>
                                    <p style={{ fontSize: '13px', color: '#c53030'
}}>Будь ласка, виправте зауваження та надішліть файл ще раз.</p>
                                </div>)}
                             {hasReport && (
                                    <div className="status-badge status-info"
style={{ marginTop: '10px' }}>
                                     <p style={{ margin: '0 0 10px 0', fontWeight:
'bold' }}>Поточна версія звіту (PDF):</p>
                                                          <button onClick={() =>
downloadFile(currentReport.id)} className="btn-base btn-primary">
                                         <i className="fas fa-file-import"></i>
Скачати згенерований файл
                                     </button>
                                </div>)}
                             {!isLocked ? (
```

```
                                    <div className="upload-actions" style={{
marginTop: '20px' }}>
                                <h4 style={{ fontSize: '16px', marginBottom:
'10px' }}>
                                    <i className="fas fa-rocket"></i>
                                        {isRejected ? " Подати виправлену
версію:" : hasReport ? " Оновити файл та дані:" : " Завантажити основну
частину:"}
                                </h4>
                                <div className="drop-zone">
                                        <input type="file" accept=".pdf"
onChange={e => setFile(e.target.files[0])} /></div>
                                    <div className="field-group" style={{
display: 'flex', gap: '10px', marginTop: '15px' }}>
                                        <button onClick={handlePreview}
disabled={previewLoading} className="btn-base btn-warning">
                                            <i className="fas fa-magnifying-
glass"></i>
                                        {previewLoading ? ' Генерація...' :
' Попередній перегляд титулки'}
                                    </button>
                                        <button onClick={handleFileUpload}
disabled={uploading} className="btn-base btn-success" style={{ flex: 1 }}>
                                        <i className="fas fa-rocket"></i>
                                            {uploading ? ' Відправка...' :
(isRejected || hasReport) ? ' Переподати звіт' : ' Згенерувати та відправити'}
                                    </button>
                                </div></div>) : (
                            !isApproved && (
