

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”**

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**До захисту допущено:
В.о. завідувача кафедри
Артем ВОЛОКИТА**

_____ (підпис)

“ _ ” _____ 2026 р.

**Дипломний проєкт
на здобуття ступеня бакалавра
за освітньо-професійною програмою “Інженерія програмного
забезпечення комп’ютерних систем”
спеціальності 121 “Інженерія програмного забезпечення”**

на тему: Веб-сервіс для управління дипломними проєктами

Виконав : студент 4 курсу, групи ІМ-21
(шифр групи)

_____ Хвиць Максим Сергійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник _____ ас. кафедри ОТ, Ковальчук Олександр Мирович _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант (нормоконтроль) ст. викл., док. філос. Гончаренко О.О. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ ас. кафедри ІСТ, PhD Омельченко О. О. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному
проєкті немає запозичень з праць інших
авторів без відповідних посилань.

Студент _____
(підпис)

Київ - 2026 р.

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”**

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти - перший (бакалавр)

Освітньо-професійна програма

“Інженерія програмного забезпечення комп’ютерних систем”

спеціальності 121 “Інженерія програмного забезпечення”

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

Артем ВОЛОКИТА

(підпис)

“ ” _____ 2026 р.

ЗАВДАННЯ

на бакалаврський дипломний проєкт студента

Хвиця Максима Сергійовича

1. Тема проєкту Веб-сервіс для управління дипломними проєктами
керівник проєкту Ковальчук Олександр Миронович, асистент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від 03 червня 2026 №2055-с

2. Термін здачі студентом закінченого проєкту 9 червня 2026 року

3. Вихідні дані до проєкту технічна документація, теоретичні дані.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)

Розділ 1. Дослідження предметної області та визначення вимог до системи

Розділ 2. Обґрунтування вибору архітектурного підходу та технологічного стеку

Розділ 3. Проєктування та реалізація веб-сервісу

Розділ 4. Оцінювання результатів роботи та аналіз ефективності розробленої системи

5. Перелік графічного матеріалу (з точним позначенням обов'язкових креслень) структурна схема системи, функціональна схема (діаграма класів), алгоритм дій програмного забезпечення.

6. Консультанта проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Гончаренко О. О.		

7. Дата видачі завдання «12» січня 2026 р.

Календарний план

№ п/п	Найменування етапів дипломного проєкту	Терміни виконання етапів проєкту	Примітки
1.	<i>Затвердження теми проєкту</i>	<i>14.12.2025 - 12.01.2026</i>	
2.	<i>Вивчення та аналіз завдання</i>	<i>13.01.2026 - 27.01.2026</i>	
3.	<i>Розробка архітектури та загальної структури системи</i>	<i>28.01.2026 - 25.02.2026</i>	
4.	<i>Розробка структур окремих Підсистем</i>	<i>26.02.2026 - 27.03.2026</i>	
5.	<i>Програмна реалізація системи</i>	<i>28.03.2026 - 19.04.2026</i>	
6.	<i>Оформлення пояснювальної записки</i>	<i>21.04.2026 - 17.05.2026</i>	
7.	<i>Захист програмного продукту</i>	<i>22.04.2026</i>	
8.	<i>Передзахист</i>	<i>01.06.2026</i>	
9.	<i>Захист</i>	<i>16.06.2026</i>	

Студент-дипломник _____ Максим ХВИЦЬ
(підпис)

Керівник проєкту _____ Олександр КОВАЛЬЧУК
(підпис)

АНОТАЦІЯ

У процесі вивчення предметної сфери виконано аналіз, як у закладах вищої освіти організовується виконання дипломних проєктів, і які є основні проблеми взаємодії між студентами, керівниками та кафедрою. Було проаналізовано контроль виконання робіт, роботу з документами та зберігання інформації. У результаті розроблено вебсервіс для управління дипломними проєктами, яка дає змогу працювати зі студентами, темами, файлами, календарем і повідомленнями. Система автоматизує процеси, спрощує взаємодію між учасниками та допомагає краще контролювати виконання робіт. Програму створено з використанням сучасних вебтехнологій, з можливістю майбутнього розвитку.

Ключові слова: вебсервіс, дипломний проєкт, управління дипломними роботами, студент, керівник, кафедра, база даних.

ANNOTATION

In the process of studying the subject area, an analysis was performed of how the implementation of diploma projects is organized in higher education institutions, and what are the main problems of interaction between students, supervisors and the department. The control of work performance, work with documents and information storage were analyzed. As a result, a web service for managing diploma projects was developed, which allows working with students, topics, files, calendar and messages. The system automates processes, simplifies interaction between participants and helps to better control the implementation of work. The program was created using modern web technologies, with the possibility of future development.

Keywords: web service, diploma project, diploma theses management, student, supervisor, department, database.

Справки	Формат	Значення			Найменування	Кіл. листів	№ екземпляр	Додаток
					Документація загальна			
					Знову розроблена			
	A4	ІАЛЦ.467200.002 ТЗ			Веб-сервіс для управління	4		
					дипломними проектами			
					Технічне завдання			
	A4	ІАЛЦ.467200.003 ПЗ			Веб-сервіс для управління	122		
					дипломними проектами			
					Пояснювальна записка			
	A4	ІАЛЦ.467200.004 Д1			Веб-сервіс для управління	1		
					дипломними проектами			
					Структурна схема			
					системи			
	A4	ІАЛЦ.467200.005 Д2			Веб-сервіс для управління	1		
					дипломними проектами			
					Функціональна схема			
					системи			
	A4	ІАЛЦ.467200.006 Д3			Веб-сервіс для управління	1		
					дипломними проектами			
					Принципова схема			
					системи			
	A4	ІАЛЦ.467200.007 Д4			Веб-сервіс для управління	14		
					дипломними проектами			
					Фрагменти програмного			
					коду			
					ІАЛЦ.467200.001 ОА			
Зм	Лист	№ докум.		Підп				
Дата								
Розроб		Хвиць М. С.			Веб-сервіс для управління дипломними проектами Опис альбому	Літ.	Аркуш	Аркушів
Перев		Ковальчук О. М.					1	1
						КПІ ім. Ігоря Сікорського ФІОТ ІМ-21		

ТЕХНІЧНЕ ЗАВДАННЯ
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «Веб-сервіс для управління дипломними проектами»

Київ - 2026

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ	2
2 ПІДСТАВИ ДЛЯ РОЗРОБКИ	2
3 МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ.....	2
4 ДЖЕРЕЛА РОЗРОБКИ.....	3
5 ТЕХНІЧНІ ВИМОГИ.....	3
5.1. Вимоги до розробленого продукту	3
5.2. Вимоги до програмного забезпечення	3
5.3. Вимоги до апаратної частини	4
6 ЕТАПИ РОЗРОБКИ	4

					ІАЛЦ.467200.002 ТЗ		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Хвиць М. С.			Веб-сервіс для управління дипломними проектами Технічне завдання	Літ.	Аркуш
Перевірів		Ковальчук О. М.					Аркушів
Реценз.		Омельченко В. В.				1	4
Н. Контр.		Гончаренко О. О.				КП ім. Ігоря Сікорського ФІОТ, ІМ-21	
Затвердив		Волокита А. М.					

1 НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

У цьому технічному завданні наведено перелік вимог до розробки веб-сервісу для управління дипломними проєктами, а також опис основних умов його створення, впровадження, подальшої підтримки та можливого розширення функціональності.

Областю застосування веб-сервісу є організація та супровід процесу виконання дипломних робіт у закладах вищої освіти. Система може використовуватися студентами, керівниками та кафедрою, з метою зручного обміну інформацією, контролю етапів виконання проєкту, управління темами, документами, дедлайнами та іншими навчально-організаційними процесами.

2 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Вихідною основою для створення програмного продукту є індивідуальне завдання кваліфікаційної роботи бакалавра за напрямом “Інженерія програмного забезпечення”, яке було офіційно затверджене кафедрою обчислювальної техніки факультету інформатики та обчислювальної техніки Національного технічного університету України “КПІ імені Ігоря Сікорського”.

3 МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою розробки є створення веб-сервісу, який дозволяє організувати керування дипломними проєктами в межах єдиної системи, забезпечуючи зберігання та обробку даних про студентів, керівників, тематику робіт, супровідні документи, календарні події та етапи виконання, а також інші процеси, пов’язані з навчальним проєктуванням.

Призначенням системи є спрощення процесу організації дипломного проєктування, зменшення кількості ручних адміністративних дій, покращення контролю за виконанням завдань і забезпечення зручної комунікації між учасниками освітнього процесу.

					ІАЛЦ.467200.002 ТЗ	Арк.
						2
Зм.	Арк.	№ докум.	Підпис	Дата		

4 ДЖЕРЕЛА РОЗРОБКИ

Підґрунтям для розробки системи є нормативні матеріали щодо організації освітнього процесу, методичні рекомендації до виконання кваліфікаційних робіт, офіційна документація використаних технологій, статті та практичні матеріали, пов'язані з розробкою веб-сервісів, баз даних, клієнт-серверних та інформаційних систем.

5 ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробленого продукту

Розроблений веб-сервіс повинен задовольняти такі критерії:

- забезпечувати інтуїтивний, зручний та адаптивний інтерфейс користувача;
- забезпечувати реєстрацію, авторизацію та розмежування прав доступу для різних ролей користувачів;
- надавати можливість студентам переглядати інформацію про дипломний проєкт, керівника, етапи виконання та важливі події;
- забезпечувати науковим керівникам доступ до інформації про закріплених студентів і стан виконання їхніх дипломних робіт;
- надавати представникам кафедри можливість керувати студентами, керівниками, темами, документами, дедлайнами та захистами;
- підтримувати завантаження, збереження та перегляд документів, пов'язаних із дипломним проєктом;
- забезпечувати відображення календарних подій, нагадувань і важливих строків виконання;
- забезпечувати збереження даних у базі даних і коректну обробку запитів користувачів;
- мати можливість подальшого розширення функціональності.

5.2. Вимоги до програмного забезпечення

Для розробки та запуску веб-сервісу необхідне таке програмне забезпечення:

					ІАЛЦ.467200.002 ТЗ	Арк.
						3
Зм.	Арк.	№ докум.	Підпис	Дата		

- операційна система Windows, macOS або Linux;
- Node.js (з менеджером пакетів npm);
- MongoDB;
- сучасні веббраузери (Chrome, Firefox, Edge);
- середовище програмування Visual Studio Code;

5.3. Вимоги до апаратної частини

Мінімальні вимоги до апаратного забезпечення:

- ЦП: рівня не нижче середнього класу (аналог Intel Core i3);
- RAM: не менше 4 ГБ;
- ROM: не менше 10 ГБ вільного дискового простору для розміщення проєкту, залежностей та даних системи;
- підключення до Інтернету: стабільне мережеве з'єднання для встановлення залежностей, роботи з веб-сервісом та зовнішніми ресурсами.

6 ЕТАПИ РОЗРОБКИ

Назва етапів виконання	Термін виконання
Затвердження теми роботи	14.12.2025 - 12.01.2026
Вивчення та аналіз завдання	13.01.2026 - 27.01.2026
Розробка архітектури та загальної структури системи	28.01.2026 - 25.02.2026
Розробка структур окремих частин системи	26.02.2026 - 27.03.2026
Програмна реалізація системи	28.03.2026 - 19.04.2026
Виправлення помилок	21.04.2026 - 17.05.2026
Оформлення пояснювальної записки	14.12.2025 - 12.01.2026

**ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО ДИПЛОМНОГО ПРОЄКТУ**

на тему: «Веб-сервіс для управління дипломними проектами»

Київ - 2026

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ.....	4
ВСТУП	5
РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИЗНАЧЕННЯ ВИМОГ ДО СИСТЕМИ	7
1.1 Особливості організації процесу виконання дипломних проєктів у закладах вищої освіти.....	7
1.2 Основні учасники процесу управління дипломними роботами та їхні функції.....	10
1.3 Аналіз проблем традиційного супроводу дипломних проєктів.....	15
1.4 Огляд існуючих цифрових рішень для управління освітніми та проєктними процесами.....	19
1.5 Формування функціональних і нефункціональних вимог до веб-сервісу	22
ВИСНОВКИ ДО РОЗДІЛУ	29
РОЗДІЛ 2. ОБҐРУНТУВАННЯ ВИБОРУ АРХІТЕКТУРНОГО ПІДХОДУ ТА ТЕХНОЛОГІЧНОГО СТЕКУ	30

					ІАЛЦ.467200.003 ПЗ				
Зм.	Арк.	№ докум.	Підпис	Дата	Веб-сервіс для управління дипломними проектами Пояснювальна записка	Літ.	Аркуш	Аркушів	
Розробив	Хвиць М. С.								
Перевірив	Ковальчук О. М.						1	123	
Реценз.	Омельченко В. В.					КПІ ім. Ігоря Сікорського ФІОТ, ІМ-21			
Н. Контр.	Гончаренко О. О.								
Затвердив	Волокита А. М.								

2.1 Вибір архітектурної моделі для розробки веб-сервісу.....	30
2.2 Обґрунтування використання клієнт-серверної архітектури	34
2.3 Вибір технологій для реалізації клієнтської частини системи.....	37
2.4 Вибір технологій для розробки серверної частини веб-сервісу.....	42
2.5 Обґрунтування вибору системи керування базами даних.....	46
2.6 Засоби забезпечення безпеки, автентифікації та розмежування прав доступу	50
ВИСНОВКИ ДО РОЗДІЛУ	55
РОЗДІЛ 3. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБ-СЕРВІСУ	56
3.1 Мета, призначення та основні завдання розроблюваної системи.....	56
3.2 Проєктування структури та користувацької взаємодії	58
3.3 Розробка бази даних та опис основних сутностей системи	62
3.4 Реалізація функціоналу всіх користувачів системи	68
3.5 Реалізація механізмів управління дипломними темами, документами та дедлайнами.....	90
3.6 Розробка інтерфейсу користувача та адаптація системи для зручної роботи.....	93
ВИСНОВКИ ДО РОЗДІЛУ	98
РОЗДІЛ 4. ОЦІНЮВАННЯ РЕЗУЛЬТАТІВ РОБОТИ ТА АНАЛІЗ ЕФЕКТИВНОСТІ РОЗРОБЛЕНОЇ СИСТЕМИ	99

4.1 Перевірка працездатності основних модулів веб-сервісу	99
4.2 Аналіз зручності використання інтерфейсу системи	103
4.3 Оцінювання ефективності автоматизації процесу управління дипломними проектами.....	107
4.4 Аналіз переваг розробленого веб-сервісу порівняно з традиційним підходом.....	111
4.5 Перспективи подальшого розвитку та вдосконалення системи	114
ВИСНОВКИ ДО РОЗДІЛУ	117
ВИСНОВКИ.....	118
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	120

ПЕРЕЛІК СКОРОЧЕНЬ

API	(Application Programming Interface) Спосіб взаємодії між частинами програмного забезпечення для обміну даними
СКБД	(Система керування базами даних) Програмне забезпечення для роботи з інформацією в базах даних
UI	(User Interface) Інтерфейс для взаємодії користувача з програмою
REST	(Representational State Transfer) Спосіб побудови API для обміну даними через веб-запити
HTTP	(HyperText Transfer Protocol) Протокол для обміну даними у вебі
HTTPS	(HyperText Transfer Protocol Secure) HTTP із додатковим захистом переданої інформації
JWT	(JSON Web Token) Токен для автентифікації та авторизації користувачів
JSON	(JavaScript Object Notation) Спосіб запису даних у структурованому вигляді для обміну між системами
WebSocket	(WebSocket Protocol) Механізм постійного двостороннього обміну даними між клієнтом і сервером у режимі реального часу
2FA	(Two-Factor Authentication) Механізм безпеки, який вимагає двоетапної перевірки користувача.

ВСТУП

У сучасній освіті дедалі більше процесів переходить у цифрове середовище, тому виникає потреба у системах, які спрощують та автоматизують організаційні задачі у закладах вищої освіти. Одним із таких напрямів є супровід дипломного проєктування, який охоплює вибір і погодження теми, призначення наукового керівника, контроль етапів виконання роботи, обмін документами, дотримання дедлайнів та взаємодію між освітніми учасниками.

Як показує аналіз, у багатьох закладах вищої освіти ці завдання часто вирішуються за допомогою розрізнених інструментів - електронної пошти, месенджерів, таблиць, окремих документів та усних домовленостей. Через це дані зберігаються в різних місцях, складніше контролювати їх актуальність, а інформація може дублюватися або втрачатися. У результаті виникає потреба в єдиному цифровому середовищі, яке б об'єднувало всі етапи супроводу дипломних проєктів.

У межах дипломної роботи здійснено проєктування та реалізацію інформаційної системи, призначеної для супроводу процесу виконання дипломних проєктів. Розроблене рішення підтримує взаємодію різних типів користувачів освіти, зокрема студентів, керівників, а також представників адміністрації кафедри й університету, забезпечуючи кожному з них відповідний набір функцій відповідно до ролі в системі.

Метою дипломної роботи є проєктування та розробка веб-сервісу для управління дипломними проєктами, який забезпечує централізоване зберігання даних, контроль виконання етапів роботи, підтримку взаємодії між учасниками освітнього процесу та автоматизацію окремих адміністративних дій.

					ІАЛЦ.467200.003 ПЗ	Арк.
						5
Зм.	Арк.	№ докум.	Підпис	Дата		

У якості об'єкта дослідження розглядається процес організації та супроводу дипломного проєктування у закладах вищої освіти. Предметом - методи, програмні засоби та архітектурні підходи до створення подібних інформаційних систем.

Для досягнення поставленої мети було визначено наступні завдання:

1. Виконати аналіз процесу управління дипломними проєктами.
2. Визначити основні проблеми традиційного супроводу дипломних робіт.
3. Сформувати вимоги до веб-сервісу, включаючи функціональні та нефункціональні аспекти.
4. Пояснити доцільність обраної архітектури та технологічного стеку.
5. Сформувати структуру даних і логіку взаємодії користувачів.
6. Виконати розробку фронтенд- і бекенд-частин веб-сервісу.
7. Перевірити працездатність основних модулів системи.
8. Проаналізувати ефективність розробленого рішення.

Розроблений веб-сервіс має практичну цінність, оскільки його можна використовувати для більш зручної організації процесу дипломного проєктування. Система дозволяє зменшити обсяг ручної адміністративної роботи, забезпечити централізований доступ до даних, покращити контроль термінів виконання та спростити обмін інформації між учасниками освітнього процесу.

Під час розробки використано клієнт-серверний підхід, за якого клієнтська частина забезпечує взаємодію користувача із системою, а серверна - обробку запитів, керування даними та контроль доступу. Інформація про користувачів, проєкти, документи й події зберігається у базі даних.

Отже, розробка веб-сервісу для управління дипломними проєктами є актуальною, оскільки спрямована на автоматизацію процесу дипломного проєктування та покращення взаємодії між студентами, керівниками й кафедрою.

					ІАЛЦ.467200.003 ПЗ	Арк.
						6
Зм.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИЗНАЧЕННЯ ВИМОГ ДО СИСТЕМИ

1.1 Особливості організації процесу виконання дипломних проєктів у закладах вищої освіти

Дипломне проєктування є фінальним етапом підготовки здобувача вищої освіти. На цьому етапі студент має показати не лише теоретичні знання, а й уміння застосовувати їх на практиці: аналізувати предметну область, приймати технічні рішення та реалізовувати програмний продукт відповідно до вимог.

Процес виконання над дипломом відбувається поетапно. Спочатку здійснюється вибір теми, погодження її з науковим керівником та затвердження на рівні кафедри. Після цього виконується планування роботи, визначаються основні завдання, термін виконання, очікувані результати та перелік необхідних матеріалів. Надалі студент виконує розробку, готує матеріали, узгоджує проміжні результати з керівником і подає матеріали на перевірку.

У межах веб-сервісу цей процес було формалізовано через набір сутностей, пов'язаних із користувачами, дипломними проєктами, етапами виконання, документами, календарними подіями та нагадуваннями. Таким чином, процес дипломного проєктування стає більш структурованим і зрозумілим, а виконання роботи - контрольованим на кожному етапі.

Основні етапи виконання дипломного проєкту та їх відображення у функціональності системи подано в таблиці 1.1. Для кожного етапу визначено відповідні програмні механізми, що забезпечують підтримку процесу дипломного проєктування в межах інформаційної системи.

					ІАЛЦ.467200.003 ПЗ	Арк.
						7
Зм.	Арк.	№ докум.	Підпис	Дата		

Таблиця 1.1 - Етапи виконання дипломного проєкту та їх відображення у веб-сервісі

Етап дипломного проєктування	Зміст етапу	Відображення у веб-сервісі
Вибір теми	Формування або вибір теми дипломної роботи	Створення теми проєкту та її подальше погодження
Призначення керівника	Закріплення студента за науковим керівником	Зв'язок між студентом, керівником і проєктом
Планування роботи	Визначення етапів, підзавдань і строків	Використання етапів, дедлайнів і календарних подій
Виконання завдань	Реалізація практичної частини та підготовка матеріалів	Ведення прогресу, підзавдань, коментарів і ресурсів
Перевірка матеріалів	Аналіз документів керівником або кафедрою	Завантаження документів, статуси перевірки, коментарі
Підготовка до захисту	Формування фінальних матеріалів і організація захисту	Події типу захисту, посилання на матеріали, комісії

Як показано в таблиці 1.1, кожен етап дипломного проєктування має відповідне відображення у веб-сервісі. Процес вибору теми реалізується через створення та погодження проєкту, етап планування - через систему дедлайнів і

календарних подій, а перевірка матеріалів - через механізми завантаження документів і контролю статусів.

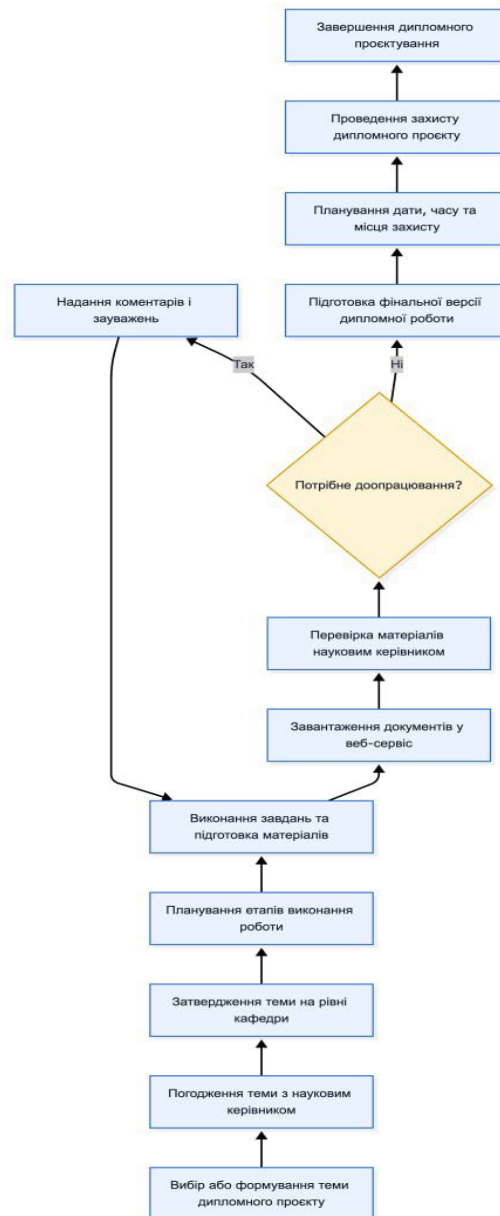


Рисунок 1.1 - Узагальнена схема життєвого циклу дипломного проєкту від вибору теми до захисту

Як видно з рисунка 1.1, життєвий цикл дипломного проєкту включає етапи формування теми, планування роботи, виконання завдань, перевірки матеріалів і підготовки до захисту. На кожному з етапів між учасниками освітнього процесу відбувається обмін документами, коментарями та інформацією про стан виконання роботи.

Процес дипломного проєктування передбачає постійну взаємодію між студентом, науковим керівником та працівниками кафедри. Через це система повинна підтримувати контроль доступу, обмін повідомленнями та відстеження стану виконання роботи.

У проєкті така логіка реалізується через рольову модель користувачів. Передбачено ролі студента, наукового керівника, адміністратора кафедри, університету та суперадміністратора. Це дає змогу обмежити доступ до функцій відповідно до обов'язків користувача та забезпечити керованість процесу.

Короткий висновок: організація дипломного проєктування є досить складним процесом, який потребує централізованого управління, контролю виконання завдань, збереження документів і підтримки комунікації між учасниками. Саме ці особливості визначають необхідність створення спеціалізованого веб-сервісу.

1.2 Основні учасники процесу управління дипломними роботами та їхні функції

У процесі управління дипломними роботами беруть участь різні категорії користувачів, кожна з яких виконує окремі функції. Для забезпечення коректної роботи системи необхідно визначити повноваження кожної ролі та обмежити доступ до функціоналу відповідно до її завдань.

У веб-сервісі це реалізовано через рольову модель, у якій кожен користувач має свій тип облікового запису. Такий підхід дає змогу відокремити функціонал студента від функціоналу керівника або адміністратора, а також забезпечити контроль доступу до даних дипломних проєктів.

Основні ролі користувачів системи, їхні функції та відповідні функціональні можливості веб-сервісу наведено в таблиці 1.2. Рольова модель використовується для організації взаємодії між учасниками освітнього процесу.

					ІАЛЦ.467200.003 ПЗ	Арк.
						10
Зм.	Арк.	№ докум.	Підпис	Дата		

Таблиця 1.2 - Рольова модель користувачів системи та їхні функціональні можливості

Учасник процесу	Основні функції	Функціональне відображення в системі
Студент	Виконання дипломного проєкту, завантаження матеріалів, перегляд дедлайнів	Особистий кабінет, проєкт, етапи, документи, календар, профіль
Науковий керівник	Контроль виконання роботи, надання коментарів, перевірка документів	Список студентів, перегляд проєктів, коментарі, статуси перевірки, нагадування
Адміністратор кафедри	Організація роботи кафедри, контроль студентів, керівників і захистів	Панель кафедри, управління користувачами, темами, дедлайнами, комісіями
Адміністратор університету	Загальний контроль структурних одиниць і політик	Адміністративні інструменти рівня університету
Суперадміністратор	Технічне та системне адміністрування	Керування глобальними налаштуваннями, безпекою та системними процесами

Як показано в таблиці 1.2, кожна роль у системі має визначений набір функцій та рівень доступу до інформації. Для візуалізації взаємодії основних користувачів із функціоналом веб-сервісу було побудовано UML-діаграму варіантів використання, наведену на Рисунок 1.2.



Рисунок 1.2 - UML-діаграма варіантів використання для ролей студента, керівника та адміністратора кафедри

На рисунку 1.2 - UML-діаграма доповнює рольову модель, подану в таблиці 1.2, та відображає взаємозв'язок між ролями користувачів і функціональними можливостями веб-сервісу.

Студент у системі розглядається як основний виконавець дипломного проєкту. Для нього важливими є можливість переглядати інформацію про власну тему, стежити за дедлайнами, працювати з етапами виконання, завантажувати документи та отримувати зворотний зв'язок від керівника. Додатково може зберігатися інформація про профіль студента, контактні дані, навчальну групу, спеціальність та інші відомості, необхідні для супроводу роботи.

Науковий керівник виконує функцію контролю та методичного супроводу. У системі для нього передбачається доступ до студентських проєктів, можливість переглядати прогрес, залишати коментарі, перевіряти

документи, створювати нагадування та брати участь у плануванні подій. Такий функціонал дозволяє зменшити кількість неформальної комунікації та зосередити основні дані про роботу в одному середовищі.

Адміністратор кафедри відповідає за організаційну складову процесу. До його функцій належать управління студентами, керівниками, дипломними темами, академічними дедлайнами, документами, подіями та захистами. У контексті розробленої системи ця роль є важливою для централізованого контролю стану дипломного проектування на рівні кафедри.

Для деталізації механізму розмежування доступу між користувачами було сформовано матрицю прав доступу, наведену в таблиці 1.3. Вона визначає можливості кожної ролі щодо роботи з основними функціями системи.

Таблиця 1.3 - Матриця розмежування прав доступу користувачів до функцій системи

Об'єкт управління	Студент	Керівник	Адміністратор кафедри
Перегляд власного проєкту	Так	Так, для закріплених студентів	Так, у межах кафедри
Редагування теми	Так	Ні	Так
Завантаження документів	Так	Так	Так
Перевірка документів	Ні	Так	Так
Організація захистів	Ні	Частково	Так

Кінець таблиці 1.3

Об'єкт управління	Студент	Керівник	Адміністратор кафедри
Створення нагадувань	Обмежено (через календарні події у власному проєкті)	Так	Так
Управління користувачами	Ні	Обмежено (лише щодо закріплених студентів)	Так

Як видно з таблиці 1.3, функціональні можливості користувачів відрізняються залежно від їхньої ролі в системі. Такий підхід забезпечує безпечну роботу з даними, підтримує централізоване керування дипломними проєктами та спрощує адміністрування веб-сервісу.