                                <div className="app-card" style={{ textAlign:
'center', color: '#666', marginTop: '20px' }}>
                                    <p style={{ margin: 0 }}><i className="fas
fa-lock"></i> Звіт знаходиться в архіві або на перевірці. Зміни неможливі.</p>
                                </div>))}</div></section></div>
```

Б.7 Фрагмент файлу DocumentService.php

```
class DocumentService{
    public function processReport($report, $uploadedFile) {
                $tempTitlePath    =    $this->generateTitlePdf($report,
$uploadedFile);
```

```
                try {
                        $finalPdfContent = $this->mergePdfs([$tempTitlePath,
$uploadedFile->getRealPath()]);
                    $fileName = 'reports/' . bin2hex(random_bytes(16)) . '.pdf';
                    if (!Storage::exists('reports')) {
                        Storage::makeDirectory('reports');
                    }
                    Storage::put($fileName, $finalPdfContent);
                    return $fileName;
                } finally {
                    $this->cleanup($tempTitlePath);
                }}

            public function previewReport($report, $uploadedFile) {
                try {
                            $tempTitlePath = $this->generateTitlePdf($report,
$uploadedFile);
                    try {
                        return $this->mergePdfs([$tempTitlePath, $uploadedFile-
>getRealPath()]);} finally {
                        $this->cleanup($tempTitlePath);
                    }
                } catch (\Exception $e) {abort(500, 'Помилка в сервісі: ' . $e-
>getMessage() . ' у файлі ' . $e->getFile() . ' на рядку ' . $e->getLine());}
            }
            private function generateTitlePdf($report, $uploadedFile)
                $fpdi = new Fpdi();
                try {
                            $pageCount = $fpdi->setSourceFile($uploadedFile-
>getRealPath());
                } catch (Exception $e) {throw new Exception("Неможливо прочитати
PDF файл: " . $e->getMessage());}
                     $student = $report->student()->with(['supervisor.user',
'user'])->first();
                if (!$student && Auth::check()) {
                   $student = \App\Models\Student::where('user_id', Auth::id())-
>with(['supervisor.user', 'user', 'practicePlace']) ->first();
                }
                 $pdf = Pdf::loadView('practice_report', ['report' => $report,
'student' => $student, 'page_count' => $pageCount + 2])->setPaper('a4',
'portrait');
                $tempPath = storage_path('app/temp_' . uniqid() . '.pdf');
                $pdf->save($tempPath);
```

```
                return $tempPath;
            }
                private function mergePdfs(array $filePaths) {
                    $finalPdf = new Fpdi('P', 'mm', 'A4');
                    $a4Width = 210;
                    $a4Height = 297;
                    foreach ($filePaths as $path) {
                        if (!file_exists($path)) continue;
                        $count = $finalPdf->setSourceFile($path);
                        for ($i = 1; $i <= $count; $i++) {
                            $tplIdx = $finalPdf->importPage($i);
                            $specs = $finalPdf->getTemplateSize($tplIdx);
                           $isLandscape = ($specs['width'] > $specs['height']);
                            if ($isLandscape) {
                                $finalPdf->AddPage('L', 'A4');
                               $finalPdf->useTemplate($tplIdx, 0, 0, $a4Height,
$a4Width, true);
                            } else {
                                $finalPdf->AddPage('P', 'A4');
                                $finalPdf->useTemplate($tplIdx, 0, 0, $a4Width,
$a4Height, true);
                     }}}
                    return $finalPdf->Output('S');
                }
            private function cleanup($path) {
                if (file_exists($path)) {unlink($path);
        }}}
```

Б.8 Фрагмент файлу Controller.php

```
        abstract class Controller{
            protected function sendTelegramMessage($chatId, $text) {
                $token = "8619154018:AAFsKdZm8WOsZsKkunPNWeZuwJjYzUg4214";
                try {
                    Http::withOptions([
                        'verify' => false,
                   ])->post("https://api.telegram.org/bot{$token}/sendMessage",
['chat_id' => $chatId, 'text' => $text, 'parse_mode' => 'HTML']);
                 } catch (\Exception $e) {Log::error("Telegram error: " . $e-
>getMessage());}
        }}
```

ДОДАТОК В

Слайди презентації

Національний університет «Запорізька Політехніка»
Кафедра програмних засобів

Розроблення та дослідження програмного забезпечення для управління студентськими стажуваннями

Виконав
Студент групи КНТ-112 Жанна ЧОРНА

Керівник
к.т.н.доцент Євгеній ГОФМАН

2026

Рисунок В.1 – Слайд №1

Об’єкт та мета дослідження

Об'єкт

Процес автоматизації та забезпечення якості управління студентськими стажуваннями.

Предмет

Методи та інструменти для розробки вебзастосунку на базі архітектури клієнт-сервер.

Мета