Рольова модель є основою для подальшого проєктування вимог до веб-сервісу, оскільки саме від функцій користувачів залежить структура інтерфейсу, маршрути серверної частини, модель доступу до даних і логіка взаємодії між модулями.

Короткий висновок: основними учасниками процесу управління дипломними роботами є студент, науковий керівник і адміністратор кафедри. Для ефективної роботи системи необхідно забезпечити розподіл функцій між ролями, контроль доступу до даних і підтримку взаємодії між учасниками.

1.3 Аналіз проблем традиційного супроводу дипломних проєктів

Традиційний супровід дипломних проєктів часто здійснюється за допомогою окремих інструментів, які не утворюють єдиної інформаційної системи. Для комунікації можуть використовуватися електронна пошта або месенджери, для збереження документів - локальні папки чи хмарні сховища, для обліку студентів - таблиці, а для контролю дедлайнів - календарі або текстові списки. У результаті дані розподіляються між різними середовищами, що ускладнює контроль актуального стану дипломних робіт.

Однією з основних проблем є відсутність централізованого збереження інформації. Якщо тема, документи, коментарі, терміни виконання та статуси перевірки зберігаються в різних місцях, збільшується ймовірність втрати даних або використання застарілої інформації. Це особливо критично для кафедри, яка має контролювати не один проєкт, а велику кількість робіт одночасно. Узагальнення основних недоліків традиційного підходу та відповідних вимог до інформаційної системи подано в таблиці 1.4.

Таблиця 1.4 - Аналіз проблем традиційного супроводу дипломних проєктів та визначення вимог до веб-сервісу

Проблема традиційного підходу	Наслідки	Вимога до веб-сервісу
Розрізнене зберігання даних	Втрата актуальності інформації, дублювання файлів	Централізована база даних і єдиний кабінет користувача

Кінець таблиці 1.4

Проблема традиційного підходу	Наслідки	Вимога до веб-сервісу
Відсутність прозорого контролю прогресу	Складність оцінювання стану виконання роботи	Етапи, підзавдання, статуси та відсоток прогресу
Неузгоджена комунікація	Втрата коментарів і домовленостей	Коментарі до етапів, документів і завдань
Ручний контроль дедлайнів	Пропущені терміни виконання	Календарні події, нагадування та дедлайни
Обмежена видимість для кафедри	Ускладнення адміністрування дипломних робіт	Панель адміністратора кафедри
Відсутність розмежування доступу	Ризик несанкціонованого перегляду або зміни даних	Рольова модель і перевірка прав доступу

Для кращого розуміння різниці між традиційним підходом та централізованим веб-сервісом було побудовано порівняльну схему. На рисунку 1.3 показано, як традиційний підхід ускладнює контроль процесу, тоді як веб-сервіс об'єднує всі дані в єдину систему.

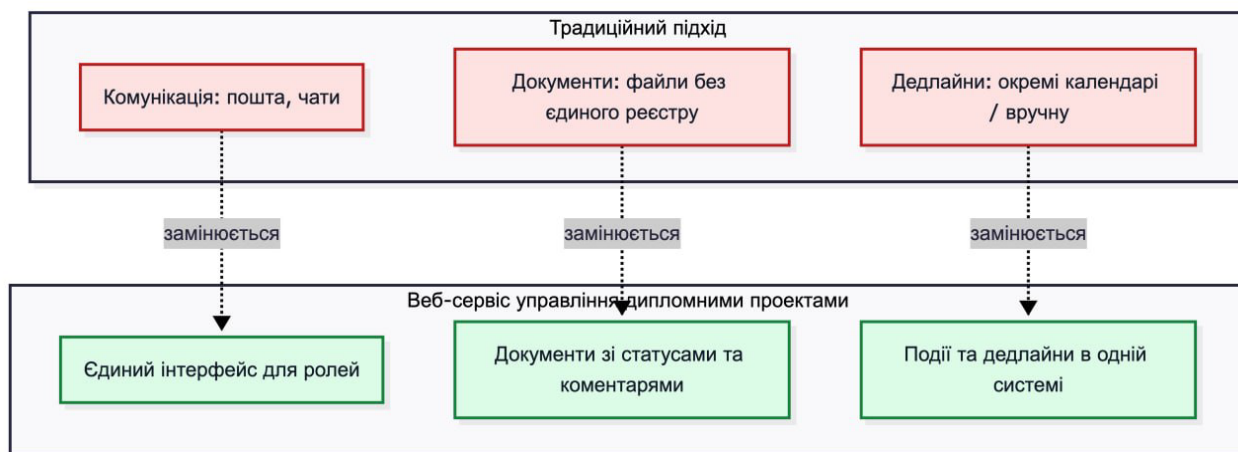


Рисунок 1.3 - Порівняльне представлення традиційного та централізованого супроводу дипломного проєкту

Наступною проблемою є складність відстеження прогресу виконання дипломної роботи. У традиційному підході керівник часто отримує інформацію про стан роботи лише під час консультацій. У такому випадку неможливо оперативно виявити затримки, визначити проблемні етапи або проконтролювати виконання підзавдань.

У створеній системі ця проблема вирішується шляхом представлення дипломного проєкту як структурованої сутності, що містить тему, опис, етапи, підзавдання, дедлайни, документи, коментарі та календарні події. Такий підхід дозволяє перейти від неформального контролю до формалізованого моніторингу стану роботи.

Окрему проблему становить управління документами. Під час виконання дипломного проєкту студент готує різні версії пояснювальної записки, презентації, додатків, звітів та інших важливих матеріалів. Якщо ці файли передаються через месенджери або електронну пошту, складно визначити актуальну версію документа, історію перевірки та зміст зауважень. У веб-сервісі для цього має бути передбачено збереження документів із типом, статусом, версією, коментарями та датою завантаження.

Додатково для систематизації підходів до контролю дипломних проєктів наведено таблицю 1.5, у якій виконано порівняння традиційного та автоматизованого способів організації процесу.

Таблиця 1.5 - Порівняльна характеристика традиційного та автоматизованого підходів до контролю дипломних проєктів

Об'єкт контролю	Традиційний підхід	Підхід у веб-сервісі
Тема дипломної роботи	Зберігається у документах або таблицях	Зберігається як окрема сутність проєкту
Дедлайни	Контролюються вручну	Фіксуються у календарі та етапах
Документи	Передаються через пошту або месенджери	Завантажуються в систему зі статусами перевірки
Коментарі керівника	Розміщуються у повідомленнях або файлах	Прив'язуються до етапів, завдань або документів
Прогрес роботи	Оцінюється неформально	Визначається через статуси, підзавдання та відсоток виконання
Адміністративний контроль	Виконується через окремі таблиці	Здійснюється через панель адміністратора

Також важливою є проблема розмежування відповідальності. Без інформаційної системи складно визначити, хто створив подію, хто оновив завдання, хто перевіряв документ або хто змінив статус. Для освітнього процесу це має значення, оскільки дії користувачів повинні бути контрольованими та прив'язаними до відповідних ролей. У програмній реалізації це враховується

через збереження ідентифікаторів користувачів і ролей, а також дат створення чи оновлення для окремих елементів проєкту.

Короткий висновок: традиційний супровід дипломних проєктів має певні недоліки, які пов'язані зі зберіганням даних, ручним контролем термінів, неформальною комунікацією та відсутністю прозорого моніторингу прогресу. Для вирішення цих проблем доцільним є використання спеціалізованого веб-сервісу з централізованою базою даних і рольовою моделлю доступу.

1.4 Огляд існуючих цифрових рішень для управління освітніми та проєктними процесами

Для визначення доцільності розробки веб-сервісу управління дипломними проєктами було розглянуто цифрові рішення, які можуть частково використовуватися для організації освітнього процесу. До таких рішень належать системи дистанційного навчання, сервіси керування завданнями та хмарні сховища. Порівняння здійснювалося за ключовими критеріями: підтримка академічних ролей, погодження тем, контроль етапів і прогресу, робота з документами, календарне планування та централізоване адміністрування. Це дозволяє оцінити повний цикл супроводу диплому.

Системи дистанційного навчання, зокрема Moodle або Google Classroom, орієнтовані переважно на розміщення матеріалів, видачу завдань і перевірку робіт. Хоча вони дозволяють обмінюватися файлами та підтримувати комунікацію між викладачем і студентом, їх функціональність не орієнтована на комплексний супровід дипломного проєктування. У таких системах зазвичай відсутні інструменти для відстеження етапів виконання роботи, координації підготовки до захисту та управління дипломними проєктами на рівні кафедри.

Сервіси керування проєктами, наприклад Trello або Jira, надають інструменти для планування задач, створення дошок, призначення виконавців і

					ІАЛЦ.467200.003 ПЗ	Арк.
						19
Зм.	Арк.	№ докум.	Підпис	Дата		

контролю статусів. Проте ці системи є універсальними та не враховують специфіку освітнього процесу. Для використання в контексті дипломного проєктування вони потребують додаткового налаштування, ручного створення структур, окремої логіки для студентів, керівників та адміністраторів кафедри.

Хмарні сховища, такі як Google Drive або OneDrive, забезпечують збереження файлів і спільний доступ до документів. Водночас вони не вирішують завдання контролю прогресу, погодження тем, фіксації статусів перевірки документів. Тому такі сервіси можуть виконувати лише допоміжну функцію.

Для узагальнення результатів аналізу наведено таблицю 1.6, у якій подано порівняння основних типів існуючих рішень, їх призначення та обмежень у контексті використання для супроводу дипломних проєктів.

Таблиця 1.6 - Порівняльна характеристика існуючих цифрових рішень та їхніх обмежень для супроводу дипломних проєктів

Тип рішення	Приклади	Основне призначення	Обмеження для дипломного проєктування
Системи дистанційного навчання	Moodle, Google Classroom	Організація навчальних курсів, завдань і матеріалів	Відсутня спеціалізована модель дипломного проєкту, погодження тем і захистів
Внутрішні освітні системи	Електронні кабінети ЗВО	Облік студентів, оцінок і навчальних даних	Зазвичай не охоплюють повний цикл виконання дипломної роботи

Кінець таблиці 1.6

Тип рішення	Приклади	Основне призначення	Обмеження для дипломного проєктування
Сервіси керування проєктами	Trello, Jira	Планування задач і контроль виконання	Не враховано академічні ролі, кафедральний контроль і освітні регламенти
Хмарні сховища	Google Drive, OneDrive	Збереження та спільне використання файлів	Відсутні статуси перевірки, контроль дедлайнів і централізована аналітика
Засоби комунікації	Telegram, Microsoft Teams, Slack	Обмін повідомленнями між учасниками	Інформація розподілена між чатами, складно контролювати історію рішень

На основі проведеного огляду було визначено, що наявні рішення можуть закривати окремі потреби, але не забезпечують комплексного супроводу дипломного проєкту. Саме тому в межах роботи було обґрунтовано необхідність створення єдиного веб-сервісу, у якому поєднуються всі функції, які необхідні для супроводу. Ця логіка відображається через окремі ролі користувачів, сутність дипломного проєкту, документи зі статусами перевірки, календарні події, нагадування, коментарі та панелі для різних рівнів доступу.

Короткий висновок: існуючі цифрові рішення мають окремі корисні функції, однак не забезпечують повного циклу управління дипломом. Для вирішення цих завдань доцільно впровадити спеціалізовану інформаційну систему.

1.5 Формування функціональних і нефункціональних вимог до веб-сервісу

Після аналізу предметної області та існуючих цифрових рішень було сформовано вимоги до веб-сервісу для управління дипломними проектами. Вимоги до системи поділяються на дві категорії. Одна описує її функції та те, що вона повинна робити, інша - якість її роботи, тобто безпеку, стабільність, швидкість, зручність використання та потенціал розвитку в майбутньому.

Функціональні вимоги сформовано з урахуванням основних ролей користувачів: студента, наукового керівника, адміністратора кафедри, університету та суперадміністратора. Повний перелік функціональних вимог до веб-сервісу наведено в таблиці 1.7.

Таблиця 1.7 - Специфікація основних функціональних вимог до розроблюваного веб-сервісу

Код вимоги	Функціональна вимога	Призначення
F1	Керування обліковими записами та автентифікація користувачів	Створення облікових записів (за запрошенням або адміністратором) і доступ до системи лише для автентифікованих користувачів

Продовження таблиці 1.7

Код вимоги	Функціональна вимога	Призначення
F2	Розмежування прав доступу за ролями	Обмеження функцій відповідно до ролі користувача
F3	Керування профілями студентів і керівників	Збереження контактних, академічних та організаційних даних
F4	Створення та редагування дипломного проєкту	Ведення теми, опису, статусу та пов'язаних даних
F5	Погодження теми дипломної роботи	Подання теми студентом на перевірку та контроль статусу погодження теми на рівні кафедри
F6	Керування етапами та підзавданнями	Формалізація процесу виконання дипломної роботи
F7	Завантаження та перевірка документів	Робота з файлами, версіями, статусами і коментарями
F8	Календар подій і дедлайнів	Контроль консультацій, строків, захистів та інших подій
F9	Система нагадувань і повідомлень	Інформування користувачів про важливі зміни та події
F10	Адміністративна панель кафедри	Контроль студентів, керівників, тем, дедлайнів і захистів

Кінець таблиці 1.7

Код вимоги	Функціональна вимога	Призначення
F11	Підтримка коментарів і зворотного зв'язку	Забезпечення взаємодії між студентом і керівником
F12	Фіксація ключових подій і змін у системі (журнал подій/аудит)	Підвищення прозорості та контрольованості процесу (для адміністративних ролей)

Представлена таблиця відображає ключові функціональні можливості системи та демонструє, що кожна вимога спрямована на підтримку окремого аспекту управління дипломним проєктом - від автентифікації користувачів до адміністративного контролю та аудиту змін.

З метою узагальнення взаємозв'язку функціональних можливостей і ролей користувачів було побудовано діаграму, наведену на рисунку 1.4. Вона демонструє розподіл функцій системи між основними категоріями користувачів та дозволяє краще зрозуміти архітектуру доступу до функціоналу.

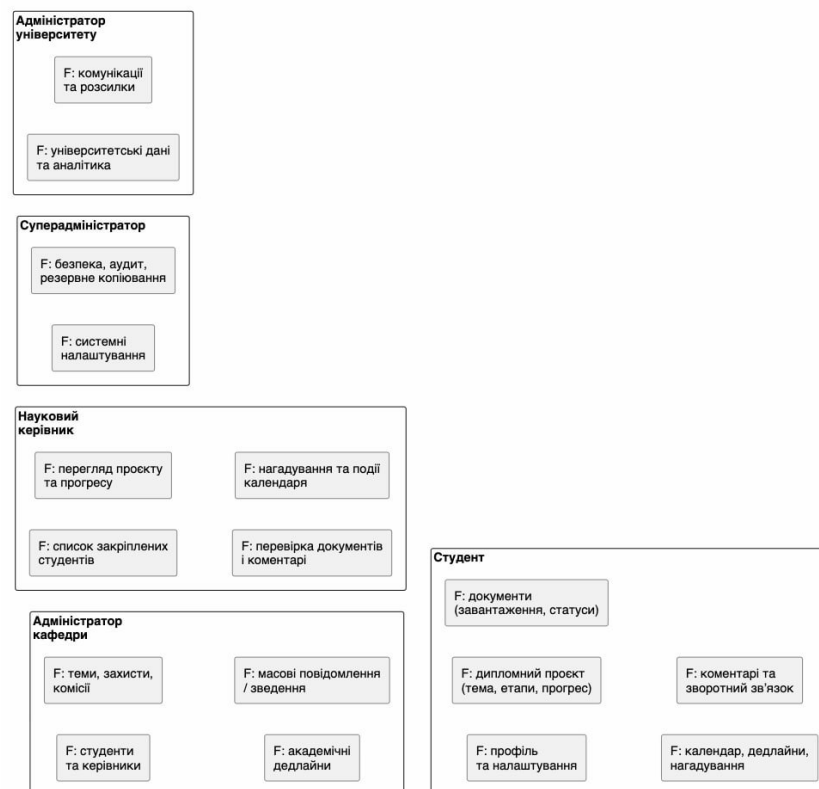


Рисунок 1.4 - Діаграма функціональних вимог веб-сервісу за ролями користувачів

Особлива увага приділяється сутності дипломного проєкту, адже вона має містити тему, опис, статус погодження, учасників, етапи, підзавдання, документи, події, нагадування та коментарі. Завдяки цьому дипломна робота розглядається як керований об'єкт, стан якого може бути відстежений упродовж усього життєвого циклу. Для деталізації розподілу функціональних можливостей між користувачами наведено таблицю 1.8.

Таблиця 1.8 - Характеристика основних ролей користувачів та їхніх функціональних можливостей

Роль користувача	Основні можливості
Студент	Перегляд власного проєкту, робота з етапами, завантаження документів, перегляд дедлайнів, отримання коментарів

Кінець таблиці 1.8

Роль користувача	Основні можливості
Науковий керівник	Перегляд закріплених студентів, контроль прогресу, перевірка документів, створення коментарів і нагадувань
Адміністратор кафедри	Керування студентами, керівниками, темами, дедлайнами, захистами та кафедральними процесами
Адміністратор університету	Контроль даних на рівні університету та робота з університетськими налаштуваннями
Суперадміністратор	Технічне адміністрування, глобальні налаштування та контроль системних процесів

Аналіз ролей показує, що кожен користувач має чітко визначені функції, що забезпечує розмежування доступу та дозволяє уникнути конфлікту прав.

Нефункціональні вимоги визначають якість роботи веб-сервісу та умови його експлуатації. Для даної системи особливе значення мають безпека даних, стабільність роботи й адаптивність інтерфейсу. Повний перелік нефункціональних вимог наведено в таблиці 1.9.

Таблиця 1.9 - Специфікація нефункціональних вимог до розроблюваного веб-сервісу та їх обґрунтування

Категорія	Нефункціональна вимога	Обґрунтування
Надійність	Збереження даних у централізованій базі	Зменшується ризик втрати інформації або дублювання

Кінець таблиці 1.9

Категорія	Нефункціональна вимога	Обґрунтування
Безпека	Використання автентифікації та перевірки прав доступу	Дані студентів і дипломних проєктів мають бути захищені від несанкціонованого доступу
Продуктивність	Швидка обробка типових запитів користувача	Система має забезпечувати стабільну роботу під час щоденного використання
Зручність	Інтерфейс має бути зрозумілим для різних категорій користувачів	Системою користуються студенти, керівники та адміністратори
Адаптивність	Підтримка роботи на різних розмірах екранів	Доступ до сервісу може здійснюватися з ноутбуків, планшетів і смартфонів
Розширюваність	Можливість додавання нових модулів	Система може розвиватися відповідно до потреб кафедри або університету
Супроводжуваність	Розділення клієнтської та серверної логіки	Спрощується підтримка, тестування та модифікація програмного коду

З урахуванням вимог до реалізації було визначено клієнт-серверну архітектуру веб-сервісу. Клієнтська частина в цій моделі забезпечує роботу з інтерфейсом користувача, а серверна - за обробку бізнес-логіки, роботу з базою

даних, автентифікацію та керування всіма процесами системи. Структуру основних компонентів системи та їх призначення наведено в таблиці 1.10.

Таблиця 1.10 - Основні компоненти архітектури веб-сервісу та їхнє функціональне призначення

Компонент системи	Основне призначення	Приклад реалізованої логіки
Клієнтська частина	Надання інтерфейсу користувача	Кабінети студента, керівника та адміністратора
Серверна частина	Обробка запитів і бізнес-логіки	Авторизація, перевірка доступу, робота з проєктами
База даних	Збереження структурованої інформації	Користувачі, дипломні проєкти, документи, події
Модуль повідомлень	Передача важливої інформації	Нагадування, коментарі, системні події
Модуль документів	Завантаження і перевірка файлів	Статуси документів, версії, коментарі

Аналіз архітектури показує, що система складається з взаємопов'язаних модулів інтерфейсу, бізнес-логіки, зберігання даних, повідомлень і роботи з документами.

Короткий висновок: визначені вимоги формують основу веб-сервісу для управління дипломними проєктами. Вони охоплюють підтримку ролей користувачів, роботу з проєктами та документами, а також вимоги до безпеки, стабільності й зручності системи.

ВИСНОВКИ ДО РОЗДІЛУ

У першому розділі досліджено предметну область управління дипломними проєктами у закладах вищої освіти. Встановлено, що процес є поетапним і передбачає взаємодію студента, керівника та адміністративних ролей.

Визначено, що традиційний підхід до супроводу дипломних робіт має низку недоліків. Виявлено основні недоліки традиційного підходу: розрізнене зберігання даних, складність контролю прогресу, відсутність єдиного середовища, ручне відстеження дедлайнів та обмежене адміністрування.

У межах огляду існуючих цифрових рішень було проаналізовано системи дистанційного навчання, сервіси керування проєктами, хмарні сховища та засоби комунікації. Було встановлено, що ці рішення можуть використовуватися лише для окремих задач, але не забезпечують повного циклу управління дипломним проєктуванням.

На основі проведеного аналізу було сформовано функціональні та нефункціональні вимоги до веб-сервісу. До основних функціональних вимог віднесено автентифікацію користувачів, розмежування прав доступу, керування дипломними проєктами, етапами, документами, дедлайнами, коментарями та адміністративними процесами. До нефункціональних вимог віднесено безпеку, надійність, продуктивність, зручність використання, адаптивність і розширюваність.

Отже, результати аналізу підтверджують доцільність розробки спеціалізованого веб-сервісу для управління дипломними проєктами, який забезпечує централізовану роботу з даними, підвищує контрольованість освітнього процесу та спрощує взаємодію між студентами, керівниками й кафедрою.

					ІАЛЦ.467200.003 ПЗ	Арк.
						29
Зм.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ОБҐРУНТУВАННЯ ВИБОРУ АРХІТЕКТУРНОГО ПІДХОДУ ТА ТЕХНОЛОГІЧНОГО СТЕКУ

2.1 Вибір архітектурної моделі для розробки веб-сервісу

Під час проєктування веб-сервісу для управління дипломними проєктами було враховано необхідність розподілу системи на окремі логічні частини. Такий підхід дає змогу відокремити інтерфейс користувача, бізнес-логіку та рівень збереження даних. Для розроблюваної системи використано багаторівневу архітектурну модель, де інтерфейс користувача реалізовано на клієнтському рівні, обробка запитів і реалізація логіки предметної області виконується на сервері, а збереження та централізація даних забезпечуються окремим шаром бази даних.

У межах розробленого проєкту така модель є доцільною, оскільки система має підтримувати декілька категорій користувачів: студентів, наукових керівників, адміністраторів кафедри, адміністраторів університету та суперадміністратора. Система підтримує різні ролі користувачів з відповідними правами доступу, тому архітектура повинна забезпечувати контроль доступу, централізовану обробку даних і можливість подальшого розширення функціональності.

Загальна структура обраної архітектури подана на рисунку 2.1, де відображено взаємодію між інтерфейс користувача, серверний рівень та база даних.

					ІАЛЦ.467200.003 ПЗ	Арк.
						30
Зм.	Арк.	№ докум.	Підпис	Дата		

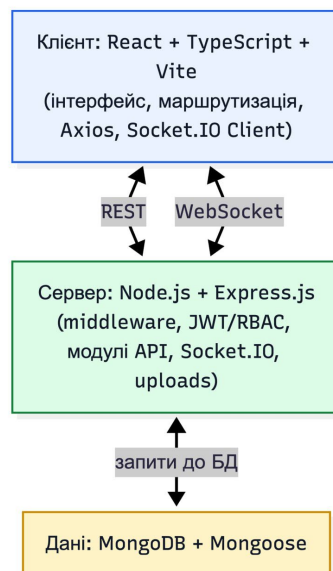


Рисунок 2.1 - Загальна багаторівнева архітектура веб-сервісу з клієнтською частиною, серверною частиною та базою даних

Для реалізації системи було визначено основні компоненти архітектури та відповідні технології, що наведено в таблиці 2.1.

Таблиця 2.1 - Основні компоненти архітектури веб-сервісу та їх технологічна реалізація

Компонент архітектури	Призначення	Реалізація у проєкті
Клієнтська частина	Забезпечення інтерфейсу користувача та взаємодії з функціями системи	React, TypeScript, Vite
Серверна частина	Обробка запитів, реалізація бізнес-логіки, автентифікація та керування доступом	Node.js, Express.js
Модуль реального часу	Передавання оновлень і подій між клієнтом та сервером	Socket.IO

Кінець таблиці 2.1

Компонент архітектури	Призначення	Реалізація у проєкті
База даних	Централізоване збереження користувачів, проєктів, документів, подій і налаштувань	MongoDB, Mongoose
Модуль документів	Завантаження, збереження та перевірка файлів	Multer, серверне файлове сховище
Модуль безпеки	Захист запитів, токени доступу, обмеження запитів	JWT, Helmet, CORS, Express-rate-limit, refresh token, CSRF, bcrypt

Архітектура системи забезпечує розподіл відповідальності між компонентами та узгоджену роботу всіх модулів, включаючи обробку даних, взаємодію в реальному часі та безпеку. Додатково було розглянуто альтернативні архітектурні підходи до розробки веб-сервісів. Їх порівняння наведено в таблиці 2.2.

Таблиця 2.2 - Порівняльний аналіз архітектурних підходів до розробки веб-сервісів

Архітектурний підхід	Переваги	Недоліки	Доцільність для проєкту
Мікросервісна архітектура	Незалежне масштабування модулів	Вища складність розробки та адміністрування	Надмірна для поточного обсягу системи

Кінець таблиці 2.2

Архітектурний підхід	Переваги	Недоліки	Доцільність для проекту
Монолітна архітектура	Простота розгортання, зручність для невеликих систем	Складніше масштабування окремих частин	Може бути використана на початковому етапі
Клієнт-серверна архітектура	Чітке розділення інтерфейсу, логіки та даних	Потребує проєктування API та механізмів безпеки	Є основним обраним підходом
Serverless-архітектура	Зменшення потреби в адмініструванні серверів	Залежність від хмарного провайдера	Не є основною через потребу в контрольованій серверній логіці

Як видно з порівняння, різні архітектурні підходи мають як переваги, так і обмеження. Монолітна архітектура є простою у реалізації, але складною для масштабування. Мікросервісна архітектура дозволяє гнучкість, однак є надмірно складною для системи такого обсягу. Serverless-підхід залежить від зовнішніх сервісів і не забезпечує повного контролю над бізнес-логікою.

Серверна частина системи реалізована як модульний моноліт із поділом на маршрути, контролери та моделі, що дозволяє поєднати простоту реалізації з можливістю структурованого розширення функціональності.

На основі проведеного аналізу встановлено, що найбільш доцільною для даного проєкту є клієнт-серверна багаторівнева архітектура з розділом на рівні. Такий підхід спрощує контроль доступу, централізовану обробку даних і можливість подальшого розвитку системи відповідно до потреб закладу освіти.

Короткий висновок: для розробки веб-сервісу було обрано багаторівневу клієнт-серверну модель, оскільки вона забезпечує розмежування відповідальності між компонентами, централізовану обробку бізнес-логіки, контроль доступу та зручну підтримку системи.

2.2 Обґрунтування використання клієнт-серверної архітектури

Клієнт-серверна архітектура передбачає розподіл функцій між клієнтом, який надсилає запити та відображає результати, і сервером, який обробляє запити, виконує перевірки, працює з базою даних та повертає відповідь. Для веб-сервісу управління дипломними проєктами такий підхід є обґрунтованим, оскільки система має обробляти дані різних користувачів, контролювати права доступу та забезпечувати стабільну роботу з документами, дедлайнами, коментарями й календарними подіями.

У розробленій системі клієнтська частина реалізує інтерфейси для студентів, керівників і адміністраторів. Вона відповідає за відображення сторінок, форм, таблиць, календарів, панелей керування та повідомлень. Серверна частина відповідає за перевірку автентифікації, обробку запитів до проєктів, документів і користувачів, а також за взаємодію з базою даних.

Для кращого розуміння взаємодії компонентів системи наведено рисунок 2.2, який відображає загальну схему обміну даними між браузером користувача, API-сервером та базою даних.

					ІАЛЦ.467200.003 ПЗ	Арк.
						34
Зм.	Арк.	№ докум.	Підпис	Дата		

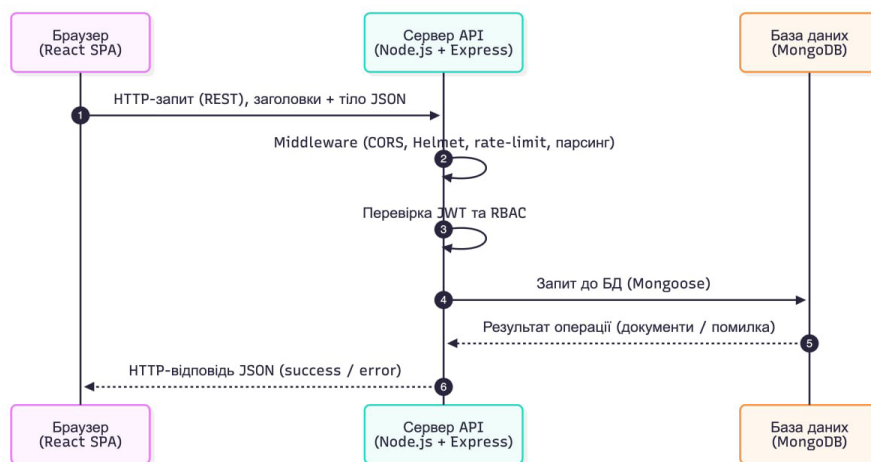


Рисунок 2.2 - Схема обміну даними між браузером, сервером API та базою даних

Типову взаємодію між компонентами системи можна описати таким чином:

1. Користувач виконує дію в інтерфейсі, наприклад завантажує документ або змінює статус етапу.
2. Клієнтська частина формує HTTP-запит до серверного API (для більшості операцій - JSON; для завантаження файлів - multipart/form-data).
3. Сервер перевіряє автентифікацію користувача та його права доступу.
4. Після перевірки виконується бізнес-логіка, пов'язана з проєктом, документом або подією.
5. Дані зберігаються або оновлюються у базі даних.
6. Клієнт отримує відповідь і оновлює інтерфейс.

Більш детальний розподіл функцій між рівнями системи наведено в таблиці 2.3, де показано відповідальність кожного рівня архітектури та приклади його реалізації у межах веб-сервісу.

Таблиця 2.3 - Розподіл відповідальності між рівнями клієнт-серверної архітектури веб-сервісу

Рівень системи	Основна відповідальність	Приклад у межах веб-сервісу
Клієнт	Відображення інтерфейсу та приймання дій користувача	Кабінет студента, панель керівника, панель кафедри
API-сервер	Обробка запитів і реалізація бізнес-правил	Авторизація, перевірка доступу, оновлення проєкту
База даних	Збереження структурованої інформації	Користувачі, дипломні проєкти, документи, події
Real-time (Socket.IO)	Передавання подій у реальному часі	Оновлення коментарів, повідомлень і статусів

Важливою перевагою клієнт-серверної архітектури є можливість централізованого контролю безпеки. У системі передбачено автентифікацію за допомогою токенів, перевірку активності користувача, контроль ролей і обмеження доступу до проєктів. Завдяки цьому студент може працювати лише зі своїм дипломним проєктом, керівник - із проєктами закріплених студентів, а адміністратор кафедри - з даними в межах відповідної кафедри.

Окремо слід зазначити використання механізму обміну подіями в реальному часі. Для цього застосовується Socket.IO, що дозволяє організувати підключення користувачів до відповідних кімнат проєкту або керівника. Такий механізм є корисним для відображення актуальних змін, пов'язаних із документами, коментарями, нагадуваннями або іншими подіями в системі.

Переваги клієнт-серверної архітектури та їх значення для розроблюваного веб-сервісу наведено в таблиці 2.4, де узагальнено основні переваги підходу та їх практичне значення для системи.

Таблиця 2.4 - Переваги клієнт-серверної архітектури та їх значення для розроблюваного веб-сервісу

Перевага клієнт-серверної архітектури	Значення для розроблюваної системи
Розділення інтерфейсу та бізнес-логіки	Спрощується підтримка клієнтської та серверної частин
Централізована робота з даними	Усі основні дані зберігаються в єдиній базі
Контроль доступу на сервері	Зменшується ризик несанкціонованих дій
Можливість масштабування	Можна розвивати окремо клієнт, сервер і базу даних
Підтримка різних клієнтів	У майбутньому може бути додано мобільний застосунок або інший інтерфейс
Інтеграція з WebSocket	Забезпечується передавання оновлень у реальному часі

Короткий висновок: використання клієнт-серверної архітектури є обґрунтованим для системи, оскільки вона забезпечує централізовану обробку даних, контроль доступу, масштабованість і можливість підтримки інтерактивної взаємодії між користувачами.

2.3 Вибір технологій для реалізації клієнтської частини системи

Клієнтська частина веб-сервісу є основним середовищем взаємодії користувача із системою. Через це виконуються перегляд дипломного проєкту,

робота з етапами, завантаження документів, перегляд календаря, отримання повідомлень і контроль стану робіт. Тому під час вибору технологій для клієнтської частини було враховано зручність розробки, підтримку компонентного підходу, типізацію та швидкість збірки.

Для реалізації клієнтської частини було використано React у поєднанні з TypeScript та Vite. React забезпечує компонентну модель побудови інтерфейсу, що є доцільним для системи з великою кількістю сторінок і модулів. TypeScript використовується для статичної типізації, що зменшує кількість помилок під час розробки. Vite застосовується як інструмент швидкої збірки та запуску клієнтського застосунку в режимі розробки. Детальніше описано у таблиці 2.5.

Таблиця 2.5 - Обґрунтування вибору технологій для реалізації клієнтської частини веб-сервісу

Технологія	Призначення	Обґрунтування використання
React	Побудова користувацького інтерфейсу	Дозволяє створювати інтерфейс із повторно використовуваних компонентів
TypeScript	Статична типізація JavaScript-коду	Підвищує надійність коду та спрощує роботу зі структурами даних
Vite	Інструмент збірки клієнтського застосунку	Забезпечує швидкий запуск і оновлення під час розробки
React Router	Маршрутизація між сторінками	Дає змогу реалізувати окремі кабінети та розділи системи
Axios	Виконання HTTP-запитів до серверного API	Спрощує взаємодію клієнтської частини із сервером

Кінець таблиці 2.5

Технологія	Призначення	Обґрунтування використання
Socket.IO Client	Підключення до подій у реальному часі	Забезпечує отримання актуальних повідомлень і оновлень
Tailwind CSS	Стилізація інтерфейсу	Дозволяє швидко створювати адаптивні інтерфейсні елементи

Загальна структура клієнтської частини системи наведена на рисунку 2.3, який відображає основні сторінки, модулі та компоненти веб-застосунку. На схемі показано поділ інтерфейсу на логічні частини для різних ролей користувачів, а також спільні компоненти, що використовуються у всій системі.

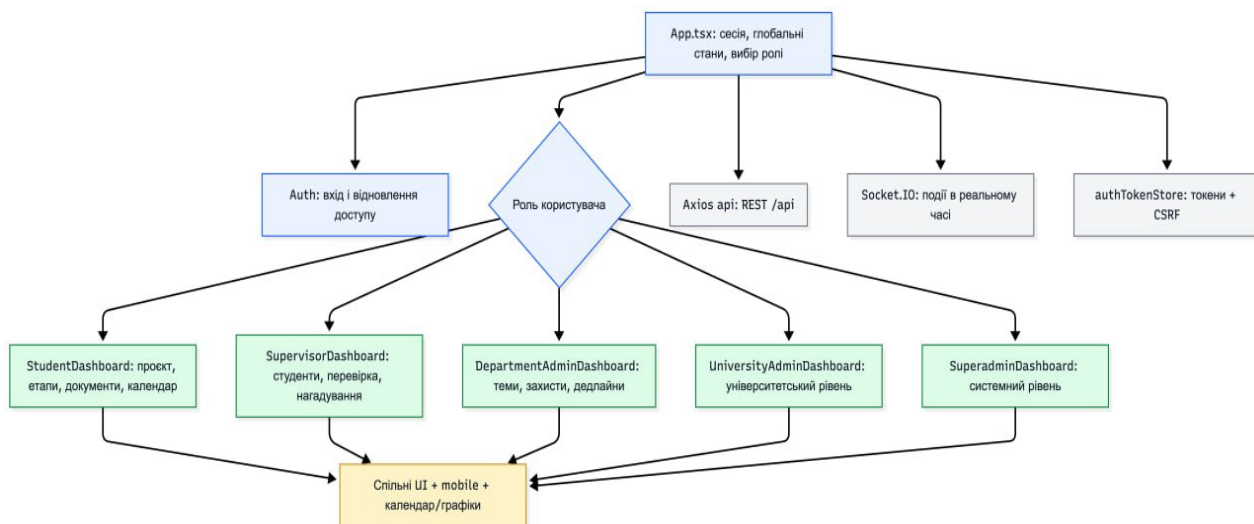


Рисунок 2.3 - Структура клієнтської частини веб-сервісу з основними сторінками та компонентами

У проєкті клієнтська частина має модульну структуру. Окремі функціональні області винесено у відповідні розділи, зокрема сторінки користувачів, спільні компоненти інтерфейсу та допоміжні модулі. Такий підхід забезпечує логічну організацію коду та спрощує підтримку системи.

Для реалізації інтерфейсу використовуються компоненти, які відповідають за окремі частини функціональності: відображення профілю користувача, календаря, списку студентів, документів, нагадувань, модальних

вікон, таблиць, форм та багато інших. У React інтерфейс складається з компонентів, які можна повторно застосовувати.

Оскільки веб-сервіс передбачає роботу з календарними подіями, дедлайнами та захистами, у клієнтській частині використовується бібліотека FullCalendar. Вона дає змогу візуально відображати консультації, дедлайни, події захисту та інші важливі дати. Для аналітичного представлення інформації застосовуються бібліотеки Chart.js, Recharts і react-chartjs-2 для побудови графіків прогресу, статистики та динаміки виконання завдань. Допоміжні бібліотеки клієнтської частини наведено в таблиці 2.6.

Таблиця 2.6 - Допоміжні бібліотеки клієнтської частини та їхнє функціональне призначення

Бібліотека	Функціональне призначення в системі
FullCalendar	Відображення академічних подій, дедлайнів, консультацій і захистів
Chart.js / Recharts	Побудова графіків, статистики та візуалізації прогресу
jsPDF / jspdf-autotable	Формування PDF-документів і табличних звітів
xlsx / read-excel-file	Імпорт або експорт табличних даних
PapaParse	Обробка CSV-файлів
Framer Motion	Анімації інтерфейсних елементів
Lucide React	Використання іконок в інтерфейсі
lodash.debounce	Оптимізація пошуку та обробки введення користувача

Для системи управління дипломними проєктами важливим є адаптивний інтерфейс, оскільки користувачі можуть працювати із сервісом з різних пристроїв. У проєкті передбачено компоненти для мобільної навігації, модальних вікон і адаптованих елементів інтерфейсу. Це дозволяє забезпечити доступ як з комп'ютера, так і з мобільного пристрою.

Порівняння технологій, які можуть використовуватися для реалізації клієнтської частини, наведено в таблиці 2.7, де проаналізовано їх переваги, недоліки та обґрунтування вибору.

Таблиця 2.7 - Порівняльний аналіз технологій розробки клієнтської частини веб-сервісу

Технологія	Переваги	Недоліки	Причина вибору / невибору
React	Компонентність, велика екосистема, гнучкість	Потребує додаткових бібліотек для маршрутизації та стану	Обрано через гнучкість і відповідність структурі системи
Angular	Повноцінний фреймворк, вбудовані інструменти	Вища складність і більший обсяг шаблонного коду	Не обрано через надмірність для даного проєкту
Svelte	Висока продуктивність, компактний код	Менша екосистема порівняно з React	Не обрано через обмежену кількість вже існуючих реалізацій

Кінець таблиці 2.7

Технологія	Переваги	Недоліки	Причина вибору / невибору
Vue.js	Простота використання, компонентний підхід	Менша поширеність у деяких корпоративних середовищах	Може бути альтернативою, але не використовувався

Таким чином, обраний стек клієнтської частини забезпечує можливість створення сучасного вебзастосунку з адаптивним інтерфейсом, маршрутизацією, інтеграцією з серверним API, календарями, графіками, документами та оновленнями в реальному часі.

Короткий висновок: клієнтську частину реалізовано на основі React, TypeScript і Vite. Обраний стек забезпечує зручну розробку інтерфейсу та підтримує необхідний функціонал системи.

2.4 Вибір технологій для розробки серверної частини веб-сервісу

Серверна частина веб-сервісу призначена для обробки запитів клієнтської частини, реалізації бізнес-логіки, взаємодії з базою даних, автентифікації користувачів, керування документами, подіями, повідомленнями та правами доступу. Ця частина побудована на основі Node.js із використанням Express.js як фреймворку. Така комбінація інструментів дозволяє обробляти вхідні запити у неблокуючому режимі та ефективно підтримує роботу системи при інтенсивному обміні даними між клієнтом і сервером. Для системи управління дипломними проектами це є важливим, оскільки сервер має обробляти запити

авторизації, отримання даних про проєкти, завантаження документів, зміну статусів, створення коментарів, роботу з календарем і повідомленнями.

Express.js було використано як серверний фреймворк для побудови API. Він забезпечує маршрутизацію запитів, підключення проміжного програмного забезпечення, обробку JSON-даних, multipart/form-data (завантаження файлів) і структурування серверної логіки за окремими маршрутами. У проєкті серверна частина поділена на маршрути автентифікації, проєктів, документів, керівників, синхронізації та адміністрування.

Серверна частина як модульний моноліт: один процес Node.js із поділом на routes, controllers, models та middleware. Для реалізації було визначено технології та їх функціональне призначення у веб-сервісі в таблиці 2.8.

Таблиця 2.8 - Обґрунтування вибору технологій для реалізації серверної частини веб-сервісу

Технологія	Призначення	Використання у веб-сервісі
Node.js	Середовище виконання JavaScript на сервері	Запуск серверної частини та обробка запитів
Express.js	Фреймворк для створення веб-API	Реалізація маршрутів, middleware та REST-запитів
Mongoose	ODM-бібліотека для MongoDB	Опис моделей користувачів, проєктів, документів і подій
Socket.IO	Обмін даними в реальному часі	Сповіщення, оновлення станів, кімнати проєктів і керівників
Bcrypt.js	Хешування паролів	Захист паролів користувачів

Кінець таблиці 2.8

Технологія	Призначення	Використання у веб-сервісі
Multer	Обробка завантаження файлів	Завантаження документів дипломного проєкту
JSON Web Token	Токени доступу	Автентифікація користувачів і перевірка сесій
cookie-parser	Обробка cookie	Зберігання refresh-токена (httpOnly cookie)
CSRF-token	Захист змінювальних запитів	Перевірка CSRF при роботі з API
Nodemailer	Надсилання електронних листів	Повідомлення та відновлення доступу
Helmet	Захист HTTP-заголовків	Підвищення базового рівня безпеки API
Express-rate-limit	Обмеження кількості запитів	Захист від надмірної кількості звернень до API

Для кращого розуміння внутрішньої організації серверної частини було побудовано структурну схему серверного застосунку. На рисунку 2.4 показано взаємодію маршрутів, контролерів, моделей та middleware у межах серверної архітектури.

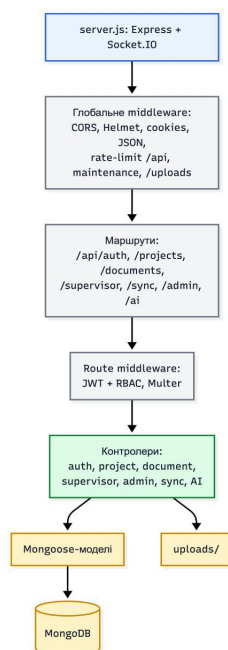


Рисунок 2.4 - Структура серверної частини веб-сервісу із маршрутами, контролерами, моделями та middleware

У серверній частині реалізовано REST-підхід до організації API. Клієнтська частина надсилає HTTP-запити до відповідних маршрутів, після чого сервер виконує перевірку автентифікації, обробляє бізнес-логіку та звертається до бази даних. Це рішення забезпечує розмежування роботи інтерфейсу та серверної частини системи.

Для організації взаємодії між користувацьким інтерфейсом і сервером застосовується REST API. Основні групи маршрутів серверного API та їх функціональне призначення наведено в таблиці 2.9.

Таблиця 2.9 - Структура маршрутів серверного API та їхнє функціональне призначення

Група маршрутів	Призначення
/api/auth	Вхід, вихід, оновлення доступу (refresh); облікові записи (за запрошенням / адміністратором)

Кінець таблиці 2.9

Група маршрутів	Призначення
/api/projects	Керування дипломними проєктами, етапами, подіями та коментарями
/api/documents	Завантаження, збереження та обробка документів
/api/supervisor	Функціонал наукового керівника для роботи зі студентами
/api/sync	Синхронізація даних і подій між частинами системи
/api/admin	Адміністративні функції кафедри та університету

Окремою складовою серверної частини є підтримка подій у реальному часі. Для цього використовується Socket.IO. Користувачі можуть підключатися до кімнат конкретного дипломного проєкту або керівника, що дає змогу оперативно отримувати оновлення, пов'язані з документами, коментарями, нагадуваннями та зміною стану проєкту.

Короткий висновок: серверна частина побудована на Node.js та Express.js, оскільки ці технології забезпечують гнучку побудову API, асинхронну обробку запитів, інтеграцію з MongoDB, підтримку файлів, авторизації та подій у реальному часі.

2.5 Обґрунтування вибору системи керування базами даних

Для збереження даних веб-сервісу було обрано MongoDB. Вибір цієї NoSQL бази даних зумовлений тим, що вона ефективно працює зі складними структурами даних, характерними для системи управління дипломними проєктами.

У межах веб-сервісу один проєкт може містити тему, опис, статус погодження, студентів, керівника, етапи, підзавдання, документи, коментарі, ресурси, календарні події та нагадування. Для такої структури MongoDB є зручним рішенням, оскільки дозволяє зберігати пов'язані вкладені дані в межах одного документа або через посилання на інші колекції.

Для роботи з MongoDB було використано Mongoose. Ця бібліотека забезпечує опис схем даних, валідацію полів, створення моделей і зручну взаємодію з базою даних на рівні серверного коду. Для реалізації структури даних у веб-сервісі виділено основні об'єкти, які наведено в таблиці 2.10

Таблиця 2.10 - Основні об'єкти бази даних веб-сервісу та їхнє призначення

Об'єкти бази даних	Призначення
User	Збереження користувачів, ролей, контактних даних, налаштувань і стану доступу
Project	Збереження дипломних проєктів, тем, етапів, документів, подій і нагадувань
UserSession	Облік активних користувацьких сесій
ExaminationCommission	Збереження даних про комісії
UniversityAcademicDeadline	Збереження академічних дедлайнів
UniversityBroadcast	Масові оголошення та повідомлення
SecurityAuditEvent	Фіксація подій, пов'язаних із безпекою
InviteToken/SupervisorInviteToken	Запрошення студентів і керівників
BackupJob	Облік резервного копіювання даних

Кінець таблиці 2.10

Сутність бази даних	Призначення
SyncEvent	Журнал ключових змін / подій синхронізації

Для наочного представлення зв’язків між основними сутностями бази даних було побудовано відповідну діаграму. На рисунку 2.5 показано структуру основних моделей і взаємозв’язків між ними.

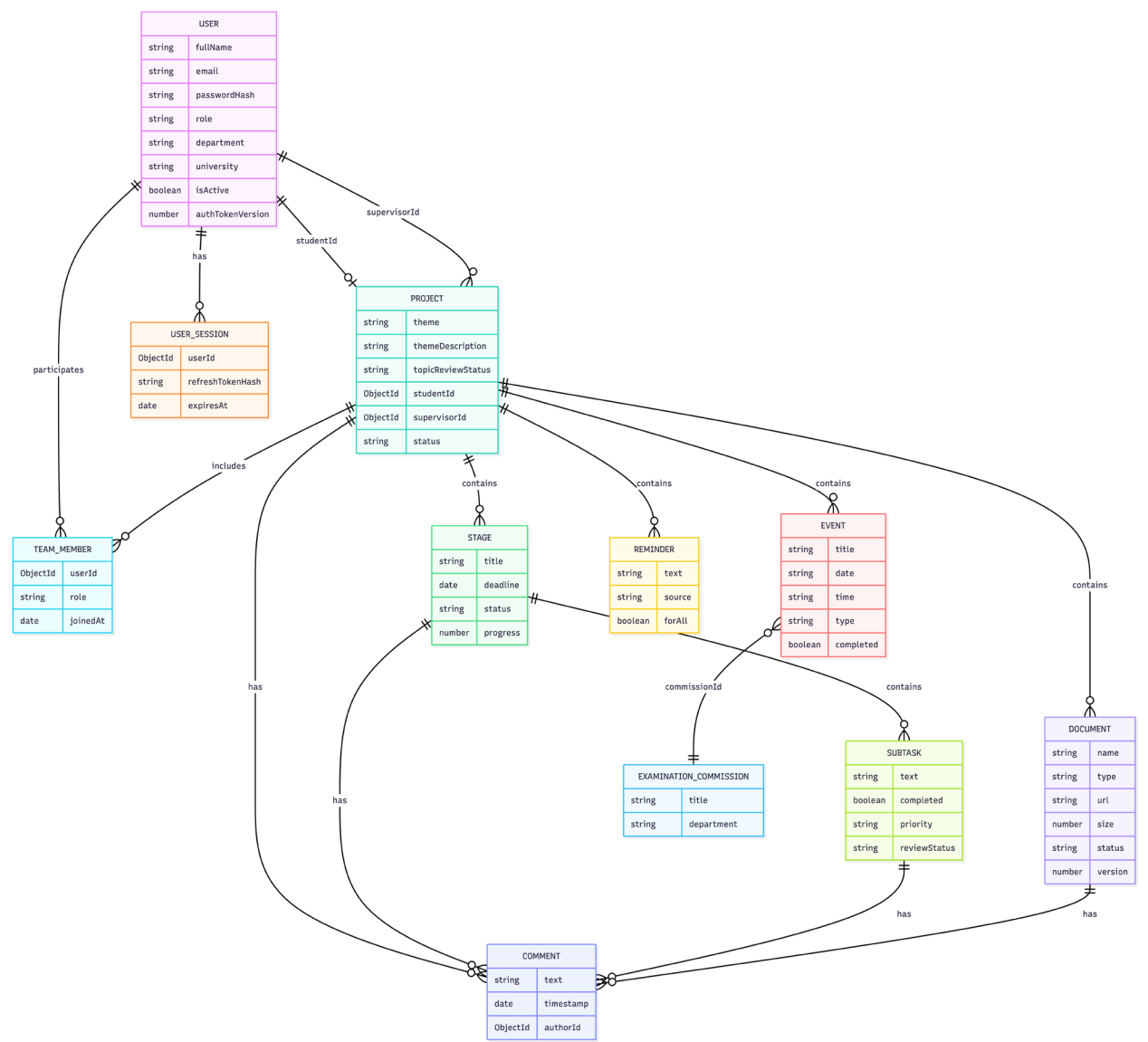


Рисунок 2.5 - ER-діаграма основних сутностей бази даних веб-сервісу

Діаграма демонструє, як у системі організовано зв’язки між користувачами, дипломними проєктами, документами та іншими сутностями.

Це дозволяє забезпечити цілісність даних і підтримувати взаємодію між різними модулями веб-сервісу. Під час вибору системи керування базами даних було розглянуто кілька популярних рішень, які наведено в таблиці 2.11.

Таблиця 2.11 - Порівняльний аналіз систем керування базами даних для реалізації веб-сервісу

СКБД	Тип	Переваги	Недоліки	Доцільність для проєкту
MongoDB	Документ оорієнтована NoSQL	Гнучка структура документів, зручність для вкладених даних	Менш сувора схема порівняно з SQL-рішеннями	Обрано для реалізації системи
PostgreSQL	Реляційна	Надійність, транзакції, складні SQL-запити	Потребує жорсткішого проєктування таблиць і зв'язків	Може бути альтернативою
MySQL	Реляційна	Поширеність, стабільність, підтримка SQL	Менша гнучкість для вкладених структур	Не обрано через структуру даних проєкту

Кінець таблиці 2.11

СКБД	Тип	Переваги	Недоліки	Доцільність для проєкту
SQLite	Файлова реляційна	Простота локального використання	Обмежена придатність для багатокористувачького веб-сервісу	Недоцільна для основної системи

Проведене порівняння показує, що MongoDB найбільше відповідає вимогам розроблюваної системи. Основною перевагою є гнучка структура документів, що є зручним для роботи зі складними даними дипломного проєкту. Вибір MongoDB також обґрунтовується необхідністю швидкого розвитку структури даних. У процесі розробки системи можуть додаватися нові поля до профілю користувача, проєкту, документів або календарних подій без складних міграцій реляційних таблиць.

Короткий висновок: MongoDB обрано як основну базу даних завдяки гнучкій структурі та зручній роботі зі складними даними, а використання Mongoose забезпечує опис структури даних і їх валідацію на сервері.

2.6 Засоби забезпечення безпеки, автентифікації та розмежування прав доступу

Оскільки веб-сервіс працює з персональними даними користувачів, інформацією про дипломні проєкти, документами, коментарями та адміністративними діями, питання безпеки є одним із ключових. У системі було передбачено механізми автентифікації, авторизації, хешування паролів, перевірки ролей, обмеження запитів і захисту HTTP-заголовків.

Автентифікація реалізована на основі JWT (access token) для API-запитів та refresh-токена (httpOnly cookie) для оновлення сесії без повторного входу. Після успішного входу клієнт отримує access JWT і використовує його в заголовку (Authorization: Bearer ...). Паралельно для частини операцій застосовується CSRF-token. Під час захищених запитів сервер перевіряє: валідність JWT (підпис і термін дії), роль користувача, прапор isActive та authTokenVersion (для примусового завершення сесій при блокуванні або зміні облікових даних). Основні засоби безпеки та їх призначення наведено в таблиці 2.12.

Таблиця 2.12 - Основні механізми забезпечення безпеки веб-сервісу та їхнє функціональне призначення

Механізм безпеки	Призначення	Роль у системі
JWT	Автентифікація користувача	Підтвердження особи під час звернення до API
Refresh token (cookie)	Оновлення сесії	Продовження роботи без повторного логіну
CSRF	Захист змінювальних запитів	Зменшення ризику підроблених запитів
authTokenVersion	Версіонування сесії	Примусове завершення всіх сесій користувача
Bcrypt.js	Хешування паролів	Захист паролів від збереження у відкритому вигляді
Role-based access control	Розмежування доступу	Обмеження функцій відповідно до ролі користувача

Кінець таблиці 2.12

Механізм безпеки	Призначення	Роль у системі
Helmet	Захист HTTP-заголовків	Зменшення ризиків типових веб-атак
CORS	Контроль доступу з клієнтських доменів	Дозвіл запитів лише з визначеного клієнтського застосунку
Express-rate-limit	Обмеження кількості запитів	Захист API від надмірного навантаження
Cookie-parser	Обробка cookie	Підтримка механізмів сесій і токенів
Перевірка активності користувача	Контроль стану облікового запису	Заборона доступу для неактивних користувачів

Використання наведених механізмів дозволяє забезпечити базовий рівень захисту веб-сервісу. Це зменшує ризик несанкціонованого доступу, підробки запитів і компрометації користувацьких даних.

Для кращого розуміння процесу автентифікації користувача було побудовано відповідну схему взаємодії між клієнтом і сервером. На рисунку 2.6 показано процес використання JWT у межах веб-сервісу.