Підвищення ефективності та рівня автоматизації управління стажуваннями.

2

Рисунок В.2 – Слайд №2

Аналіз існуючих рішень

Критерій	Google Classroom	GitHub IMS	Moodle	Вебзастосунок
Автоматизація	Відсутня	Відсутня	Відсутні	Автогенерація титулок та рефератів
Сповіщення	Email та Push	Відсутні	Внутрішні та Email	Telegram–бот (миттєві дедлайни)
Інтерфейс	Обмежений	Середній	Складний, перевантажений налаштуваннями	Інтуїтивний (покрокові дії)
База даних	Google Cloud	MySQL	Хмарна або Локальна	PostgreSQL + API
Управління документами практики	Відсутні	Відсутні	Відсутні	Шаблони, автозаповнення, PDF
Завдання для практики	Є, але не адаптовані під стажування	Є, але без оцінювання	Є, але не адаптовані під стажування	Керівник створює завдання
Ролі	Лише викладач та учень	Адміністратор, персонал, викладач та стажер	Багато ролей, але не для стажувань	Студент, керівник практики, адміністратор

3

Рисунок В.3 – Слайд №3

Обрані технології розробки

Критерій	PHP
Продуктивність	Висока для веб
Швидкість розробки	Дуже висока
Підтримка PostgreSQL	Повна
Інтеграція з React	Дуже проста
Спільнота	Найбільша
Вбудовані засоби Web	Сесії, Cookies, Роутинг

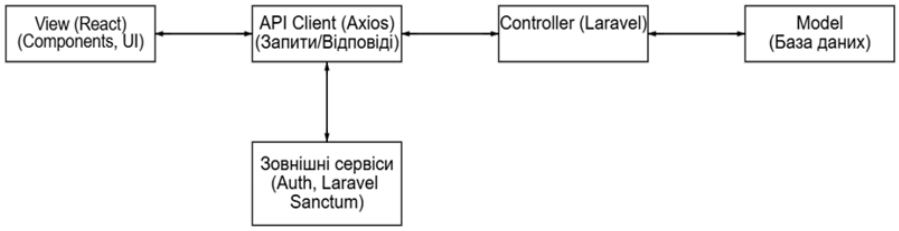
Критерій	React
Мова програмування	JavaScript / TypeScript
Гнучкість архітектури	Висока
Продуктивність	Висока завдяки Virtual DOM
Масштабованість	Висока
Підтримка спільноти	Дуже велика
Підходить для великих проєктів	Так
Кількість бібліотек	Дуже велика

Критерій	PostgreSQL
Тип БД	Реляційна
Цілісність даних	Дуже висока (строге дотримання ACID)
Складні запити	Ідеально
Робота з JSON	Відмінна
Безпека	Шифрування (на сервері)
Складність інтеграції	Складна (потрібен сервер) але легка з Laravel
Підходить для великих даних	Так

4

Рисунок В.4 – Слайд №4

Взаємодія компонентів системи

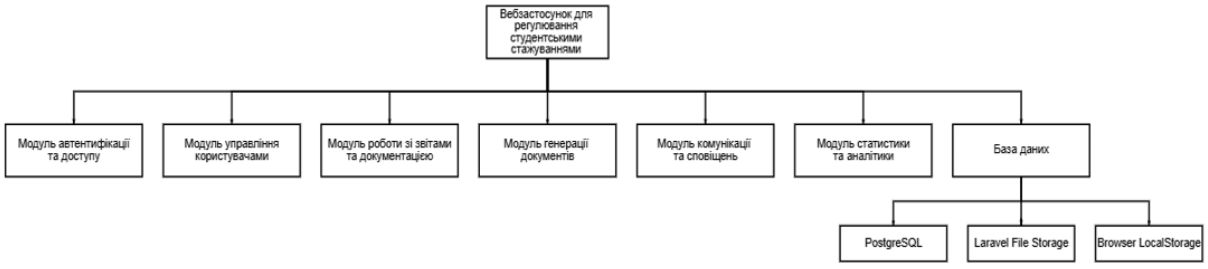


- **View (React):** Користувацький інтерфейс та компоненти UI.
- **API Client:** Використання Axios для запитів до серверу.
- **Backend:** Laravel Controller та Model для обробки даних.

5

Рисунок В.5 – Слайд №5

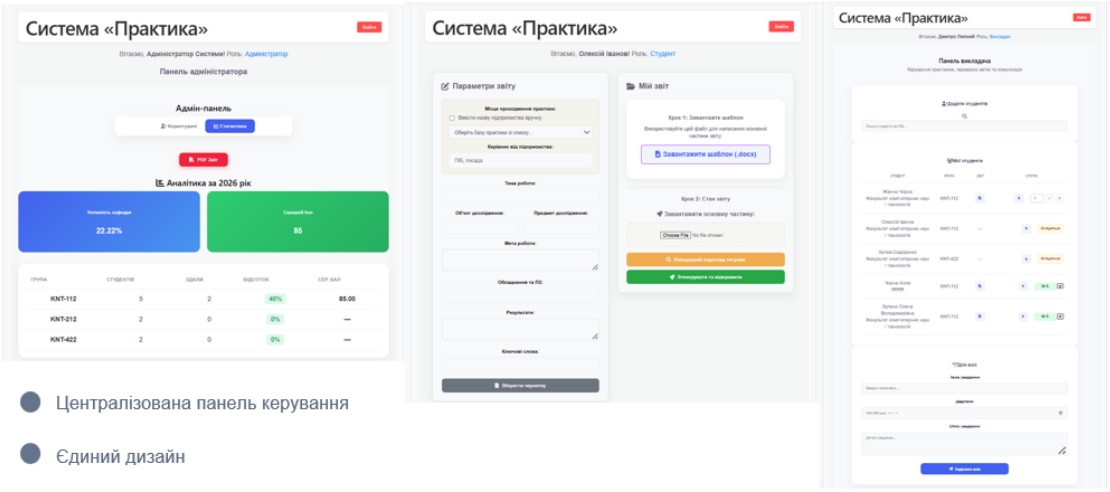
Структурна схема вебзастосунку



6

Рисунок В.6 – Слайд №6

Інтерфейс користувача



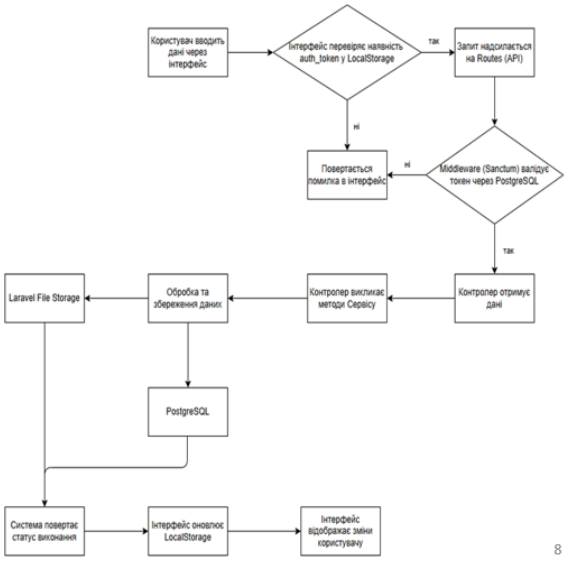
- Централізована панель керування
- Єдиний дизайн
- Інтуїтивний UX

7

Рисунок В.7 – Слайд №7

Структура зберігання даних

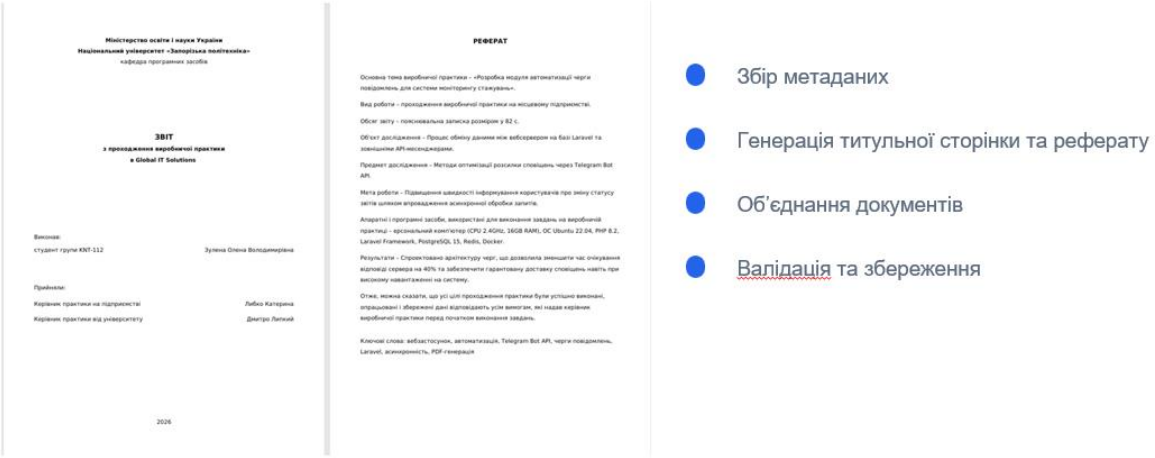
- **PostgreSQL:** Таблиці користувачів, ролей, завдань та звітів.
- **Laravel File Storage:** Зберігання шаблонів оформлення та PDFзвітів.
- **Local Storage:** Токени авторизації та чернетки звітів.