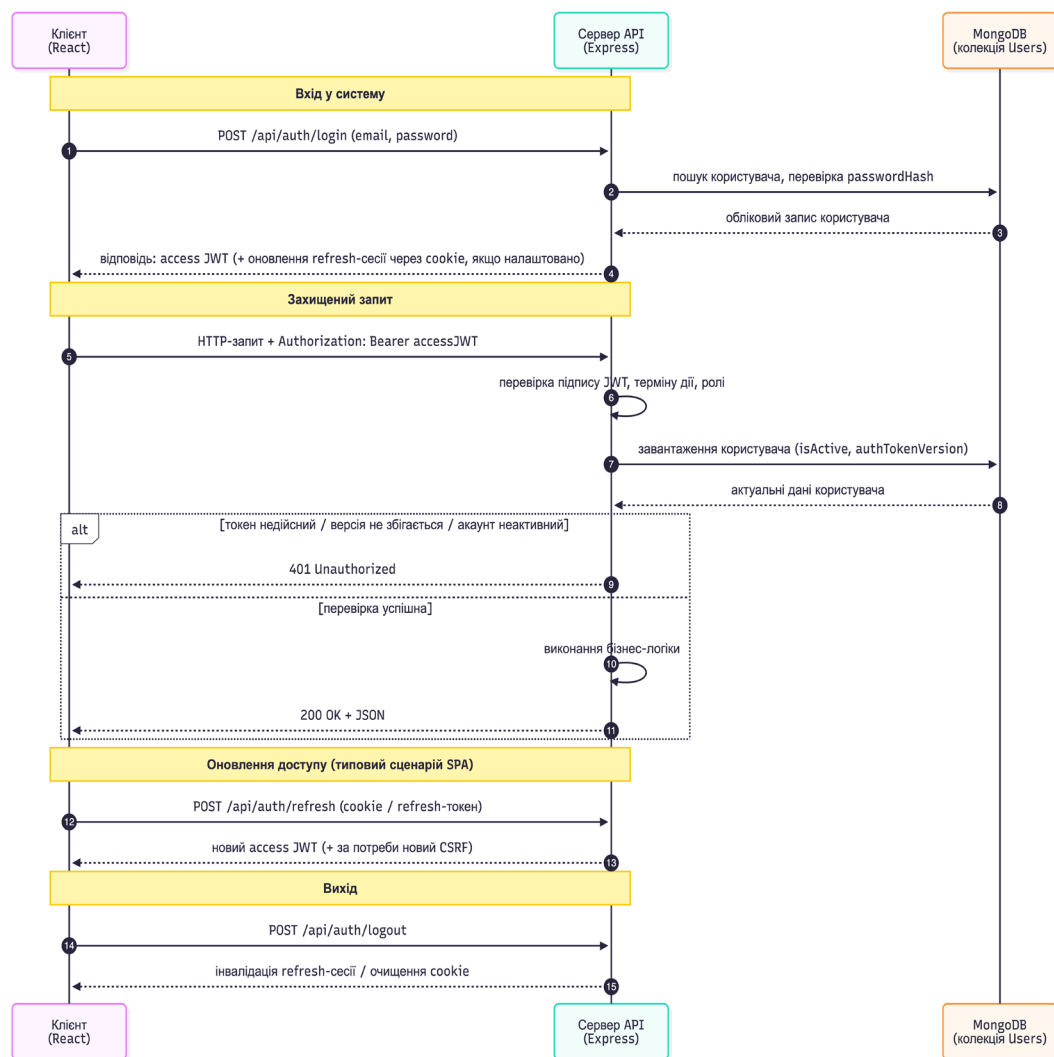


Рисунок 2.6 - Схема автентифікації користувача за допомогою JWT

Представлена схема на рисунку 2.6 демонструє послідовність перевірки користувача та механізм роботи access і refresh токенів.

У системі передбачено рольову модель доступу, а саме: студена, наукового керівника, адміністратора кафедри, університету та суперадміністратор, і для кожної з ролей є свій набір дозволених функцій. Наприклад, студент може працювати зі своїм дипломним проєктом, керівник - із проєктами закріплених студентів, адміністратор кафедри і університету - з даними відповідної кафедри, а суперадміністратор - із глобальними налаштуваннями системи. Основні рівні їх доступу до функціональності веб-сервісу наведено в таблиці 2.13.

Таблиця 2.13 - Рівні доступу користувачів веб-сервісу відповідно до рольової моделі

Роль	Рівень доступу
Student	Доступ до власного профілю, проєкту, документів, дедлайнів і повідомлень
Supervisor	Доступ до студентів, закріплених за керівником, їхніх проєктів і документів
DepartmentAdmin	Доступ до студентів, керівників, тем, дедлайнів і захистів у межах кафедри
UniversityAdmin	Доступ до адміністративних даних рівня університету
SuperAdmin	Доступ до системних налаштувань і глобального адміністрування

Розмежування прав доступу дозволяє забезпечити контроль дій користувачів у межах системи, та запобігає несанкціонованому доступу до даних дипломних проєктів.

Для захисту даних система перевіряє не лише роль користувача, а й його доступ до конкретного проєкту. Це не дозволяє перегляд або зміну інформації сторонніми користувачами. Ця перевірка виконується на серверному рівні, що є правильним підходом з погляду безпеки. Додатково реалізовано захист WebSocket-з'єднань. Під час підключення перевіряються JWT-токен, активність користувача та актуальність його сесії, після чого надається доступ до відповідних каналів.

Короткий висновок: у системі реалізовано основні механізми безпеки, зокрема автентифікацію, контроль доступу, захист паролів, обмеження запитів і захист WebSocket-з'єднань. Це забезпечує безпечну роботу з даними та функціями веб-сервісу.

ВИСНОВКИ ДО РОЗДІЛУ

У другому розділі було обґрунтовано вибір архітектурного підходу та технологічного стеку для розробки веб-сервісу управління дипломними проєктами. Визначено, що для реалізації системи доцільно використовувати клієнт-серверну архітектуру з поділом на користувацький інтерфейс, серверну логіку та сховище даних.

У фронтенд-частині системи обрано React, TypeScript і Vite, що забезпечують компонентну побудову інтерфейсу, типізацію, швидку розробку та інтеграцію з серверним API. У бекенд-частині - Node.js та Express.js, які дозволяють реалізувати REST API, бізнес-логіку, роботу з файлами, автентифікацію та взаємодію з базою даних.

Для збереження даних обрано MongoDB у поєднанні з Mongoose. Такий вибір обґрунтовано складною вкладеною структурою даних дипломного проєкту, що включає етапи, підзавдання, документи, події, коментарі та нагадування.

Також було розглянуто засоби безпеки, автентифікації та розмежування прав доступу. У системі передбачено використання JWT, хешування паролів, рольової моделі доступу, перевірки належності користувача до проєкту, захисту HTTP-заголовків, CORS, обмеження кількості запитів і перевірки WebSocket-з'єднань.

Отже, обраний технологічний стек відповідає вимогам до веб-сервісу управління дипломними проєктами та забезпечує можливість подальшого розвитку, масштабування й підтримки системи.

					ІАЛЦ.467200.003 ПЗ	Арк.
						55
Зм.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБ-СЕРВІСУ

3.1 Мета, призначення та основні завдання розроблюваної системи

Метою розробки є створення веб-сервісу для управління дипломними проєктами, який забезпечує централізовану організацію роботи студентів, наукових керівників та адміністративних працівників. Система призначена для збереження даних про користувачів, теми, етапи виконання, документи, календарні події, дедлайни, коментарі та службові повідомлення. Для підтвердження наявності базового сценарію доступу до системи на рисунку 3.1 наведено сторінку автентифікації користувача.

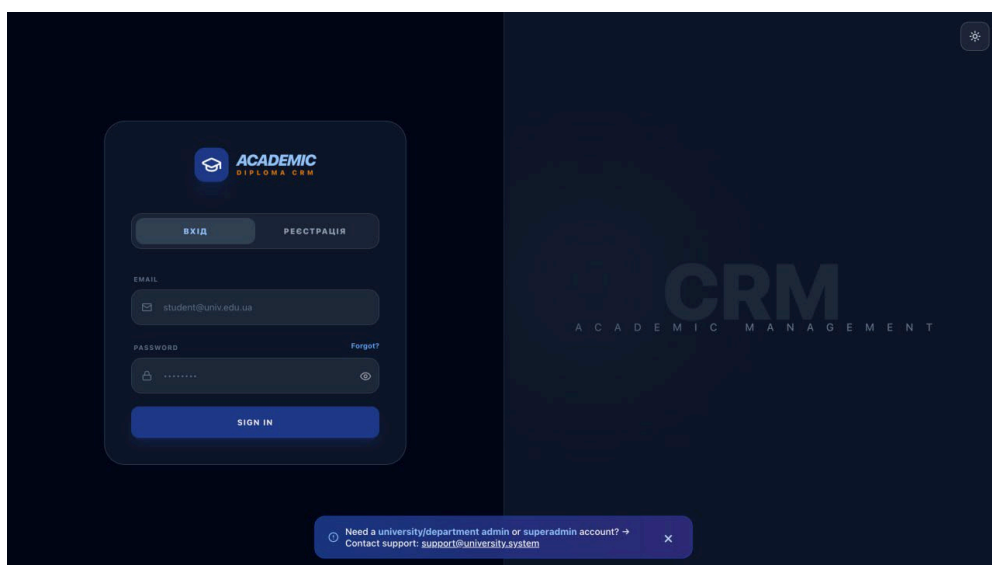


Рисунок 3.1 - Сторінка автентифікації користувача (вхід до системи)

Розроблена система орієнтована на автоматизацію процесів, які за традиційного підходу виконуються вручну або за допомогою різних інструментів. До таких процесів належать погодження тем, контроль виконання етапів, перевірка документів, інформування дедлайнів, організація захистів і

					ІАЛЦ.467200.003 ПЗ	Арк.
						56
Зм.	Арк.	№ докум.	Підпис	Дата		

взаємодія між студентом та керівником. Загальне призначення системи та взаємодія основних категорій користувачів із веб-сервісом на рисунку 3.2.

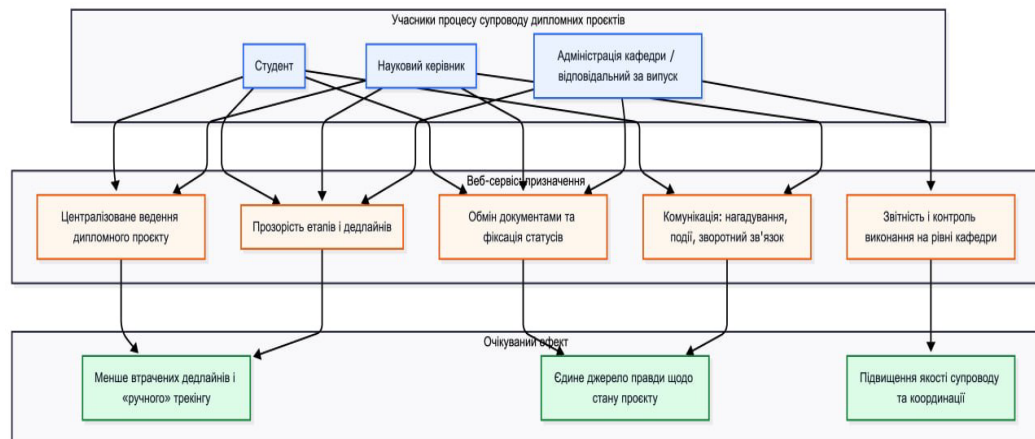


Рисунок 3.2 - Загальна схема призначення веб-сервісу для управління дипломними проєктами

Основні завдання розроблюваного веб-сервісу та способи їх реалізації наведено в таблиці 3.1.

Таблиця 3.1 - Основні завдання розроблюваного веб-сервісу та шляхи їх реалізації

Завдання системи	Опис реалізації
Централізоване збереження даних	Інформація про користувачів, проєкти, документи та події зберігається в базі даних
Розмежування користувачів за ролями	Передбачено ролі студента, керівника, адміністратора кафедри, адміністратора університету та суперадміністратора
Контроль дедлайнів	Використовуються календарні події, етапи роботи та нагадування
Управління дипломними проєктами	Реалізовано роботу з темами, описами, статусами, етапами та підзавданнями

Кінець таблиці 3.1

Завдання системи	Опис реалізації
Робота з документами	Передбачено завантаження файлів, збереження версій, статусів перевірки та коментарів
Комунікація між учасниками	Реалізовано коментарі, нагадування та сповіщення
Адміністративний контроль	Адміністратор кафедри отримує інструменти для контролю студентів, керівників, тем і захистів

Наведені в таблиці 3.1 завдання формують основу функціональності системи та охоплюють як організаційні, так і інформаційні аспекти супроводу дипломного проєктування.

Функціональна структура системи була сформована відповідно до ролей користувачів. Студент працює з власним проєктом, керівник контролює своїх студентів, адміністратор кафедри здійснює організаційне управління, а адміністративні ролі вищого рівня забезпечують контроль системних процесів.

Короткий висновок: система призначена для автоматизації супроводу дипломного проєктування та забезпечує єдине середовище для роботи з темами, документами, дедлайнами, користувачами й адміністративними процесами.

3.2 Проєктування структури та користувацької взаємодії

Під час проєктування структури веб-сервісу використано клієнт-серверний підхід. Клієнтська частина відповідає за відображення інтерфейсу користувача, приймання дій і надсилення запитів до серверного API. Серверна частина виконує перевірку автентифікації, обробку бізнес-логіки, взаємодію з базою даних, роботу з файлами та передачу подій у реальному часі.

Логіка взаємодії побудована навколо дипломного проєкту як основної сутності системи. До проєкту прив'язуються студент, керівник, тема, етапи виконання, підзавдання, документи, календарні події, нагадування та коментарі. Це дозволяє об'єднати дані, пов'язані з дипломною роботою, в межах одного керованого процесу.

На рисунку 3.3 подано схему взаємодії між клієнтською частиною, серверним API, базою даних і модулем реального часу. Така структура забезпечує централізовану обробку даних та контроль доступу.

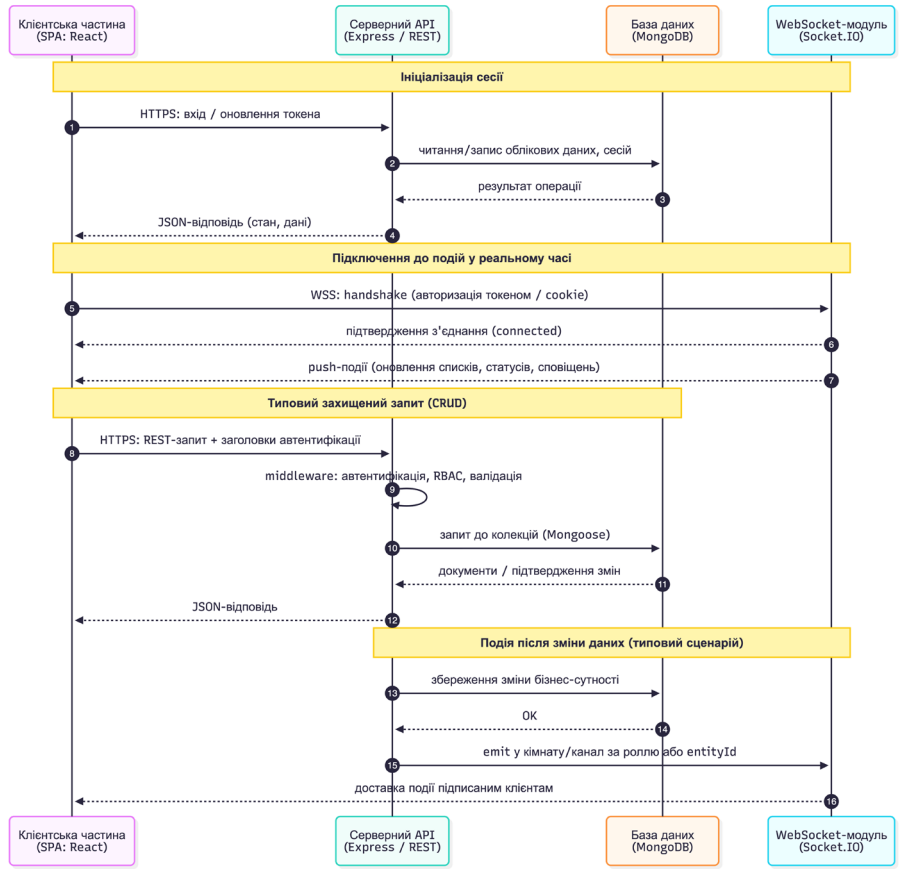


Рисунок 3.3 - Схема взаємодії клієнтської частини, серверного API, бази даних і WebSocket-модуля

Для кращого розуміння організації веб-сервісу в таблиці 3.2 наведено багаторівневу структуру системи та призначення кожного рівня. Це дозволяє показати розподіл функцій між клієнтською частиною, сервером, базою даних і допоміжними модулями.

Таблиця 3.2 - Загальна багаторівнева структура веб-сервісу та призначення його рівнів

Рівень системи	Призначення	Основні компоненти
Клієнтський рівень	Взаємодія користувача із системою	Кабінет студента, панель керівника, панель адміністратора кафедри
API-рівень	Обробка запитів і виконання бізнес-логіки	Маршрути автентифікації, проєктів, документів, керівника, адміністратора
Рівень даних	Збереження інформації системи	Колекції користувачів, проєктів, сесій, комісій, дедлайнів, сповіщень
Рівень реального часу	Передача оперативних оновлень	Socket.IO-кімнати проєктів і керівників
Файловий рівень	Збереження завантажених документів	Серверна директорія для файлів і посилання на документи в базі

Основний сценарій взаємодії користувача із системою можна описати так:

1. Користувач проходить автентифікацію.
2. Сервер перевіряє облікові дані та роль користувача.
3. Клієнтська частина отримує доступний функціонал відповідно до ролі.
4. Користувач виконує дію, наприклад завантажує документ або змінює етап.
5. Сервер перевіряє право роботи відповідного проєкту.

6. Дані оновлюються в базі даних.

7. За потреби інші користувачі отримують оновлення через механізм реального часу.

Логіка взаємодії користувачів із системою залежить від їхньої ролі та рівня доступу. Основні можливості кожної категорії користувачів наведено в таблиці 3.3.

Таблиця 3.3 - Основна логіка взаємодії користувачів із веб-сервісом відповідно до їхніх ролей

Роль користувача	Основна логіка взаємодії
Студент	Переглядає власний проєкт, виконує етапи, завантажує документи, отримує коментарі
Науковий керівник	Переглядає студентські проєкти, перевіряє документи, залишає коментарі, створює нагадування
Адміністратор кафедри	Контролює студентів, керівників, теми, дедлайни, захисти та кафедральні процеси
Адміністратор університету	Працює з даними та налаштуваннями університетського рівня
Суперадміністратор	Виконує системне адміністрування та контроль глобальних параметрів

Таким чином, логіка взаємодії в системі безпосередньо залежить від ролі користувача та його функцій у процесі дипломного проєктування.

Для взаємодії між клієнтом і сервером застосовується API, структуроване за функціональними групами. Це спрощує підтримку системи та дозволяє відокремити логіку авторизації, роботи з проєктами, документами, синхронізацією й адміністративними діями.

Отож структура веб-сервісу була спроектована як сукупність взаємопов'язаних модулів, що забезпечують роботу з користувачами, проєктами, документами, подіями та правами доступу. Основою логіки системи є дипломний проєкт, навколо якого формуються всі основні процеси.

3.3 Розробка бази даних та опис основних сутностей системи

Основними сутностями системи є користувач і дипломний проєкт. Сутність користувача містить персональні дані, роль, контактну інформацію, налаштування інтерфейсу та службові параметри контролю доступу. Сутність дипломного проєкту містить тему, опис, статус погодження, учасників, етапи, підзавдання, документи, календарні події, нагадування та коментарі.

На рисунку 3.4 наведено UML-діаграму основних сутностей бази даних веб-сервісу та зв'язків між ними. Діаграма демонструє структуру збереження даних і взаємозв'язки між користувачами, дипломними проєктами, документами, етапами виконання та іншими компонентами системи.

					ІАЛЦ.467200.003 ПЗ	Арк.
						62
Зм.	Арк.	№ докум.	Підпис	Дата		

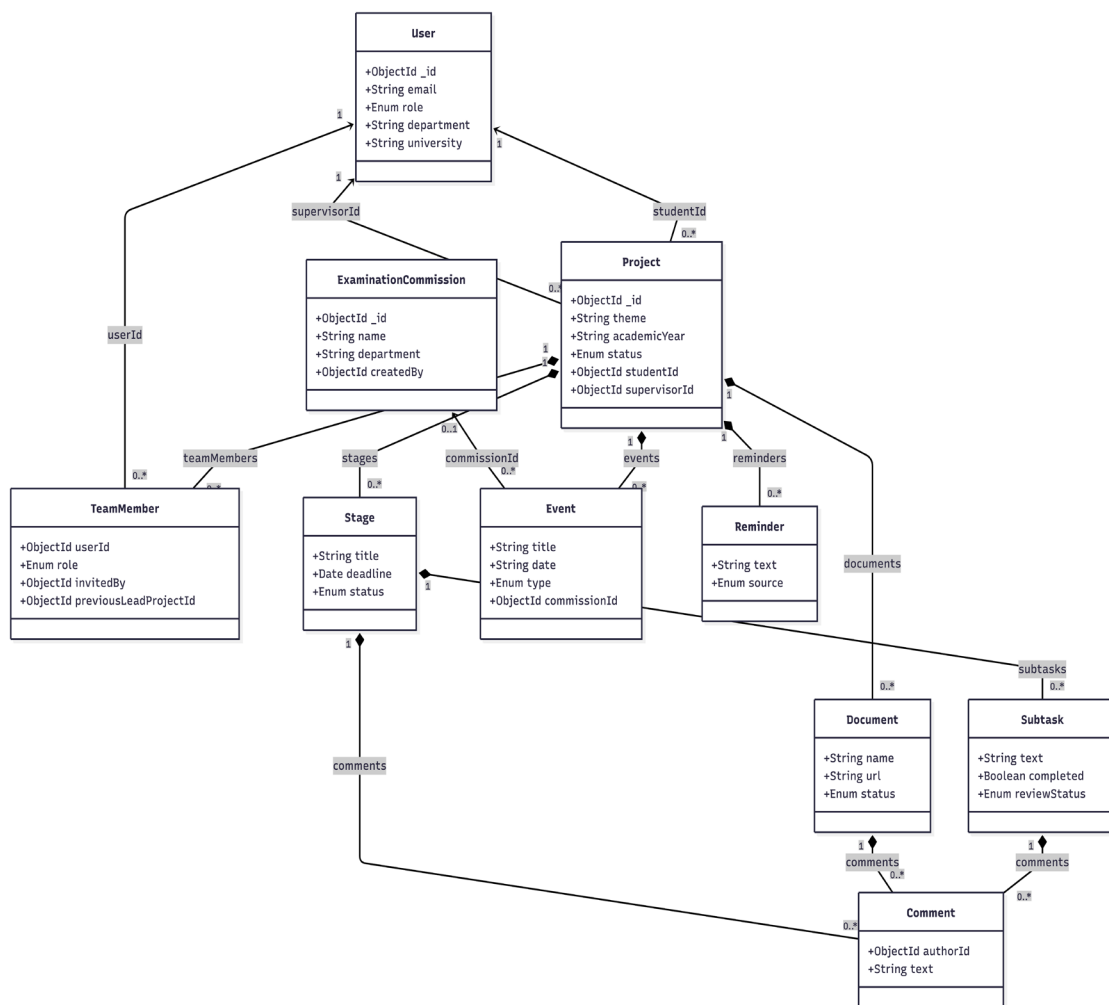


Рисунок 3.4 - UML-діаграма основних сутностей бази даних веб-сервісу

Для кращого розуміння структури бази даних у таблиці 3.4 наведено основні колекції системи та їхнє функціональне призначення. Кожна колекція відповідає за окрему частину логіки веб-сервісу та підтримує єдине збереження інформації.

Таблиця 3.4 - Основні колекції бази даних веб-сервісу та їхнє призначення

Колекція	Призначення
Users	Збереження користувачів, ролей, контактів, налаштувань і службових параметрів доступу
UserSessions	Облік активних сесій користувачів

Кінець таблиці 3.4

Колекція	Призначення
Projects	Збереження дипломних проєктів, тем, етапів, документів, подій і нагадувань
ExaminationCommissions	Збереження інформації про екзаменаційні комісії
UniversityAcademicDeadlines	Збереження академічних дедлайнів
UniversityBroadcasts	Збереження масових повідомлень і оголошень
SecurityAuditEvents	Фіксація подій безпеки
BackupJobs	Облік резервного копіювання системних даних

Центральною сутністю системи є користувач, оскільки саме від ролі користувача залежить доступний функціонал веб-сервісу. У таблиці 3.5 наведено структуру основних полів колекції Users та їхнє призначення.

Таблиця 3.5 - Структура та призначення полів сутності користувача (Users)

Поле	Призначення
fullName	Повне ім'я користувача
email	Електронна адреса для входу в систему
passwordHash	Хеш пароля користувача
role	Роль користувача в системі
group	Навчальна група студента
department	Кафедра користувача

Кінець таблиці 3.5

Поле	Призначення
university	Університет, до якого належить користувач
phone, telegram, personalEmail	Контактні дані
isActive	Ознака активності облікового запису
authTokenVersion	Версія токена для примусового завершення сесій
uiPreferences	Налаштування мови та теми інтерфейсу
notificationPrefs	Налаштування сповіщень
libraryItems	Матеріали та ресурси користувача

Іншою ключовою сутністю є дипломний проєкт, навколо якого побудована основна логіка системи. У таблиці 3.6 наведено основні поля колекції Projects та їхнє функціональне призначення.

Таблиця 3.6 - Структура та призначення полів сутності дипломного проєкту (Projects)

Поле	Призначення
theme	Назва теми дипломного проєкту
themeDescription	Опис теми або змісту проєкту
topicReviewStatus	Стан погодження теми
studentId	Посилання на студента
supervisorId	Посилання на наукового керівника

Кінець таблиці 3.6

Поле	Призначення
teamMembers	Учасники командного дипломного проєкту
stages	Етапи виконання дипломної роботи
documents	Завантажені документи
events	Календарні події, консультації, дедлайни та захисти
reminders	Нагадування для учасників процесу
comments	Коментарі до завдань, етапів або документів

Окрему увагу під час проєктування бази даних було приділено вкладеним структурам. Це дозволяє відобразити логіку організації дипломного проєкту, у якому кожен етап може містити підзавдання, ресурси, коментарі та документи. Наприклад, документ може мати статус перевірки, інформацію про автора завантаження та коментарі керівника. Для кращого опису таких структур у таблиці 3.7 наведено основні вкладені сутності, які використовуються в межах дипломного проєкту.

Таблиця 3.7 - Призначення вкладених сутностей (субдокументів) у структурі дипломного проєкту

Вкладена сутність	Призначення
Stage	Описує окремий етап виконання дипломного проєкту
Document	Зберігає дані про файл, його тип, статус, версію та коментарі
Reminder	Використовується для інформування користувачів

Кінець таблиці 3.7

Вкладена сутність	Призначення
Subtask	Містить конкретне завдання в межах етапу
Event	Описує календарну подію, консультацію, дедлайн або захист
Comment	Забезпечує фіксацію зворотного зв'язку між учасниками

Статуси, які використовуються в системі, дозволяють формалізувати процес виконання роботи. Наприклад, етапи можуть мати стани виконання, документи - стани перевірки, а тема - стан погодження. Завдяки цьому система може відображати актуальний стан дипломного проєкту без ручного аналізу. У таблиці 3.8 наведено приклади статусів, які використовуються для основних об'єктів системи, а також показано їхнє функціональне призначення в процесі супроводу дипломного проєктування.

Таблиця 3.8 - Статусна модель основних об'єктів системи та її функціональне призначення

Об'єкт	Приклади статусів	Призначення статусу
Тема	На перевірку, Зауваження, Прийнято	Відображення стану погодження теми
Етап	До виконання, У процесі, Виконано, Прострочено	Контроль виконання етапів роботи
Документ	Очікує перевірки, Переглянуто, Схвалено, Відхилено	Організація перевірки завантажених файлів
Подія	Активна, Виконана	Контроль календарних дій

Короткий висновок: база даних була спроектована з урахуванням предметної області дипломного проектування. Основою структури є сутності користувача та дипломного проекту, навколо яких організовано роботу з етапами, документами, подіями та коментарями. Це підтримує централізовану роботу з даними дипломного проекту.

3.4 Реалізація функціоналу всіх користувачів системи

У розробленому веб-сервісі функціональність розподілено відповідно до ролей користувачів. Такий підхід дозволяє розмежувати права доступу та відображати кожному користувачу лише необхідні інструменти для роботи з проектом. Основними ролями системи є студент, науковий керівник і адміністратор кафедри. Додатково в системі передбачено ролі адміністратора університету та суперадміністратора. Основні можливості користувачів подано в таблиці 3.9.

Таблиця 3.9 - Функціональні можливості основних категорій користувачів системи

Роль користувача	Основне призначення	Реалізований функціонал
Студент	Виконання дипломного проекту	Перегляд теми, робота з етапами, завантаження документів, перегляд дедлайнів, перегляд коментарів

Кінець таблиці 3.9

Роль користувача	Основне призначення	Реалізований функціонал
Науковий керівник	Контроль і супровід студентів	Перегляд студентів, перевірка документів, створення коментарів, контроль прогресу, надсилання нагадувань
Адміністратор кафедри	Організаційне управління процесом	Керування студентами, керівниками, темами, дедлайнами, захистами, документами та повідомленнями
Адміністратор університету	Контроль університетських процесів	Керування кафедрами, аналітика, масові повідомлення, контроль контингенту
Суперадміністратор	Системне адміністрування	Моніторинг платформи, аудит безпеки, керування університетами та системними параметрами

Для студента реалізовано повноцінний особистий кабінет із декількома функціональними розділами. Центральним елементом є головний екран, де відображається тема роботи, поточний прогрес, найближчі дедлайни, системні повідомлення та багато чого іншого. Такий інтерфейс допомагає швидко оцінити стан виконання диплому після входу до системи, рисунок 3.5.

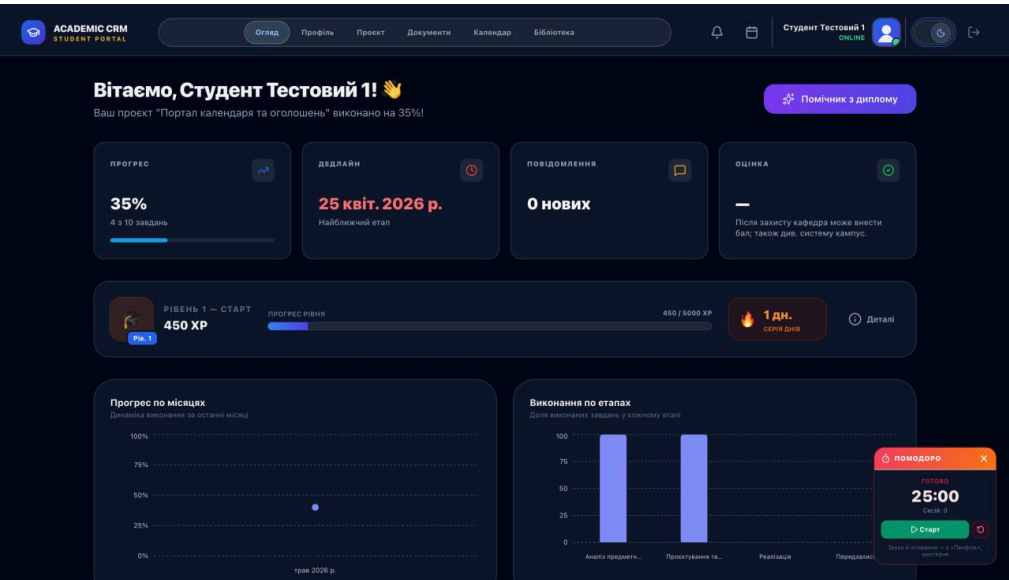


Рисунок 3.5 - Кабінет студента: головний екран (Dashboard)

Окремий розділ “Проект” призначений для роботи з етапами виконання роботи та підзавданнями. У цьому модулі студент може змінювати статуси виконання, переглядати дедлайни, додавати короткі нотатки та контролювати прогрес окремих частин проекту. Це дозволяє деталізувати процес виконання дипломної роботи та спрощує контроль проміжних результатів. Інтерфейс роботи з етапами та підзавданнями показано на рисунку 3.6.

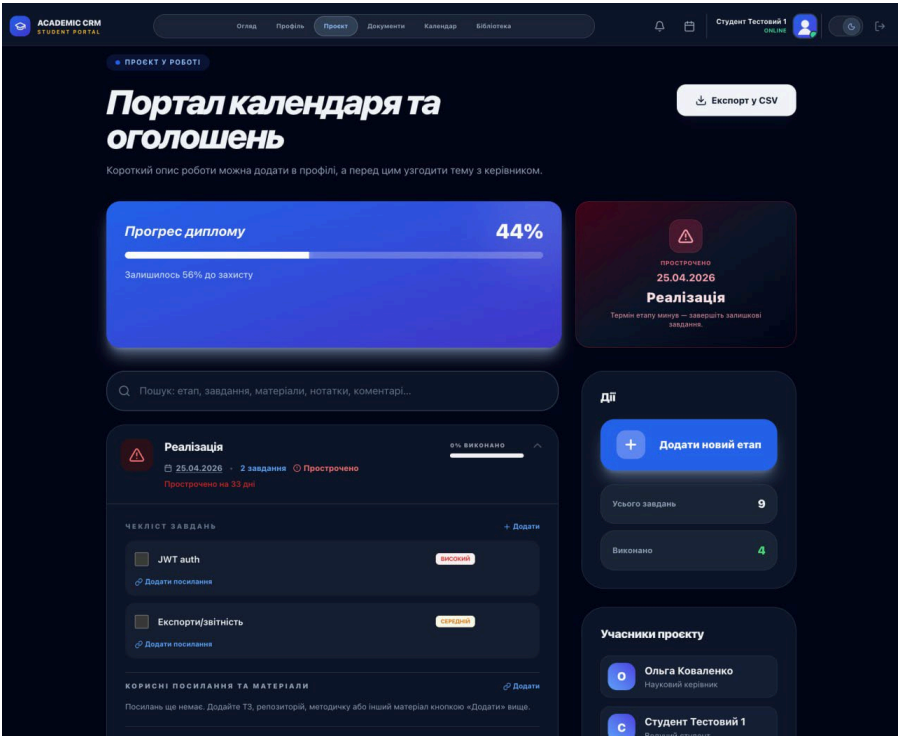


Рисунок 3.6 - Кабінет студента: розділ “Проект” (етапи та підзавдання)

Для забезпечення роботи з файлами у системі створено розділ “Документи”. Студент може завантажувати документи, переглядати статуси перевірки, бачити коментарі керівника та відстежувати історію змін. Завдяки цьому всі матеріали дипломного проєкту зберігаються в межах єдиного середовища. Інтерфейс розділу документів представлено на рисунку 3.7.

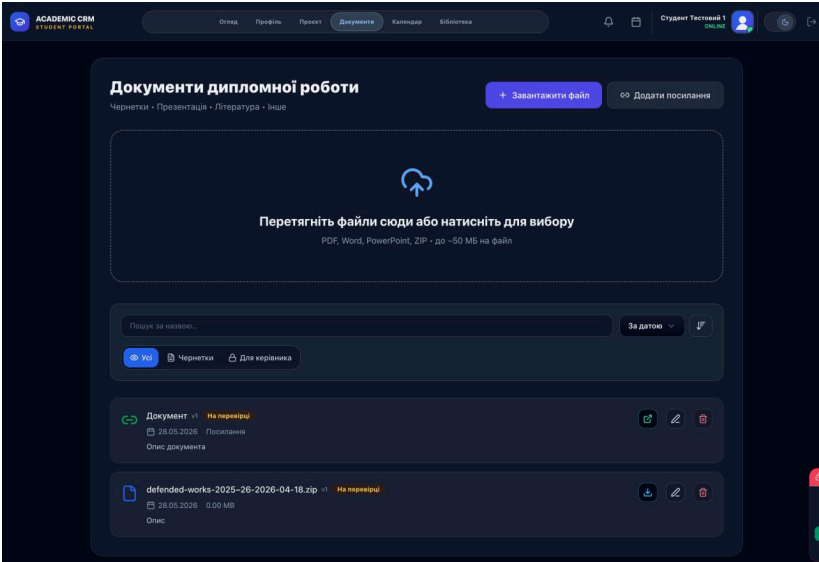


Рисунок 3.7 - Кабінет студента: розділ “Документи”

Контроль термінів виконання дипломного проєкту здійснюється за допомогою календарного модуля, який поєднує консультації, дедлайни, події захисту та інші організаційні події. Календар дозволяє студентові бачити розклад роботи над дипломним проєктом. Роботу календарного модуля в кабінеті студента показано на рисунку 3.8.

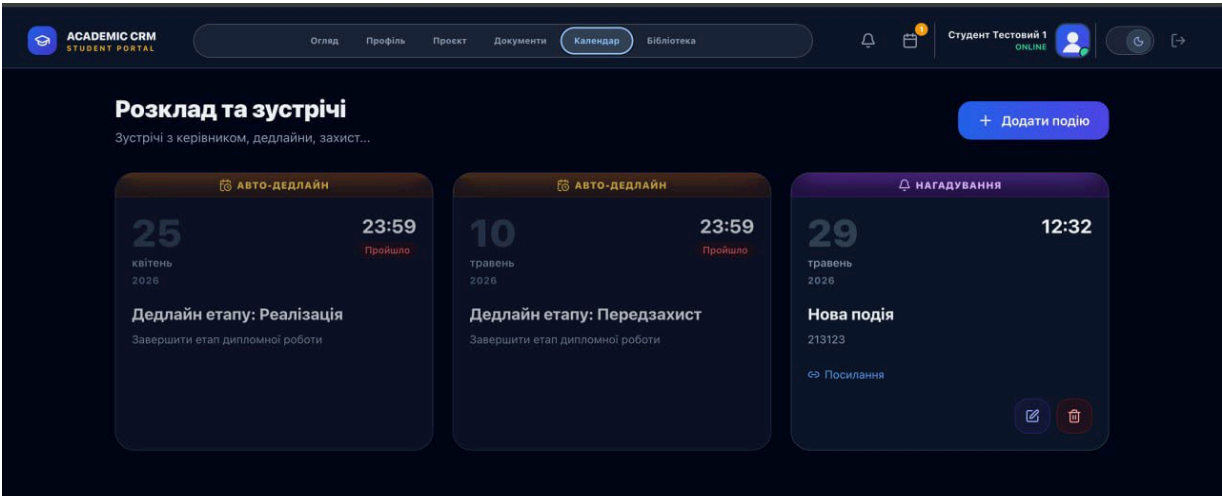


Рисунок 3.8 - Кабінет студента: розділ “Календар”

У кабінеті студента також реалізовано розділ “Бібліотека”, де доступні методичні матеріали, навчальні ресурси та інші файли, потрібні для виконання дипломного проєкту. Приклад розділу бібліотеки наведено на рисунку 3.9.

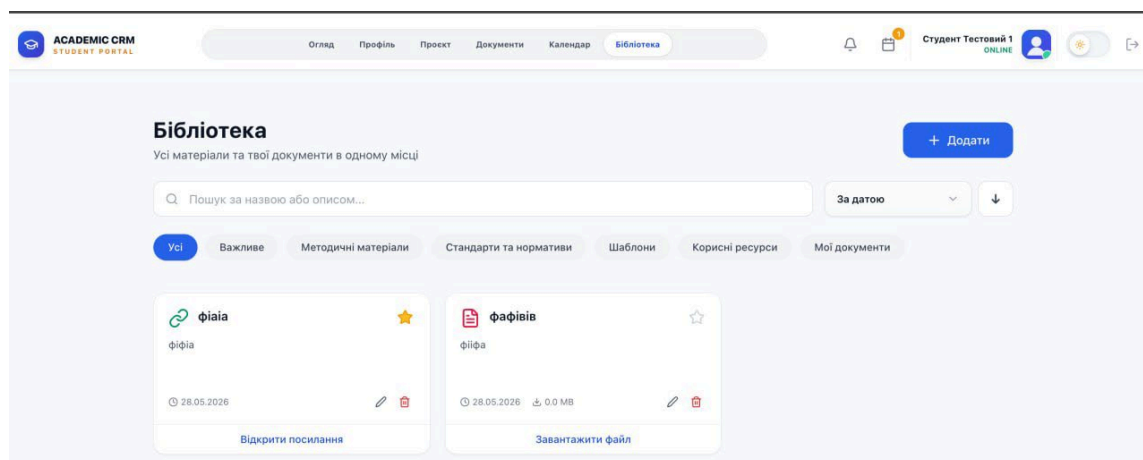


Рисунок 3.9 - Кабінет студента: розділ “Бібліотека”

У розділі “Профіль” користувач може переглядати власні персональні дані, навчальну групу, контактну інформацію та налаштування системних повідомлень. Наявність окремого профілю спрощує персоналізацію роботи із сервісом, рисунок 3.10.

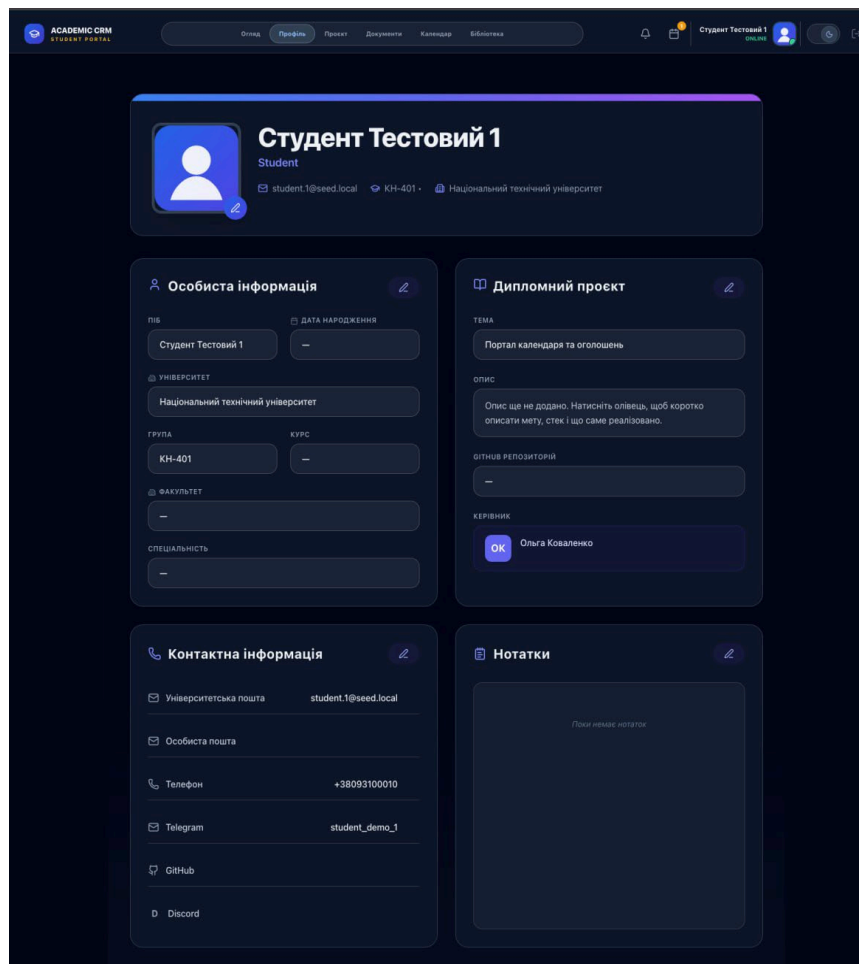


Рисунок 3.10 - Кабінет студента: розділ “Профіль”

Функціонал наукового керівника було спрямовано на контроль виконання дипломних робіт закріплених студентів. Основним елементом кабінету керівника є Dashboard, який відображає кількість студентів, стан виконання дипломних проєктів, найближчі дедлайни та інформацію про проблемні етапи. Головний екран кабінету керівника показано на рисунку 3.11.

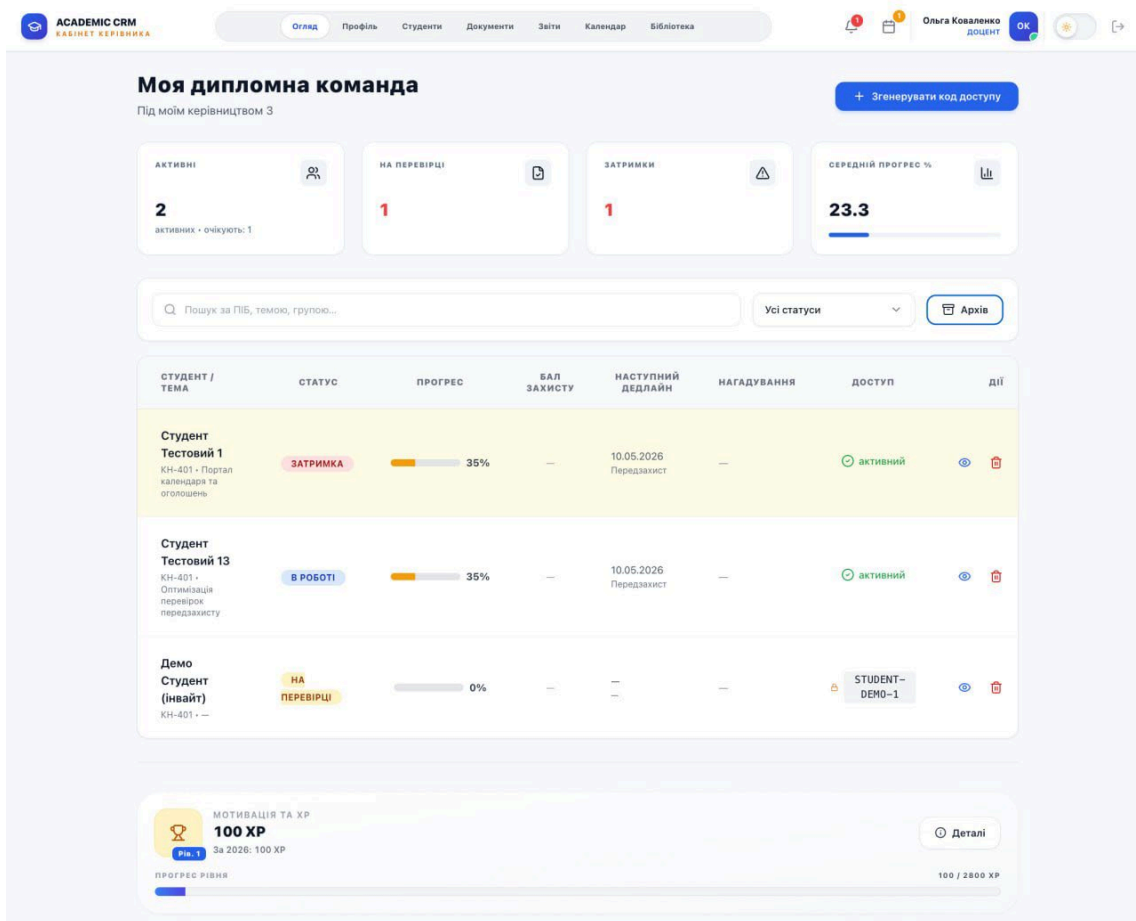


Рисунок 3.11 - Кабінет наукового керівника: головний екран (Dashboard)

Окремий розділ “Документи” дозволяє централізовано перевіряти файли студентів, встановлювати статус перевірки та залишати коментарі щодо доопрацювання матеріалів, рисунок 3.12.

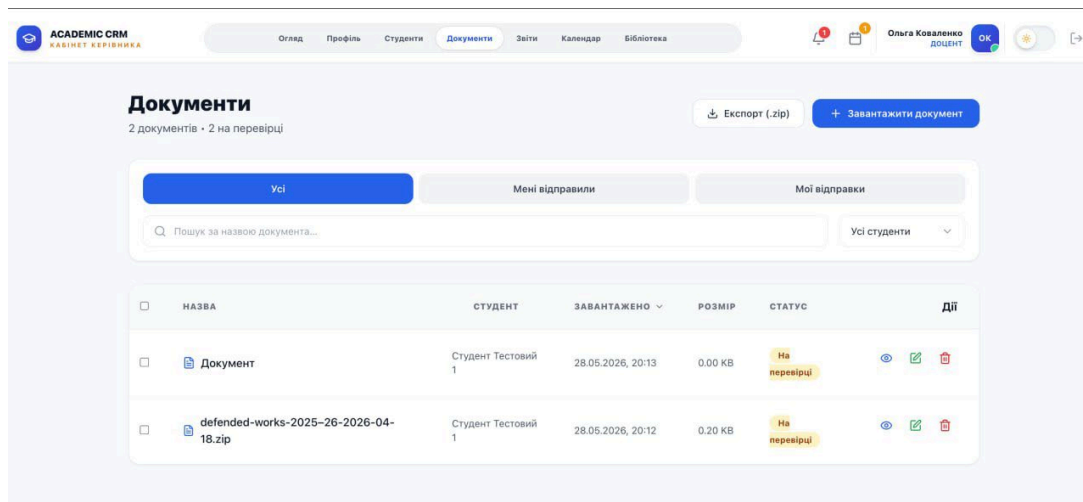


Рисунок 3.12 - Кабінет наукового керівника: розділ “Документи”

Для аналізу загального стану дипломного процесу реалізовано вкладку “Звіти”, у якому відображаються статистичні дані щодо прогресу студентів, виконання дедлайнів та кількості перевірених документів. Приклад сторінки звітності наведено на рисунку 3.13.

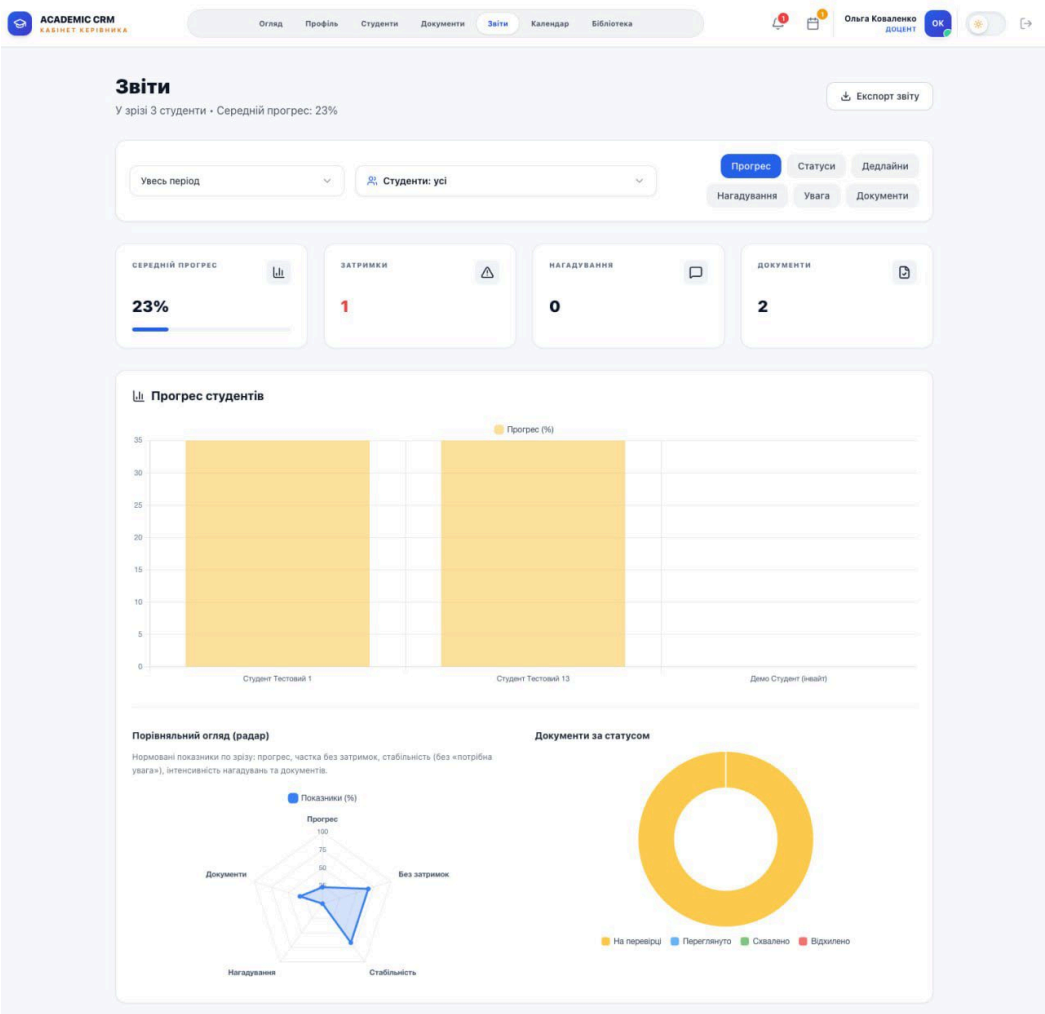


Рисунок 3.13 - Кабінет наукового керівника: розділ “Звіти”

У розділі “Студенти” керівник отримує доступ до списку закріплених дипломників із можливістю перегляду прогресу, пошуку та фільтрації за статусами виконання роботи. Інтерфейс наведено на рисунку 3.14.

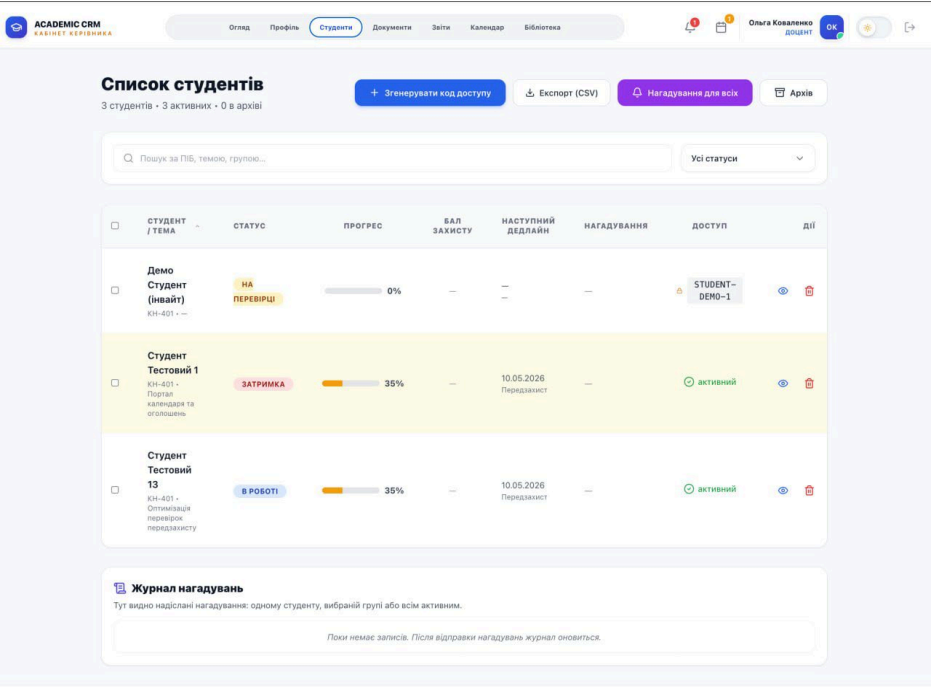


Рисунок 3.14 - Кабінет наукового керівника: розділ “Студенти”

Для детального аналізу конкретного дипломного проєкту передбачено окрему сторінку студента, де керівник може переглядати етапи, документи, коментарі та поточний стан виконання роботи, рисунок 3.15.

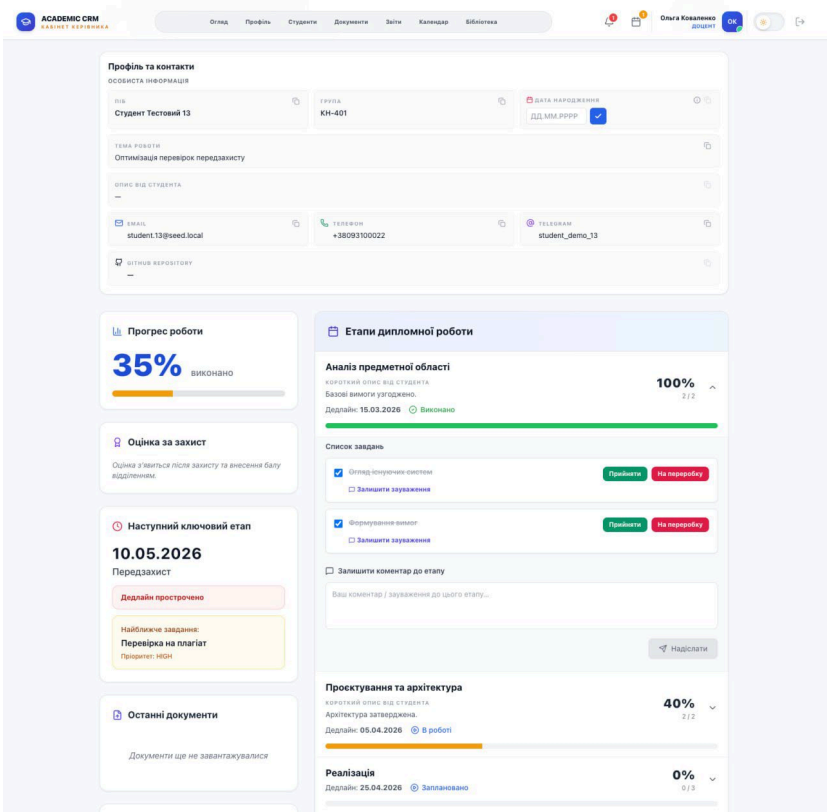


Рисунок 3.15 - Кабінет наукового керівника: картка “Деталі студента”

Календар керівника використовується для планування консультацій, контролю дедлайнів та організації зустрічей зі студентами. Наявність централізованого календаря спрощує координацію навчального процесу. Календарний модуль дозволяє керівнику контролювати консультації та терміни виконання етапів, що спрощує координацію роботи зі студентами. Цю реалізацію показано на рисунку 3.16.