8

Рисунок В.8 – Слайд №8

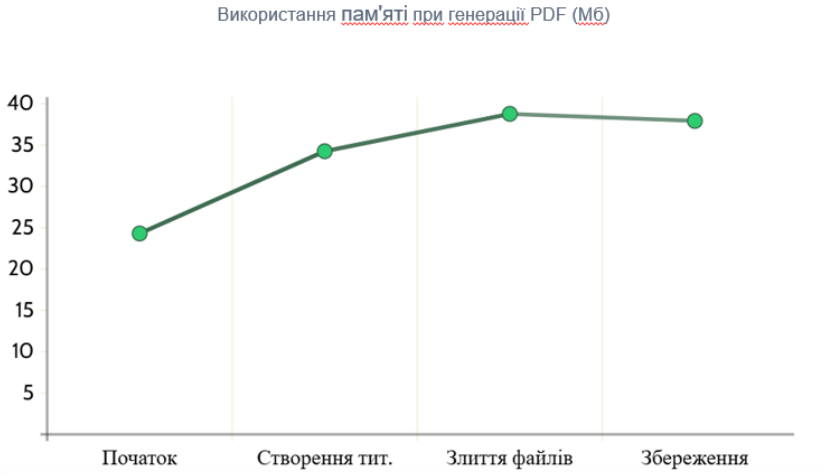
Процес генерації документа



9

Рисунок В.9 – Слайд №9

Аналіз споживання ресурсів



10

Рисунок В.10 – Слайд №10

Рисунок В.11 – Слайд №11

Рисунок В.12 – Слайд №12

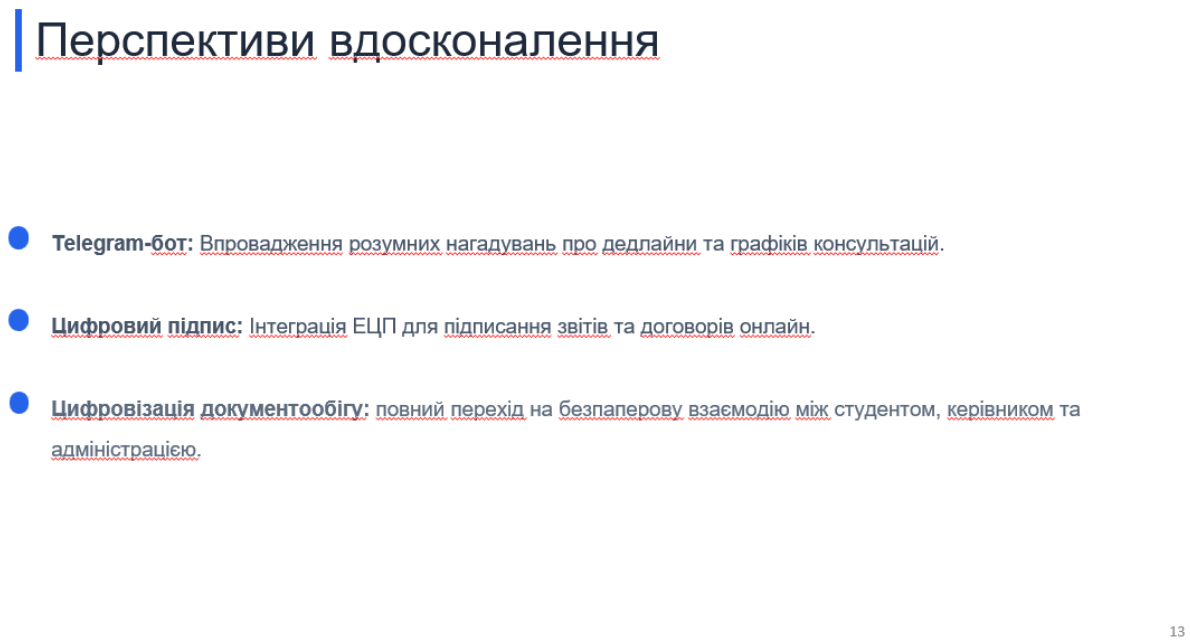


Рисунок В.13 – Слайд №13