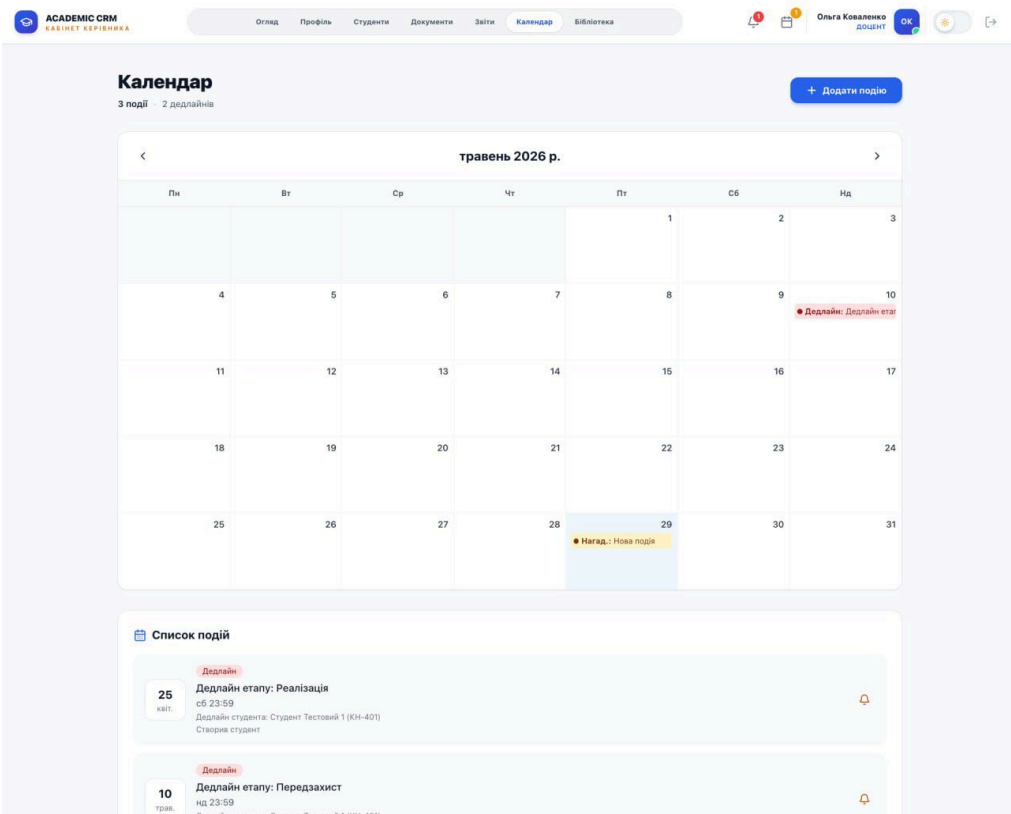


Рисунок 3.16 - Кабінет наукового керівника: розділ “Календар”

Для роботи з методичними матеріалами, шаблонами документів та додатковими навчальними ресурсами реалізовано вкладку “Бібліотека”. Цей модуль спрощує централізований доступ до файлів, необхідних під час супроводу дипломного проектування. Вигляд для наукового керівника показано на рисунку 3.17.

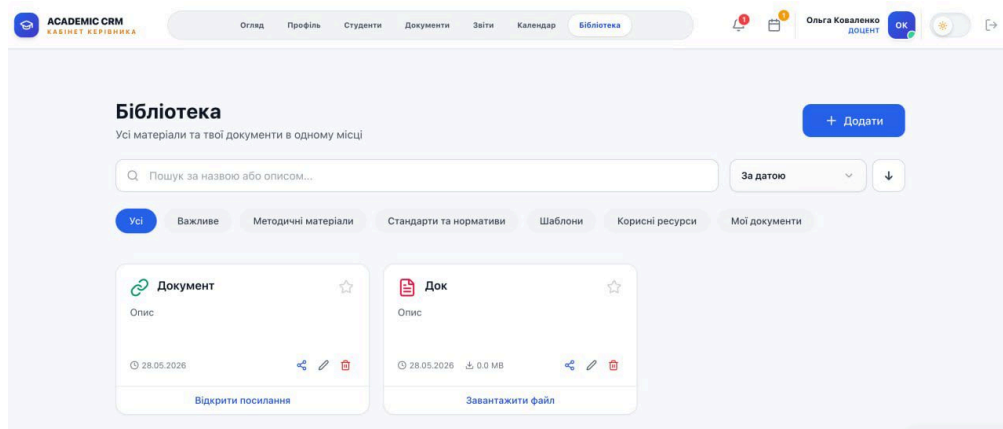


Рисунок 3.17 - Кабінет наукового керівника: розділ “Бібліотека”

Окрім засобів контролю дипломних проєктів, у системі є додаткові модулі. Зокрема, передбачено окрему сторінку профілю, яка використовується для перегляду персональних даних. Приклад вигляду наведено на рисунку 3.18.

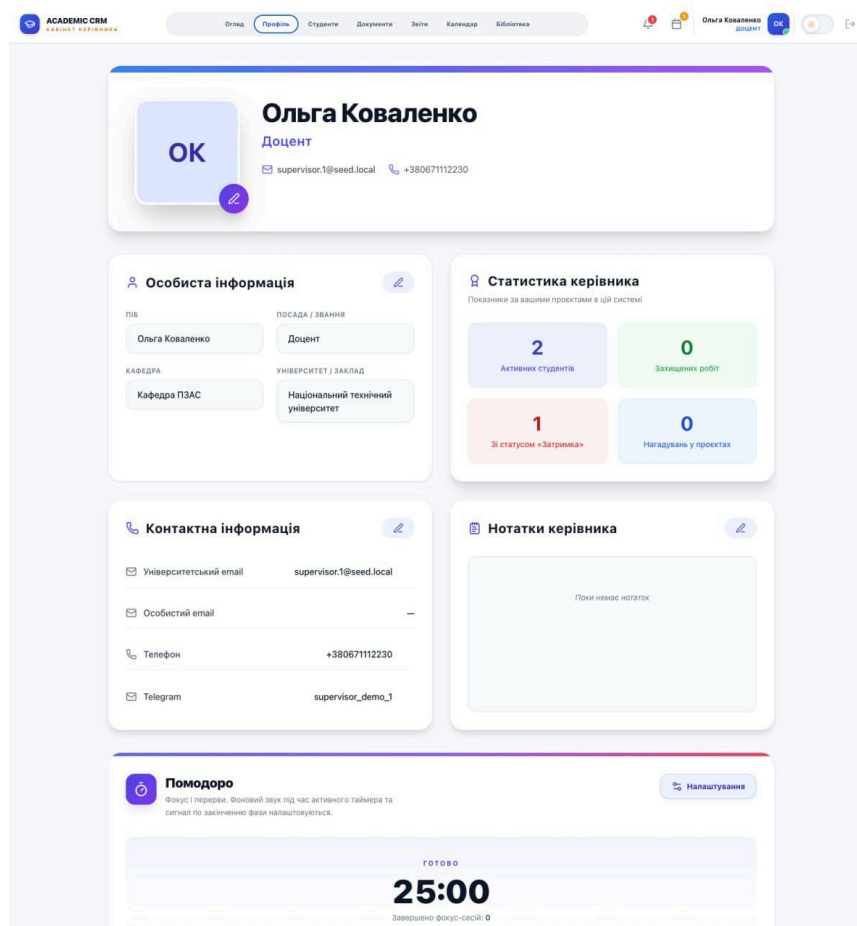


Рисунок 3.18 - Кабінет наукового керівника: розділ “Профіль”

Для узагальнення можливостей кабінету наукового керівника у таблиці 3.10 наведено основні функції та їхнє призначення.

					ІАЛЦ.467200.003 ПЗ	Арк.
						78
Зм.	Арк.	№ докум.	Підпис	Дата		

Таблиця 3.10 - Функціональні можливості кабінету наукового керівника та їхнє призначення

Функція керівника	Призначення
Перегляд списку студентів	Контроль закріплених дипломників
Аналіз прогресу	Визначення стану виконання роботи
Коментування документів	Надання зворотного зв'язку студенту
Перевірка файлів	Фіксація статусу документа
Створення нагадувань	Інформування студентів про важливі дії
Робота з календарем	Планування консультацій і контроль дедлайнів

Адміністратор кафедри має найбільш розширений функціонал для організації дипломного процесу в межах кафедри. Основний Dashboard містить зведену статистику щодо студентів, керівників, тем дипломних робіт, дедлайнів і захистів, рисунок 3.19.

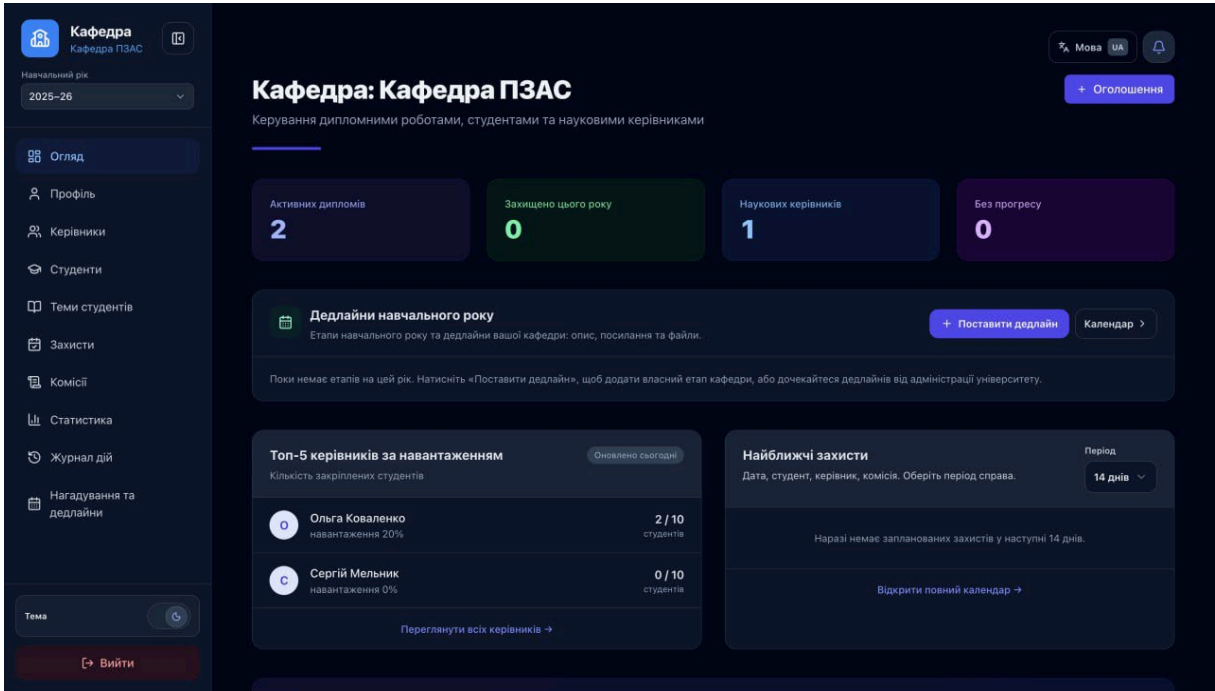


Рисунок 3.19 - Панель адміністратора кафедри: головний екран (Dashboard)

Розділ “Профіль” використовується для налаштування облікового запису адміністратора та керування персональними параметрами системи. Сторінка облікового запису та налаштуваннями системи на рисунку 3.20.

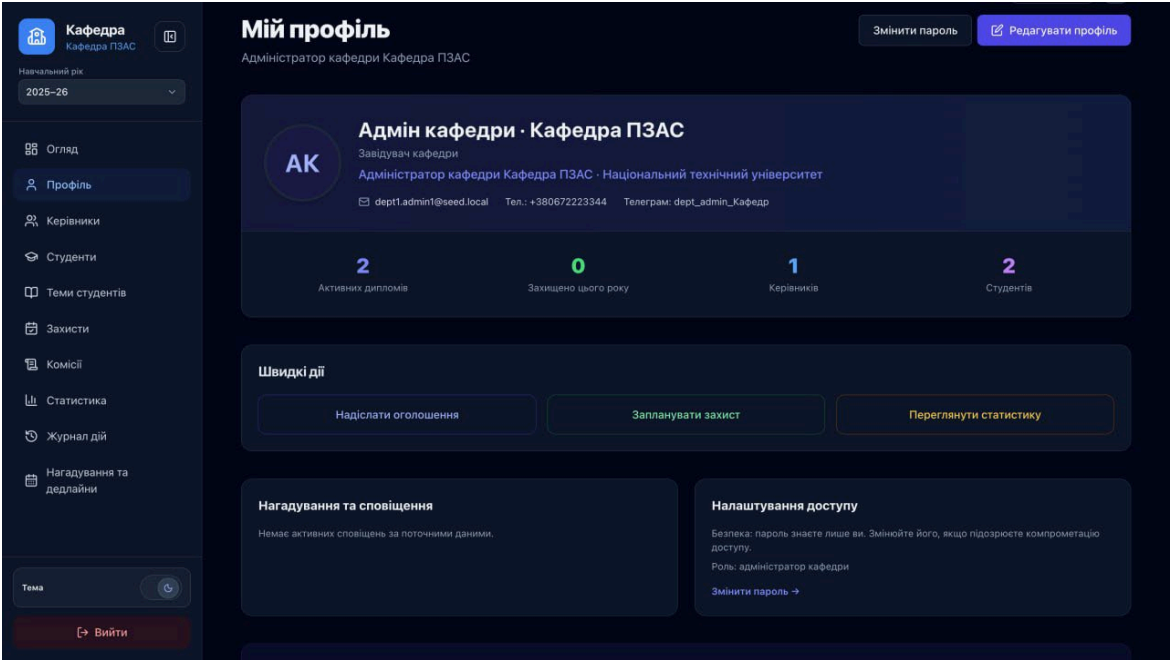


Рисунок 3.20 - Панель адміністратора кафедри: розділ “Профіль”

Для керування науковими керівниками реалізовано окремий компонент, у якому відображається список викладачів, їх навантаження та кількість закріплених студентів. Інтерфейс наведено на рисунку 3.21.

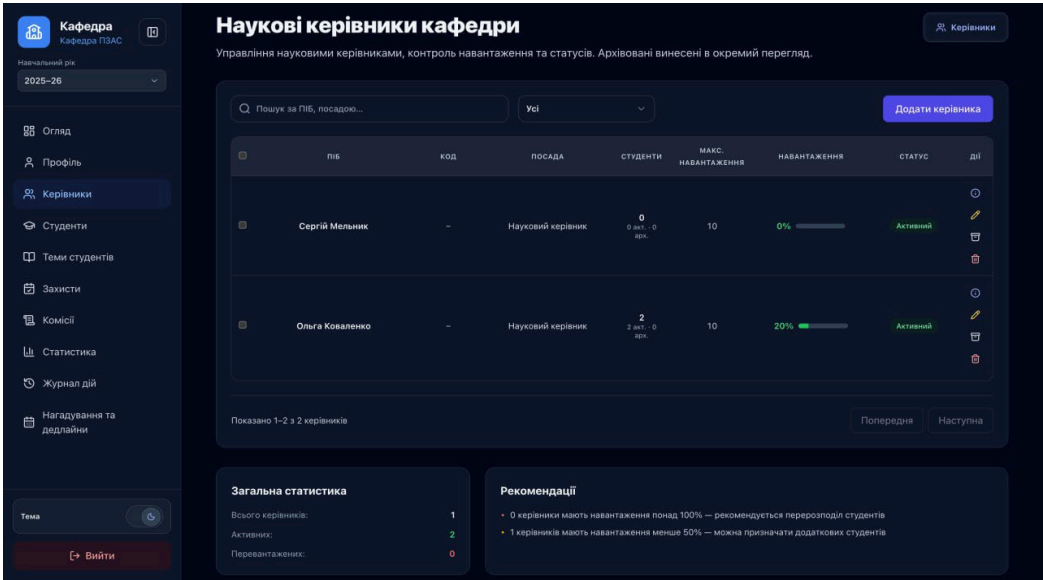


Рисунок 3.21 - Панель адміністратора кафедри: розділ “Керівники”

У вкладці “Студенти” адміністратор кафедри отримує доступ до централізованого списку дипломників із можливістю фільтрації, пошуку та виконання адміністративних дій. Приклад сторінки наведено на рисунку 3.22.

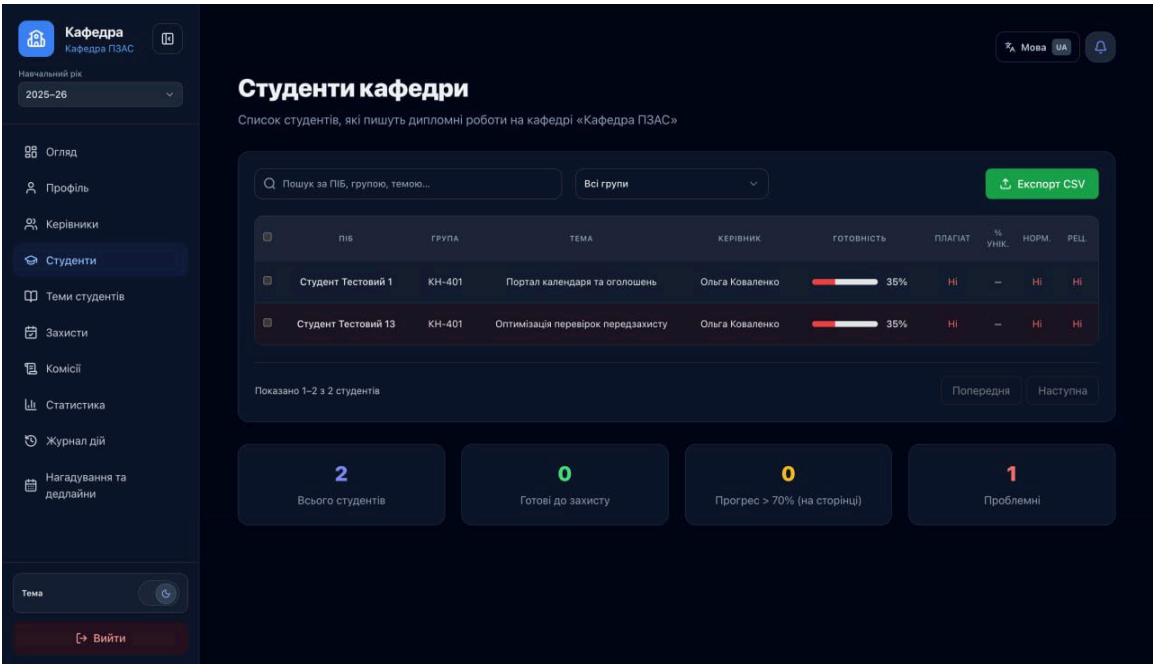


Рисунок 3.22 - Панель адміністратора кафедри: розділ “Студенти”

Для організації процесу погодження дипломних тем реалізовано окремий розділ “Теми”, який дозволяє контролювати статуси тем і призначення наукових керівників. Приклад сторінки тем наведено на рисунку 3.23.

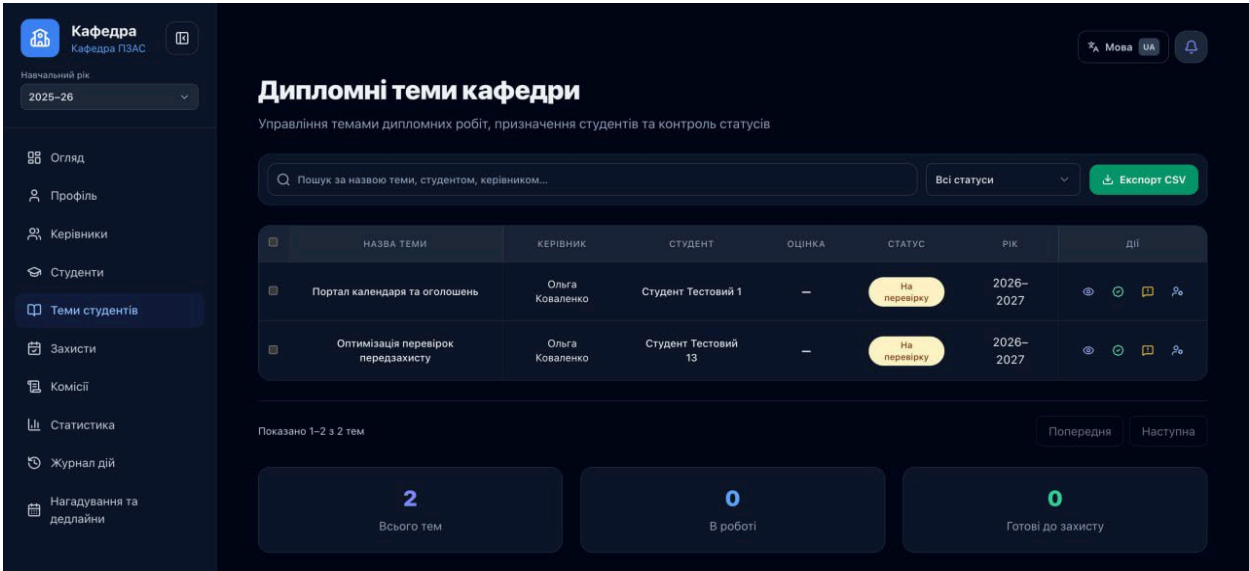


Рисунок 3.23 - Панель адміністратора кафедри: розділ “Теми”

Секція “Захисти” використовується для формування графіків захисту, призначення дат та контролю організаційних процесів, пов’язаних із проведенням захистів дипломних робіт. Інтерфейс наведено на рисунку 3.24.

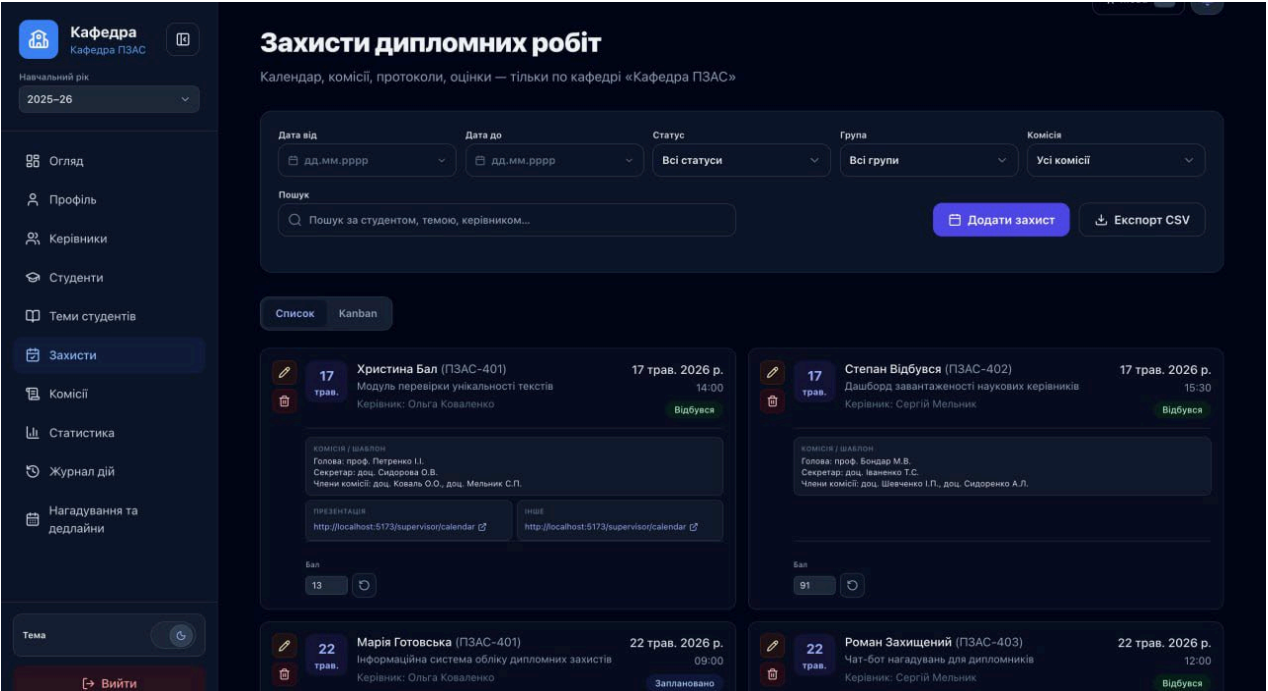


Рисунок 3.24 - Панель адміністратора кафедри: розділ “Захисти”

Для роботи з екзаменаційними комісіями передбачено окремий модуль, який дозволяє створювати та редагувати склад комісій. Приклад сторінки комісій наведено на рисунку 3.25.

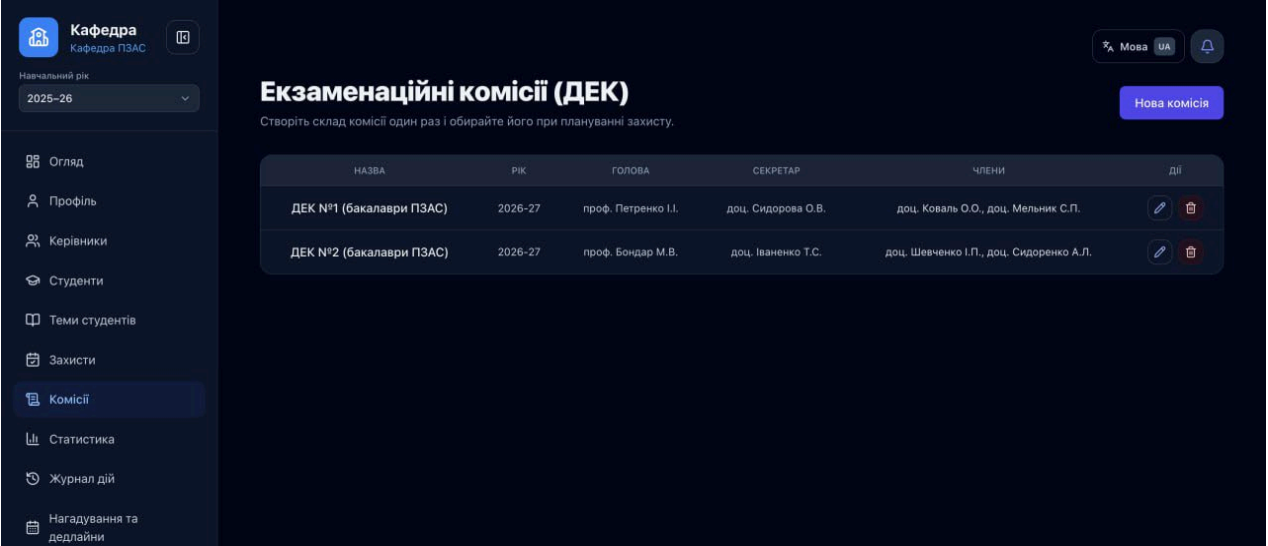


Рисунок 3.25 - Панель адміністратора кафедри: розділ “Комісії”

У вкладці “Статистика” відображаються діаграми та аналітичні показники щодо стану дипломного процесу на кафедрі. Це дозволяє адміністрації оцінювати ефективність виконання дипломних робіт і своєчасно реагувати на проблемні ситуації, рисунок 3.26.

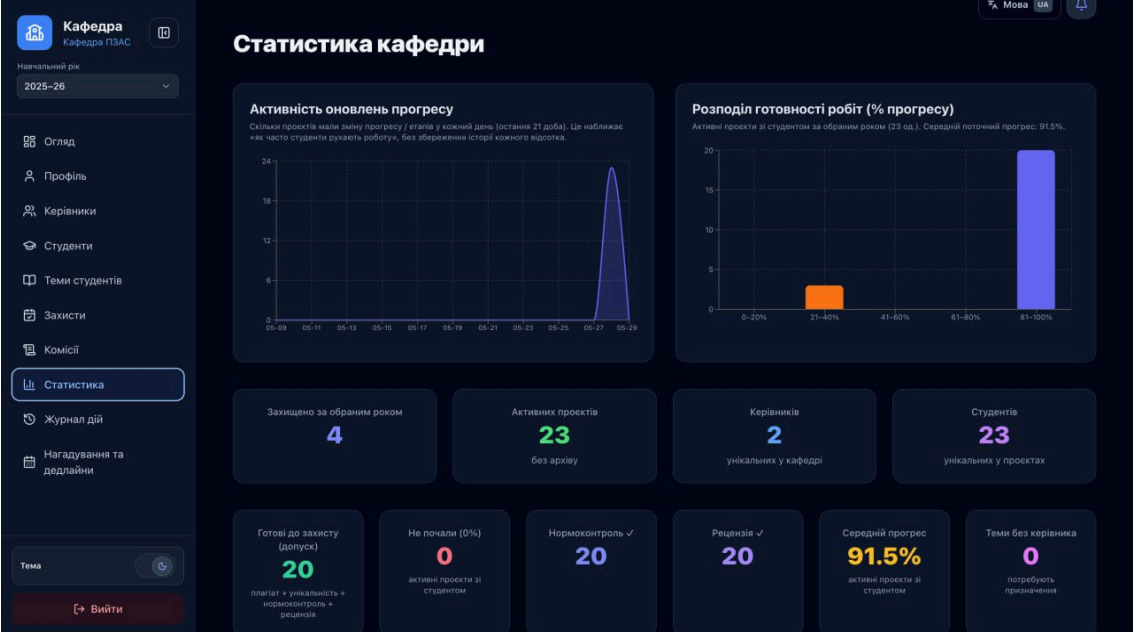


Рисунок 3.26 - Панель адміністратора кафедри: розділ “Статистика”

Для контролю подій у системі реалізовано “Журнал дій”, у якому фіксуються основні дії користувачів. Приклад журналу на рисунку 3.27.

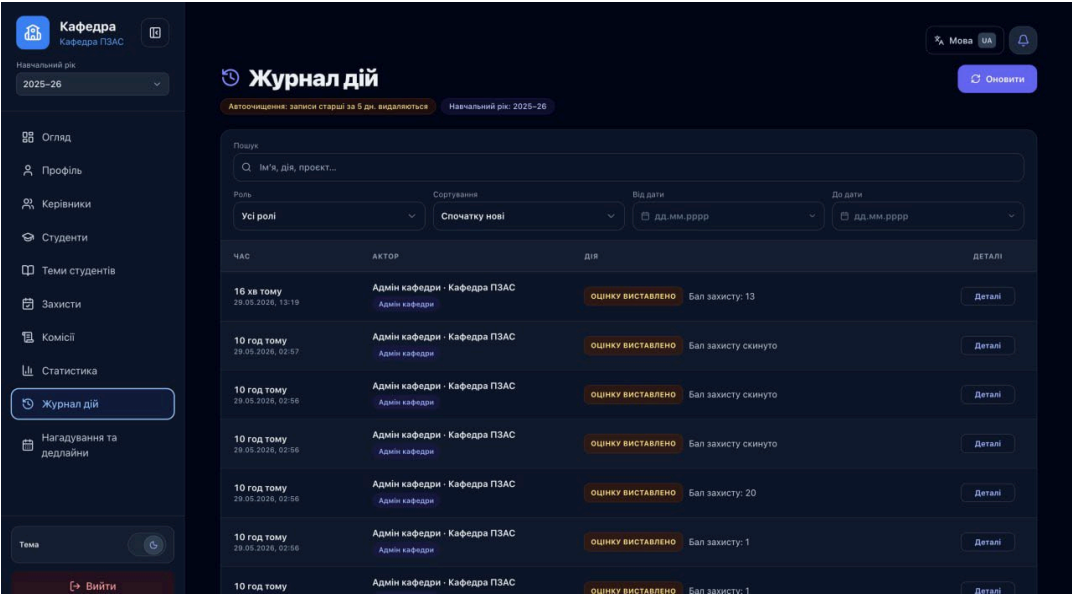


Рисунок 3.27 - Панель адміністратора кафедри: розділ “Журнал дій”

Також адміністратор кафедри має доступ до модуля нагадувань і дедлайнів, що використовується для централізованого інформування студентів та керівників про важливі події. Приклад розділу наведено на рисунку 3.28.

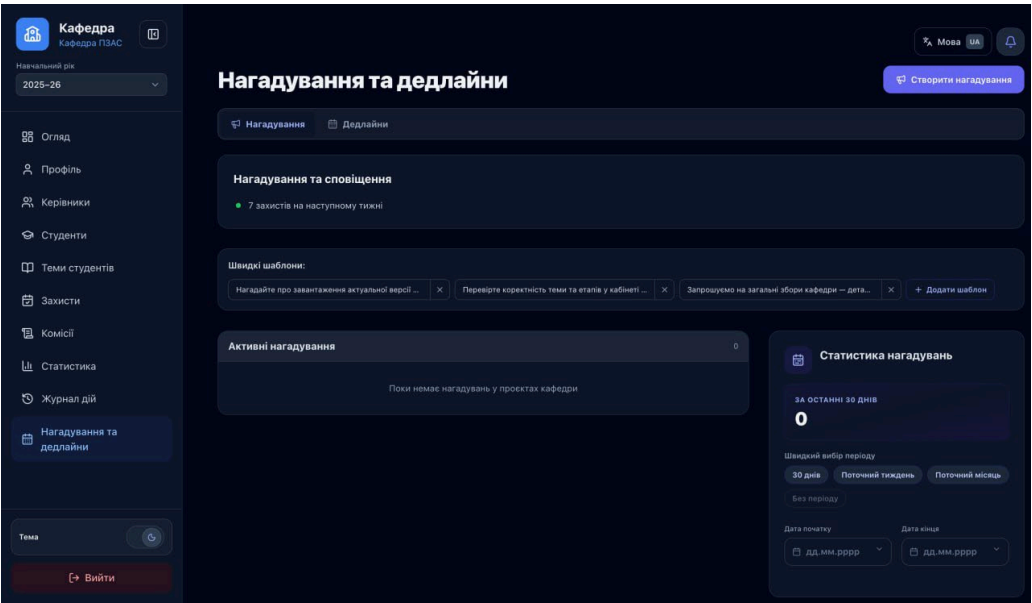


Рисунок 3.28 - Панель адміністратора кафедри: розділ “Нагадування та дедлайни”

Для університетського рівня реалізовано окремий кабінет адміністратора університету. У цьому модулі доступні засоби контролю кафедр, аналітики, комунікацій та роботи з контингентом користувачів. Головний екран адміністратора університету наведено на рисунку 3.29.

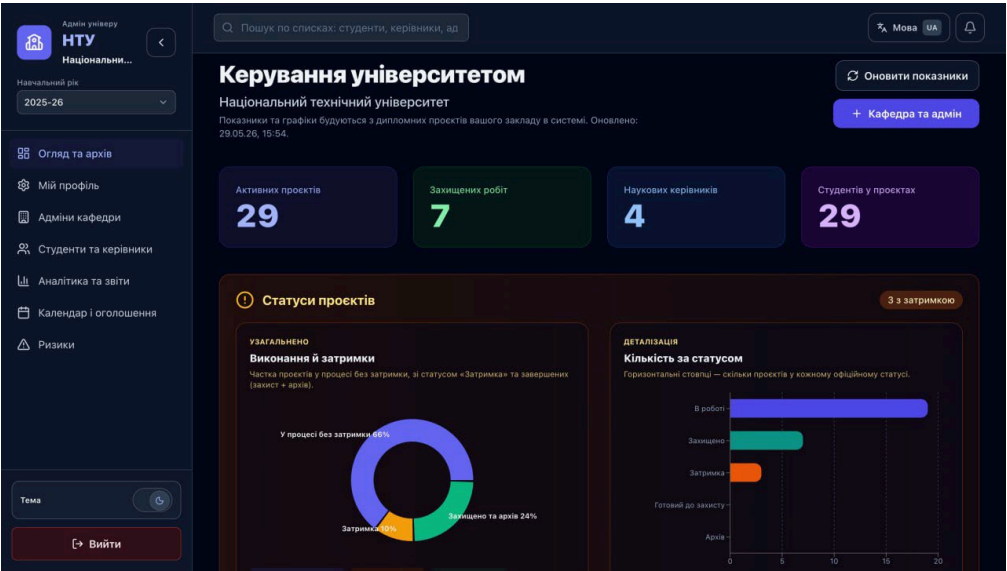


Рисунок 3.29 - Кабінет адміністратора університету: розділ “Огляд”

Сторінки профілю, кафедр, користувачів та аналітики забезпечують централізоване керування структурою закладу освіти й дозволяють аналізувати статистичні показники роботи системи. Зокрема, сторінка профілю використовується для налаштування облікового запису адміністратора університету та перегляду персональних параметрів, рисунок 3.30.

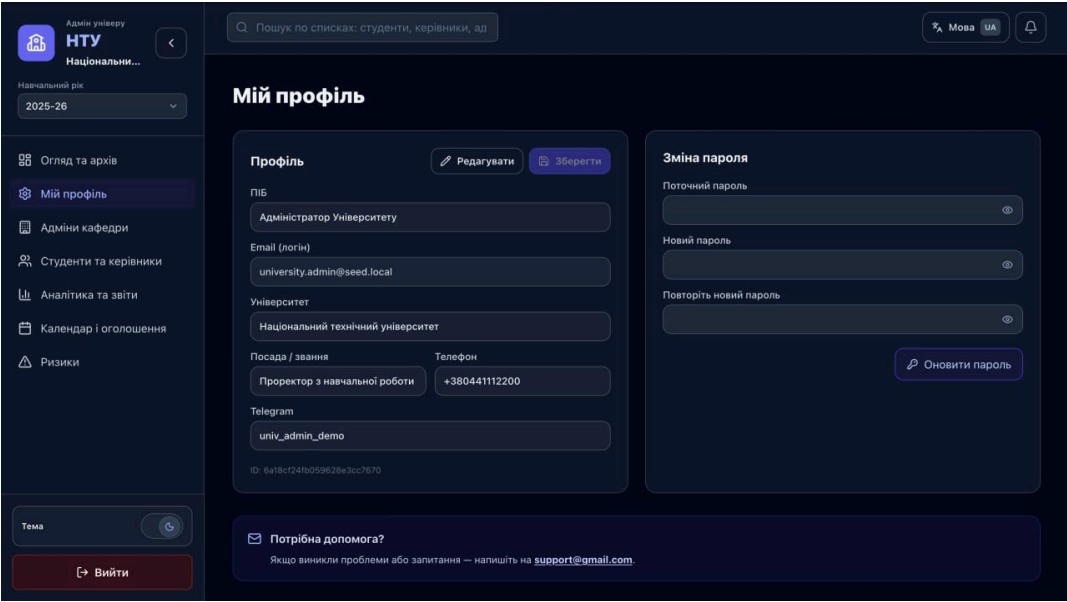


Рисунок 3.30 - Кабінет адміністратора університету: розділ “Профіль”

Для керування адміністраторами кафедр у системі зроблено модуль, який дозволяє переглядати список відповідальних осіб та контролювати їхні повноваження. Інтерфейс цього розділу наведено на рисунку 3.31.

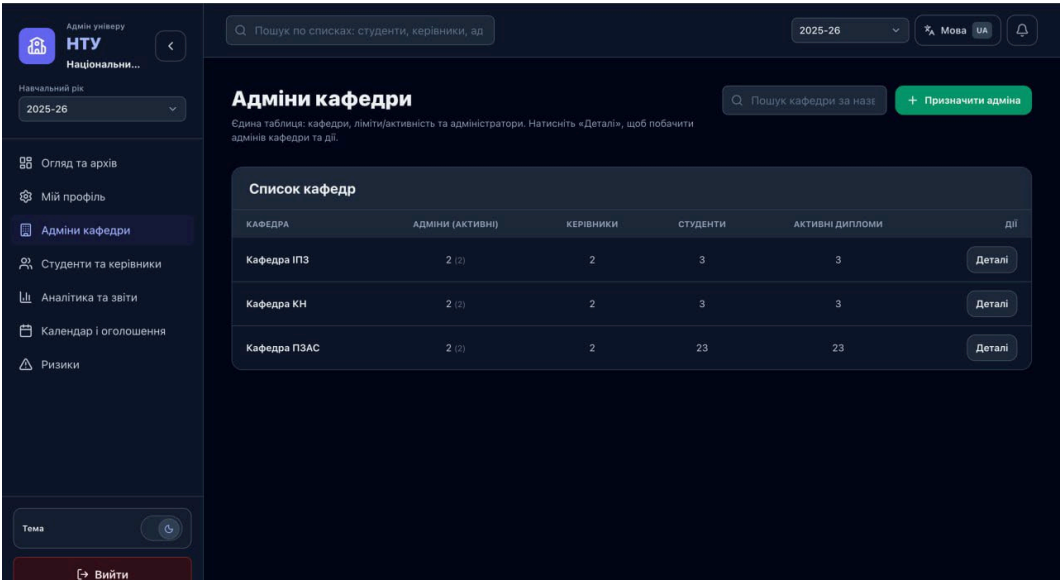


Рисунок 3.31 - Кабінет адміністратора університету: розділ “Адміні кафедр”

Окремий розділ призначений для роботи з студентами та керівниками університету. У ньому відображається інформація із можливістю пошуку, фільтрації та перегляду основних даних. Приклад цього модуля наведено на рисунку 3.32.

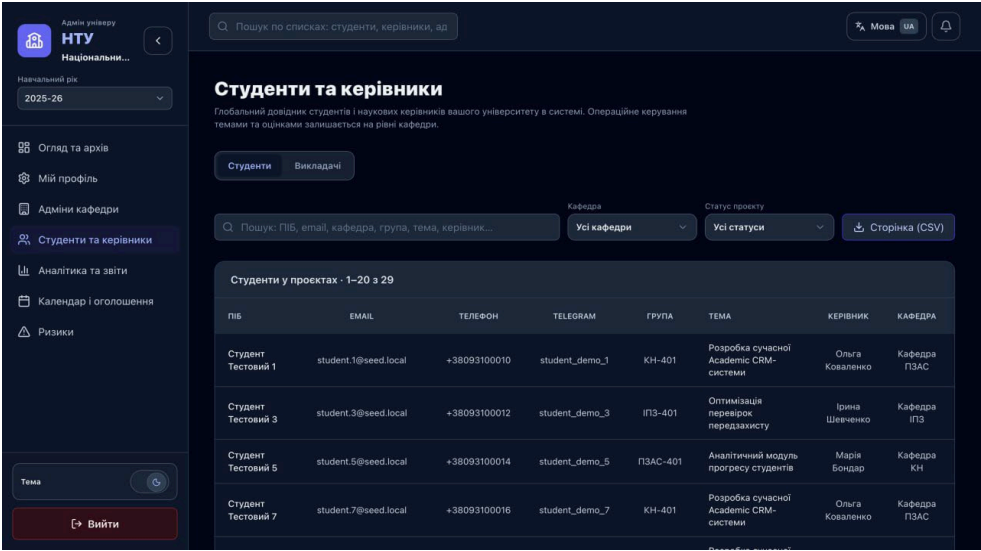


Рисунок 3.32 - Кабінет адміністратора університету: розділ “Студенти та керівники”

Для аналізу загального стану дипломного процесу реалізовано розділ “Аналітика та звіти”, приклад якого наведено на рисунку 3.33. У цьому модулі відображаються статистичні показники, діаграми та узагальнені дані щодо роботи системи на рівні університету.

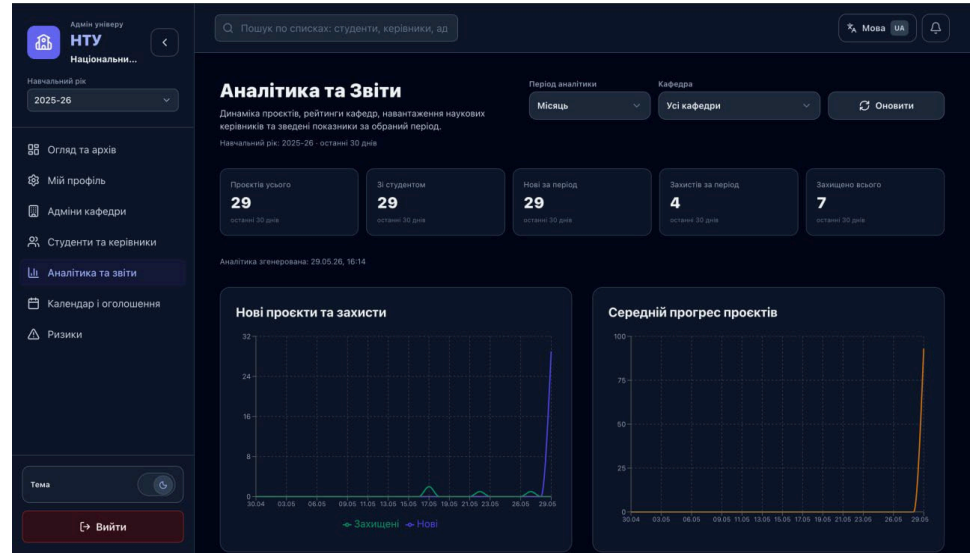


Рисунок 3.33 - Кабінет адміністратора університету: розділ “Аналітика та Звіти”

Окремі модулі “Комунікації” використовуються для формування оголошень, централізованого інформування користувачів та контролю відповідності організаційним вимогам. Приклад цього на рисунку 3.34.

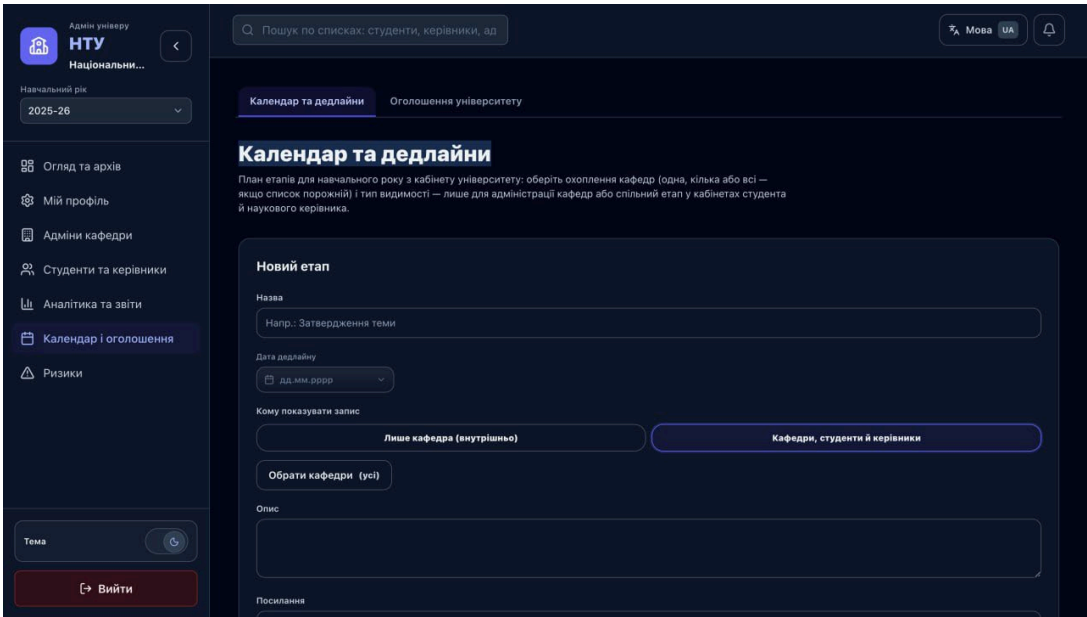


Рисунок 3.34 - Кабінет адміністратора університету: розділ “Календар та оголошення”

Для системного адміністрування реалізовано панель суперадміністратора, яка надає можливість контролювати роботу всієї платформи. Основний дашборд відображає системні показники, активність користувачів та загальний стан серверної частини, рисунок 3.35.

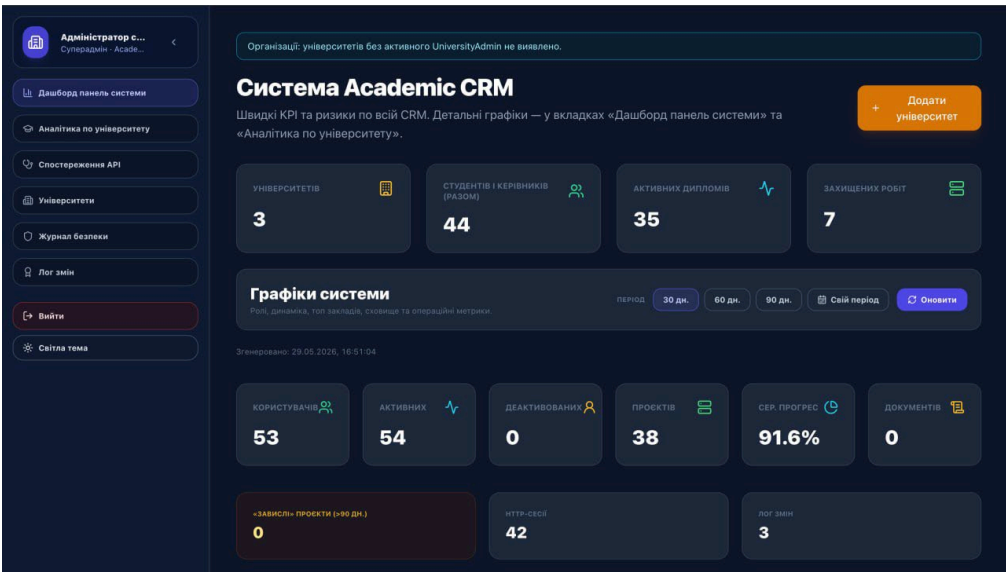


Рисунок 3.35 - Панель суперадміністратора: дашборд системної аналітики

Модуль аналітики університетів використовується для перегляду статистичних показників роботи окремих закладів освіти, аналізу активності користувачів та оцінювання загального навантаження системи. Приклад сторінки аналітики університетів наведено на рисунку 3.36.

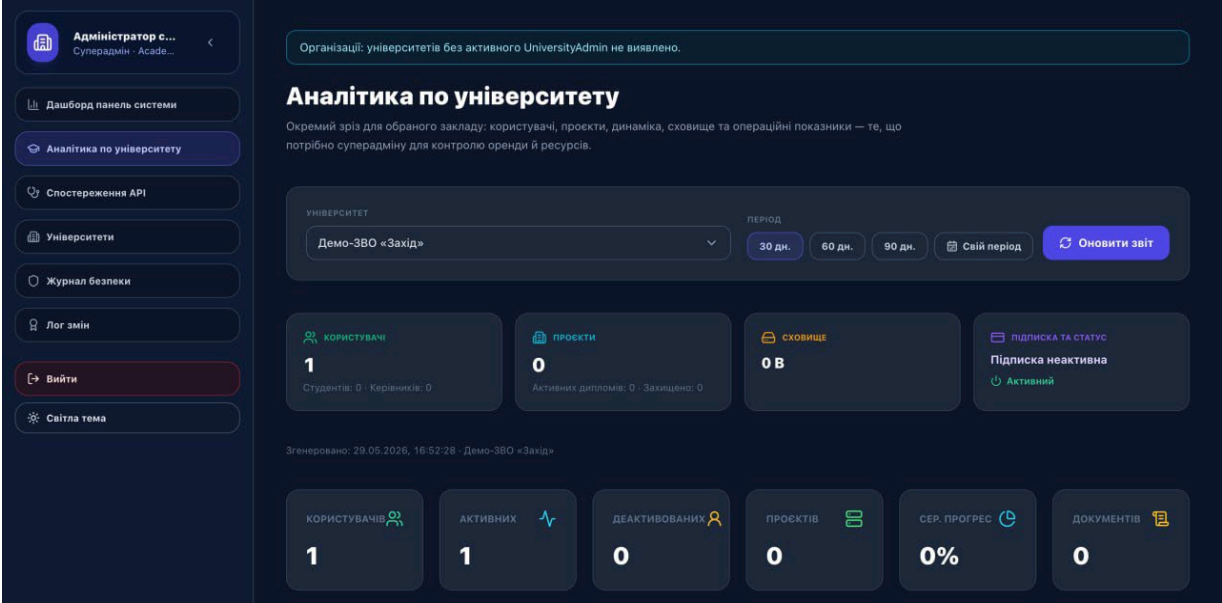


Рисунок 3.36 - Панель суперадміністратора: аналітика по університету

З метою контролю безпеки у веб-сервісі реалізовано журнал безпеки, у якому фіксуються події, пов’язані з автентифікацією, змінами доступу та іншими критичними системними діями. Приклад журналу безпеки наведено на рисунку 3.37.

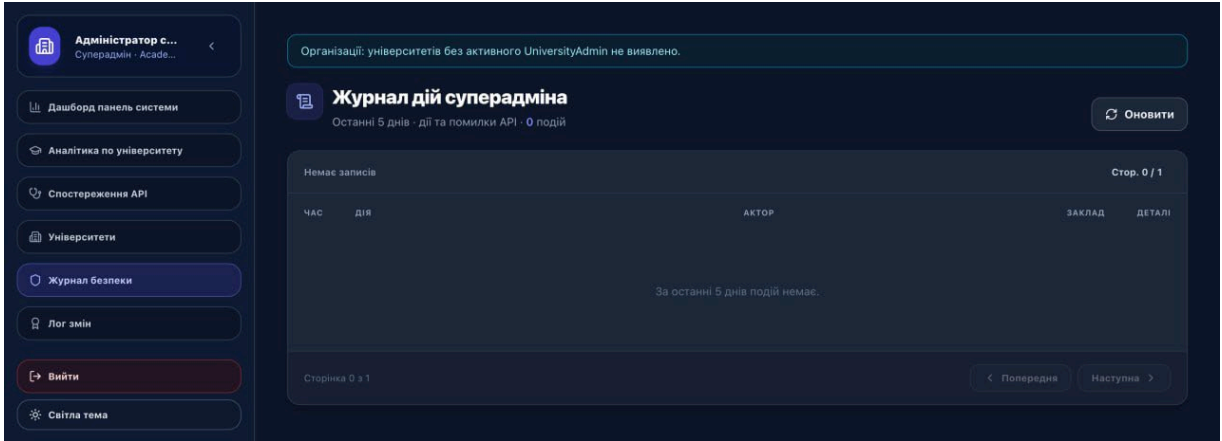


Рисунок 3.37 - Панель суперадміністратора: журнал безпеки

Для централізованого керування університетами реалізовано окремий адміністративний модуль, який дозволяє створювати, редагувати та

контролювати параметри закладів освіти, підключених до системи. Інтерфейс цього модуля наведено на рисунку 3.38.

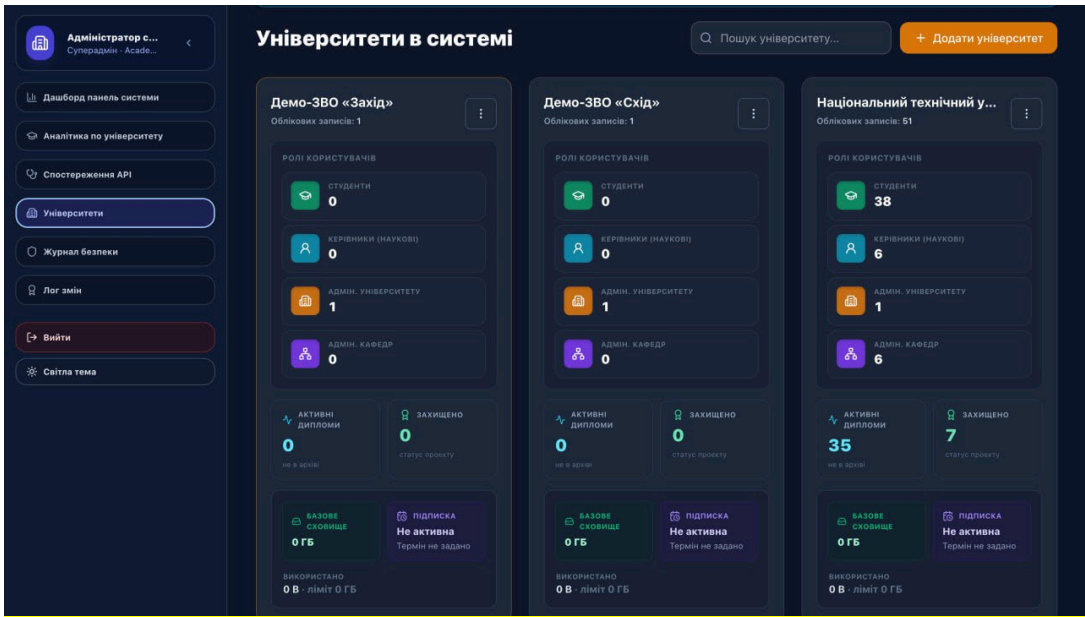


Рисунок 3.38 - Панель суперадміністратора: керування університетами

З метою контролю безпеки у веб-сервісі реалізовано журнал безпеки, у якому фіксуються події, пов’язані з автентифікацією, змінами доступу та іншими критичними системними діями. Приклад сторінки журналу безпеки наведено на рисунку 3.39.

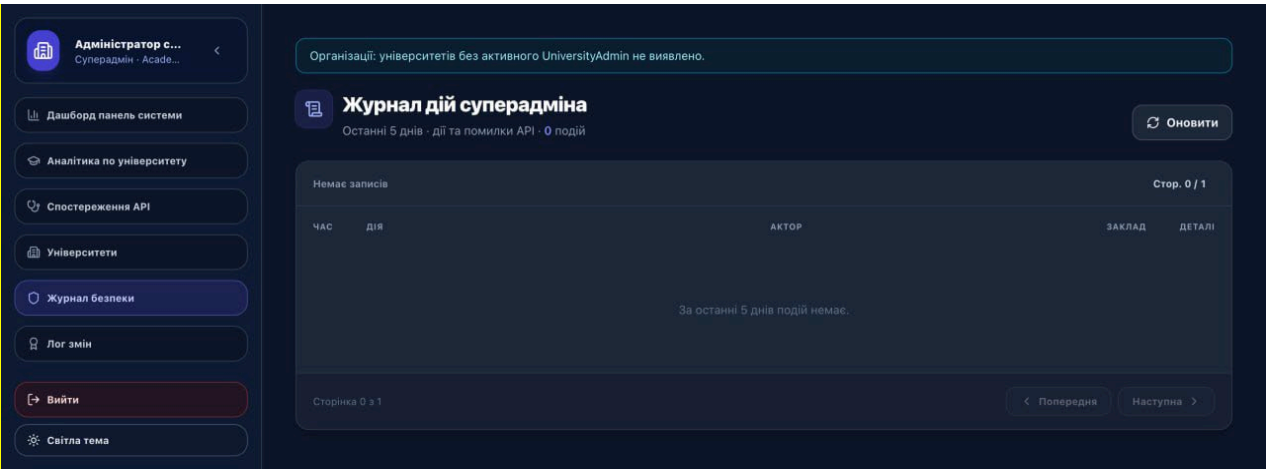


Рисунок 3.39 - Панель суперадміністратора: журнал безпеки

Для забезпечення прозорості адміністрування та можливості відстеження змін у системі реалізовано модуль логування дій користувачів і системних оновлень. Приклад сторінки журналу змін наведено на рисунку 3.40.

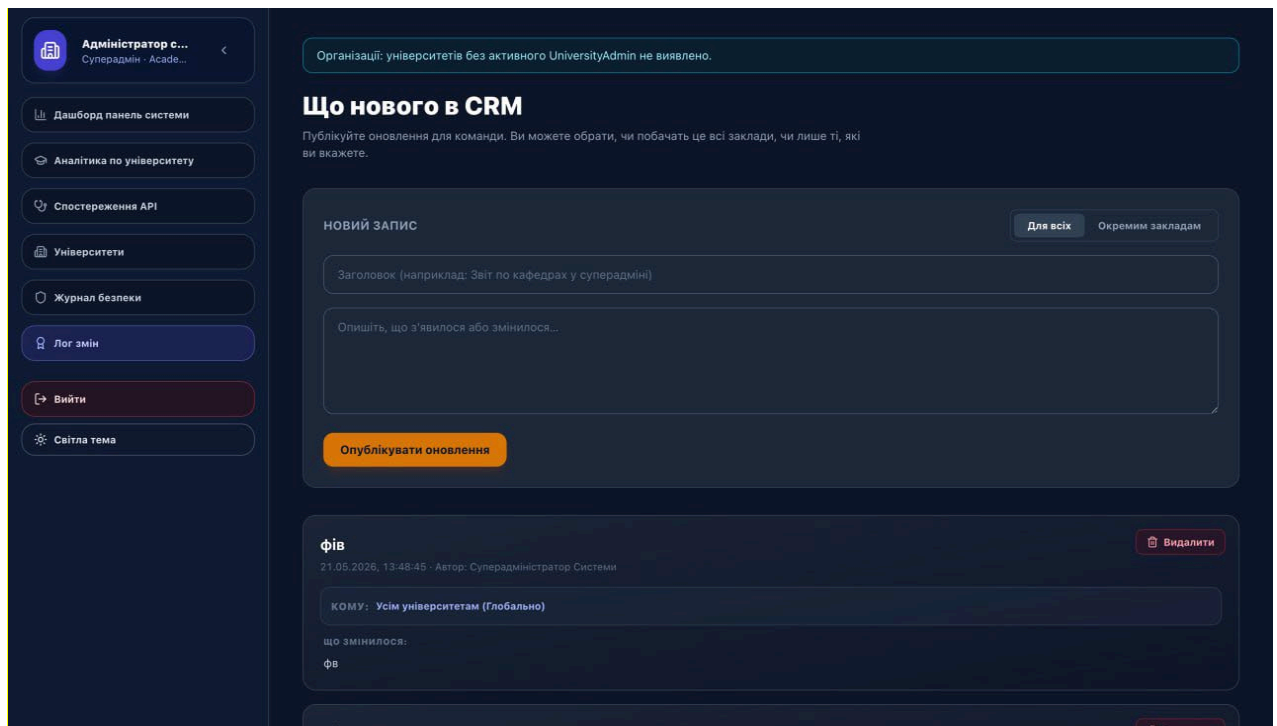


Рисунок 3.40 - Панель суперадміністратора: лог змін

Таким чином, для кожної категорії користувачів було реалізовано окремий набір функціональних модулів, що відповідає їхнім задачам у процесі дипломного проєктування. Розроблена структура інтерфейсів дозволяє забезпечити центральну взаємодію між користувачами.

Короткий висновок: реалізований функціонал відповідає ролям учасників дипломного процесу та забезпечує ефективну організацію роботи в межах єдиного веб-сервісу.

3.5 Реалізація механізмів управління дипломними темами, документами та дедлайнами

Одним із ключових компонентів веб-сервісу є механізм централізованого управління дипломним проєктуванням. У розробленому рішенні проєкт виступає центральним об'єктом, навколо якого організовано роботу з темою, етапами виконання, документами, календарними подіями, нагадуваннями та взаємодією між учасниками освітнього процесу.

Реалізований механізм роботи з темами забезпечує збереження назви дипломної роботи, її короткого опису та поточного статусу погодження. Після створення теми передбачено можливість призначення наукового керівника та закріплення її за відповідним студентом. Такий підхід спрощує організацію супроводу дипломного проєктування та забезпечує централізоване зберігання інформації.

Для структуризації роботи над дипломом передбачено поділ на окремі етапи та підзавдання. Кожен етап містить назву, опис, кінцевий термін виконання, поточний статус і показник прогресу. Це дає змогу впорядкувати виконання роботи та здійснювати поетапний контроль досягнення запланованих результатів. Основні об'єкти веб-сервісу та їхнє функціональне призначення наведено в таблиці 3.11.

Таблиця 3.11 - Основні сутності керування дипломним проєктом та їх характеристики

Об'єкт управління	Основні характеристики	Призначення
Дипломна тема	Назва, опис, статус погодження	Формалізація теми роботи
Етап виконання	Назва, дедлайн, статус, прогрес	Контроль виконання частин роботи
Підзавдання	Текст, пріоритет, стан виконання	Деталізація роботи в межах етапу
Документ	Назва, тип, файл, статус, версія	Збереження та перевірка наданих файлів

Кінець таблиці 3.11

Об'єкт управління	Основні параметри	Призначення
Подія	Назва, дата, час, тип, учасники	Планування консультацій, дедлайнів і захистів
Нагадування	Текст, джерело, адресати	Інформування користувачів

У системі реалізовано механізм централізованої роботи з документами. Під час завантаження файл автоматично прив'язується до відповідного дипломного проєкту та зберігається із зазначенням автора, дати створення та статусу перевірки. Для кожного документа може зберігатися коментар наукового керівника, що забезпечує підтримку зворотного зв'язку між учасниками процесу.

Перевірка документів виконується відповідно до визначеної послідовності дій: після завантаження документа студентом керівник отримує доступ до файлу, аналізує його вміст та встановлює відповідний статус перевірки. У разі необхідності документ може бути повернений на доопрацювання із додаванням коментарів щодо виправлень. Узагальнену схему процесу перевірки документа наведено на рисунку 3.41.

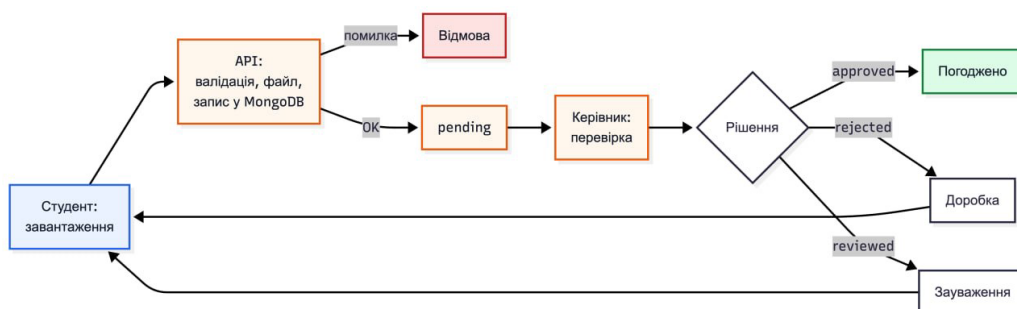


Рисунок 3.41 - Блок-схема процесу завантаження та перевірки документа

Для контролю термінів виконання дипломного проєкту в системі реалізовано календар, який використовується для планування консультацій,

дедлайнів та інших важливих подій. Події відображаються користувачам відповідно до їхньої ролі. Основні типи календарних подій в таблиці 3.12.

Таблиця 3.12 - Типологія календарних подій веб-сервісу та їхнє функціональне призначення

Тип події	Призначення
Консультація	Планування зустрічі студента з керівником
Дедлайн	Контроль строку виконання певного етапу
Захист	Організація підсумкового представлення дипломної роботи
Нагадування	Повідомлення про майбутню дію
Інша подія	Фіксація додаткових організаційних активностей

Реалізовані механізми дозволяють забезпечити супровід дипломного проєкту, єдине збереження матеріалів та контроль виконання основних етапів роботи. Завдяки автоматизації процесів взаємодії між освітніми учасниками підвищується ефективність організації дипломного проєктування.

Короткий висновок: реалізовані механізми управління темами, документами та дедлайнами забезпечують супровід дипломного проєкту та дозволяють централізовано контролювати його виконання.

3.6 Розробка інтерфейсу користувача та адаптація системи для зручної роботи

Інтерфейс веб-сервісу було розроблено з урахуванням потреб різних категорій користувачів. Оскільки системою користуються студенти, керівники та адміністратори кафедри, структура інтерфейсу має бути зрозумілою, логічно поділеною та адаптованою до різних сценаріїв.

Використання компонентного підходу дозволило уніфікувати зовнішній вигляд елементів інтерфейсу та зменшити дублювання коду. Повторно використовувані компоненти застосовуються в різних ролях користувачів, що спрощує подальший розвиток системи та внесення змін до інтерфейсу. Окремі елементи інтерфейсу, такі як таблиці, форми, модальні вікна, календарі, навігаційні панелі, картки профілю та елементи керування, були винесені у самостійні компоненти. Це спрощує підтримку коду та забезпечує повторне використання інтерфейсних рішень у різних частинах системи. Основні елементи користувацького інтерфейсу та їх функціональне призначення узагальнено в таблиці 3.13, що відображає ключові UI-компоненти системи та їх роль у взаємодії користувача з платформою.

Таблиця 3.13 - Основні елементи користувацького інтерфейсу веб-сервісу та їхнє функціональне призначення

Елемент інтерфейсу	Призначення
Особистий кабінет	Відображення основної інформації користувача
Панель навігації	Перехід між розділами системи
Календар	Перегляд дедлайнів, консультацій і захистів
Таблиці	Відображення списків студентів, керівників, документів і тем
Модальні вікна	Редагування даних без переходу на окрему сторінку
Повідомлення	Інформування про зміни, нагадування та системні події

Наведені в таблиці 3.13 елементи формують основу користувацького інтерфейсу веб-сервісу. Їх поєднання забезпечує швидкий доступ до функціональних можливостей системи, спрощує навігацію між розділами та зменшує кількість дій, необхідних для виконання типових операцій.

Для роботи з даними в системі активно використовуються форми та модальні вікна, що дозволяє редагувати інформацію без переходу між окремими сторінками. Це спрощує взаємодію користувача із сервісом і робить інтерфейс більш зручним для щоденної роботи. Приклад модального вікна наведено на рисунку 3.42.

Рисунок 3.42 - Модальне вікно для створення події в календарю студента

Окрему увагу під час розробки було приділено адаптивності інтерфейсу. Система підтримує різні розміри екранів, зокрема мобільні пристрої та планшети. Для цього використовуються адаптивні компоненти навігації, мобільне меню та перебудова структури сторінки залежно від ширини екрана. Приклад мобільної версії інтерфейсу наведено на рисунку 3.43.



Рисунок 3.43 - Мобільний інтерфейс веб-сервісу

Для покращення взаємодії користувача із системою реалізовано модуль повідомлень і сповіщень, який використовується для інформування про зміни статусів документів, нові дедлайни, коментарі керівника та інші події. Приклад системних повідомлень наведено на рисунку 3.44.

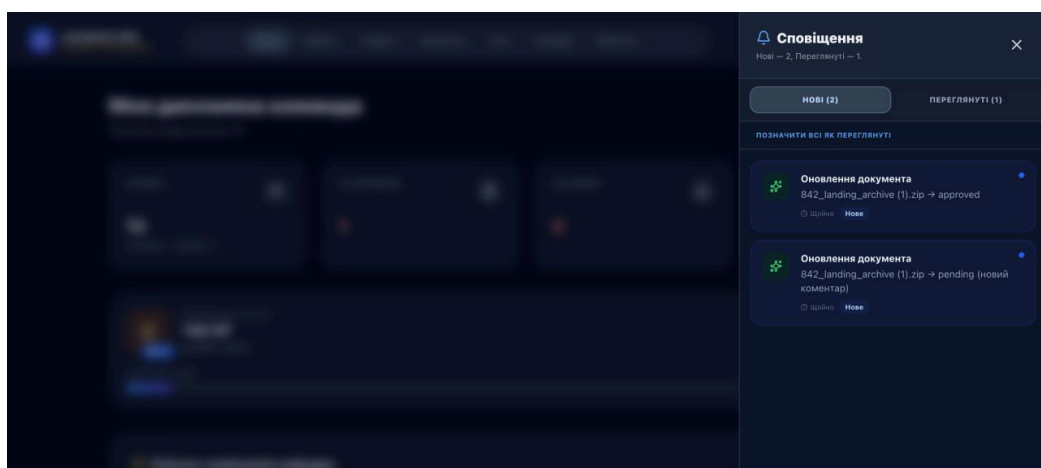


Рисунок 3.44 - Модуль сповіщень користувача

Важливим аспектом інтерфейсу є візуалізація стану дипломного проєкту. Для цього використовуються статуси, прогрес виконання, календарні позначки та інформаційні повідомлення. Такий підхід дозволяє користувачу швидко оцінити поточний стан роботи без необхідності аналізувати окремі документи чи повідомлення.

Короткий висновок: інтерфейс системи було реалізовано з урахуванням ролей користувачів, адаптивності та зручності щоденної роботи. Компонентний підхід забезпечує структурованість клієнтської рівня та подальшого масштабування.

					ІАЛЦ.467200.003 ПЗ	Арк.
						97
Зм.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ ДО РОЗДІЛУ

У третьому розділі було розглянуто проектування та реалізацію веб-сервісу для управління дипломними проєктами. Було визначено мету системи, її призначення та основні завдання, пов'язані з автоматизацією процесу супроводу дипломних робіт.

Було описано структуру системи, логіку взаємодії користувачів і основні сутності бази даних. Центральною сутністю визначено дипломний проєкт, навколо якого організовано роботу з темами, етапами, документами, дедлайнами, подіями та коментарями.

Також було розглянуто реалізацію функціоналу для студентів, наукових керівників і адміністраторів кафедри. Система використовується для розмежування доступу відповідно до ролі користувача та підтримує основні сценарії роботи кожного учасника дипломного процесу.

Отже, розроблений веб-сервіс забезпечує централізовану організацію дипломного проєктування, спрощує контроль виконання роботи та створює єдине середовище для взаємодії між студентами, керівниками й кафедрою.

					ІАЛЦ.467200.003 ПЗ	Арк.
						98
Зм.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 4. ОЦІНЮВАННЯ РЕЗУЛЬТАТІВ РОБОТИ ТА АНАЛІЗ ЕФЕКТИВНОСТІ РОЗРОБЛЕНОЇ СИСТЕМИ

4.1 Перевірка працездатності основних модулів веб-сервісу

Для оцінювання системи виконано перевірку основних функціональних модулів системи. Основну увагу було приділено модулям автентифікації, керування користувачами, роботи з дипломними проєктами, документами, календарними подіями, дедлайнами, нагадуваннями та розмежуванням прав доступу.

Перевірка працездатності виконувалася з урахуванням ролей користувачів, передбачених у системі. Такий підхід дозволяє оцінити не лише окремі функції, а й коректність взаємодії між учасниками процесу дипломного проєктування. На рисунку 4.1 подано структуру веб-сервісу та зв'язки між його основними модулями.

					ІАЛЦ.467200.003 ПЗ	Арк.
						99
Зм.	Арк.	№ докум.	Підпис	Дата		

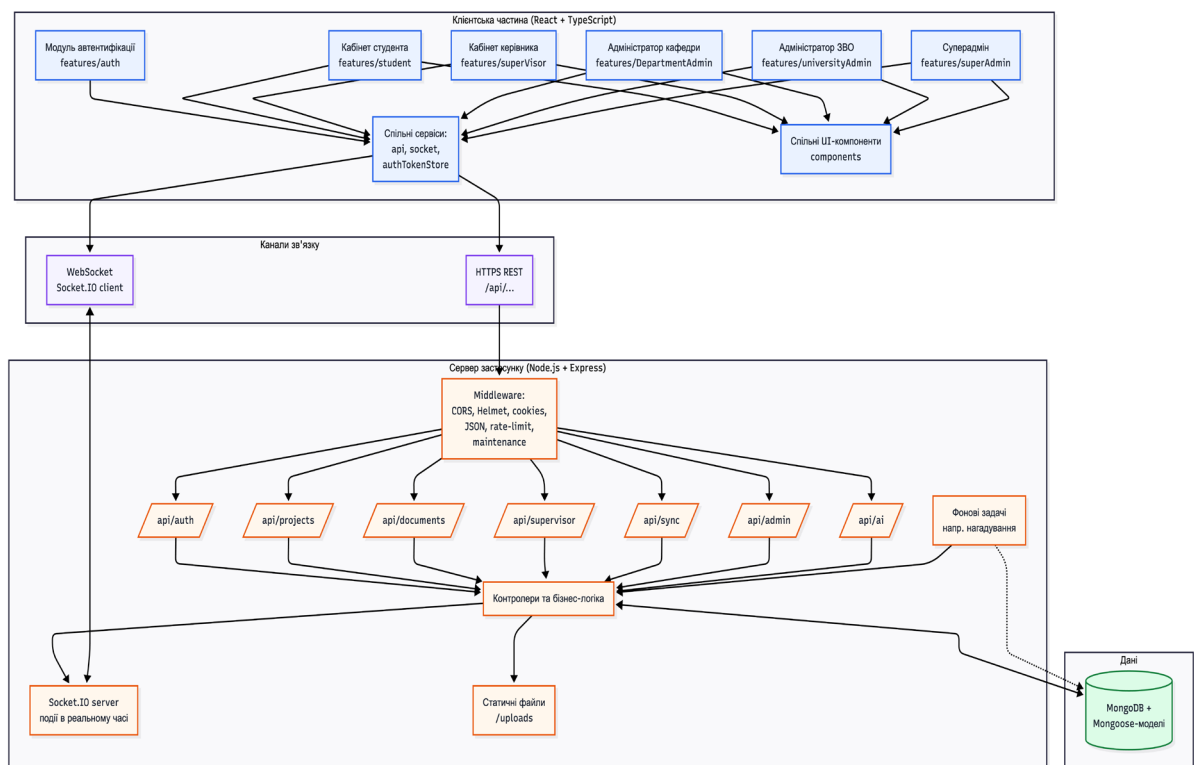


Рисунок 4.1 - Схема взаємодії основних модулів веб-сервісу

Результати функціонального тестування, наведені в таблиці 4.1, свідчать про коректну роботу всіх основних модулів системи та їх відповідність поставленим функціональним вимогам.

Таблиця 4.1 - Результати функціонального тестування основних модулів веб-сервісу

Модуль системи	Призначення	Результат перевірки
Модуль автентифікації	Вхід користувача до системи та перевірка токена доступу	Авторизований доступ виконується коректно
Модуль ролей	Обмеження функцій відповідно до ролі користувача	Доступ розмежовується між студентом, керівником і адміністратором

Кінець таблиці 4.1

Модуль системи	Призначення	Результат перевірки
Модуль дипломних проєктів	Збереження теми, опису, статусу та етапів роботи	Дані проєкту створюються, оновлюються та відображаються коректно
Модуль документів	Завантаження, збереження, перевірка та коментування файлів	Документи обробляються відповідно до заданої логіки
Модуль календаря	Відображення консультацій, дедлайнів, захистів та інших подій	Події створюються та відображаються у календарі
Модуль нагадувань	Інформування користувачів про важливі дії	Повідомлення зберігаються та доступні відповідним користувачам
Адміністративний модуль	Керування студентами, керівниками, темами та захистами	Адміністративні функції доступні відповідно до прав доступу

Аналіз результатів, наведених у таблиці 4.1, показує, що всі перевірені модулі коректно виконують передбачені функції та забезпечують стабільну роботу системи.

Для тестування веб-сервісу були використані типові ситуації, що виникають під час його використання. Результати тестування наведено в таблиці 4.2.

Таблиця 4.2 - Результати виконання базових тестових сценаріїв взаємодії користувачів із системою

№	Тестовий сценарій	Очікуваний результат	Фактичний результат
1	Авторизація юзера з коректними даними	Користувач отримує доступ до системи	Виконано успішно
2	Вхід користувача з неправильним паролем	Доступ має бути відхилено	Виконано успішно
3	Перегляд студентом власного дипломного проєкту	Відображаються лише дані власного проєкту	Виконано успішно
4	Перегляд керівником списку закріплених студентів	Відображаються лише доступні студентські проєкти	Виконано успішно
5	Завантаження документа студентом	Документ зберігається та отримує статус перевірки	Виконано успішно
6	Додавання коментаря керівником до документа	Коментар зберігається та відображається студенту	Виконано успішно
7	Створення календарної події	Подія додається до проєкту та відображається в інтерфейсі	Виконано успішно
8	Спроба доступу до чужого проєкту	Доступ має бути обмежено	Виконано успішно

Як показують результати, наведені в таблиці 4.2, усі тестові сценарії були виконані успішно. Це підтверджує коректність реалізації основних механізмів веб-сервісу та їх відповідність поставленим функціональним вимогам. У процесі тестування було встановлено, що модулі системи взаємодіють між собою відповідно до заданої логіки.

Короткий висновок: перевірка основних модулів показала, що веб-сервіс виконує ключові функції, необхідні для супроводу дипломних проєктів, зокрема авторизацію, розмежування доступу, роботу з проєктами, документами, дедлайнами та адміністративними даними.

4.2 Аналіз зручності використання інтерфейсу системи

Зручність використання інтерфейсу є важливим критерієм оцінювання веб-сервісу, оскільки система призначена для користувачів із різними ролями та різним рівнем технічної підготовки. Інтерфейс має забезпечувати швидкий доступ до основних функцій, зрозумілу навігацію, логічне групування даних і коректне відображення інформації на різних пристроях.

У розробленій системі інтерфейс побудовано з урахуванням рольової моделі. Для студента основний акцент зроблено на перегляді власного проєкту, дедлайнів, документів та коментарів. Для керівника важливими є список студентів, контроль прогресу, перевірка документів і створення нагадувань. Для адміністратора кафедри передбачено панель управління студентами, керівниками, темами, захистами та академічними подіями. Приклад реалізованого інтерфейсу студента з відображенням інформації про дипломний проєкт, прогрес виконання та найближчі дедлайни наведено на рисунку 4.2.

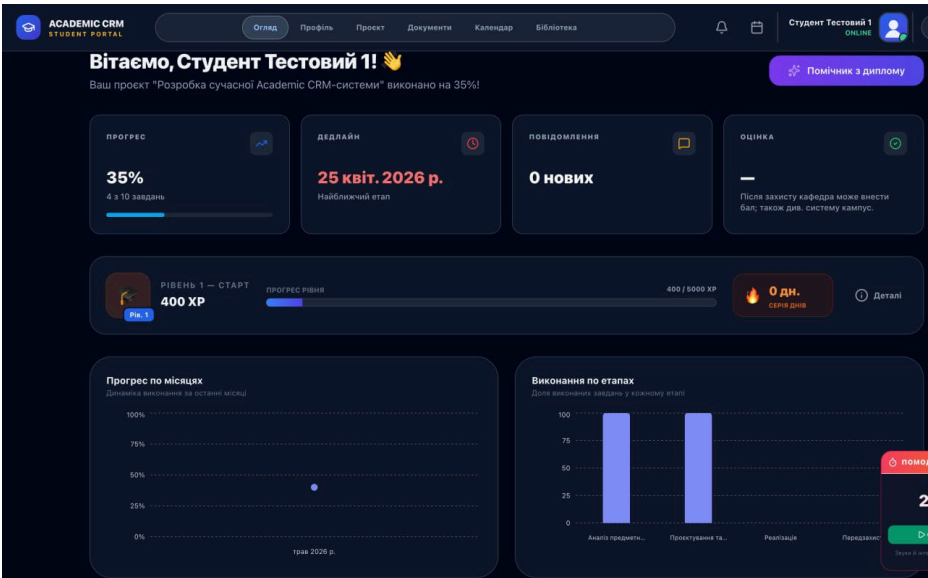


Рисунок 4.2 - Головна сторінка студента з блоками проєкту, прогресу та дедлайнів

Для оцінювання зручності використання інтерфейсу було проаналізовано його відповідність основним критеріям користувацького досвіду. Результати такого оцінювання наведено в таблиці 4.3.

Таблиця 4.3 - Оцінювання користувацького інтерфейсу веб-сервісу за критеріями зручності

Критерій	Реалізація в системі	Оцінка результату
Зрозумілість навігації	Функції розподілено за рольовими кабінетами та вкладками	Забезпечено швидкий перехід між модулями
Візуальне відображення стану роботи	Використовуються статуси, прогрес, календарні події та списки документів	Користувач може швидко оцінити стан проєкту
Адаптивність	Передбачено компоненти для мобільної навігації та модальних вікон	Система може використовуватися на різних екранах

Кінець таблиці 4.3

Критерій	Реалізація в системі	Оцінка результату
Послідовність інтерфейсу	Форми, таблиці, кнопки та модальні вікна мають єдину логіку використання	Зменшується складність навчання користувача
Інформативність	Відображаються дедлайни, повідомлення, статуси та коментарі	Користувач отримує актуальну інформацію
Розподіл за ролями	Кожна роль має власний набір доступних функцій	Інтерфейс не перевантажується зайвими діями

Дані таблиці 4.3 свідчать про те, що інтерфейс забезпечує зрозумілу навігацію, інформативне представлення даних та адаптацію до різних категорій користувачів.

Інтерфейс зроблено з окремих частин, які можна використовувати знову в різних місцях, щоб усе виглядало однаково і зручно працювало. Основні компоненти інтерфейсу та їх призначення наведено в таблиці 4.4.

Таблиця 4.4 - Основні компоненти графічного інтерфейсу вебдодатка та їхнє призначення

Інтерфейсний елемент	Призначення
Картка профілю	Відображення основних даних користувача
Календар	Відображення дедлайнів, консультацій і захистів
Панель навігації	Перехід між основними розділами

Кінець таблиці 4.4

Інтерфейсний елемент	Призначення
Таблиця студентів	Перегляд і фільтрація студентів у кабінеті керівника або адміністратора
Модальне вікно	Створення або редагування даних без переходу на іншу сторінку
Повідомлення	Інформування про зміни, зауваження або важливі події

Елементи, наведені в таблиці 4.4, використовуються для реалізації основних можливостей системи та взаємодії користувача з нею. На рисунку 4.3 показано сторінку адміністратора кафедри, що містить інструменти для пошуку, фільтрації та керування даними.

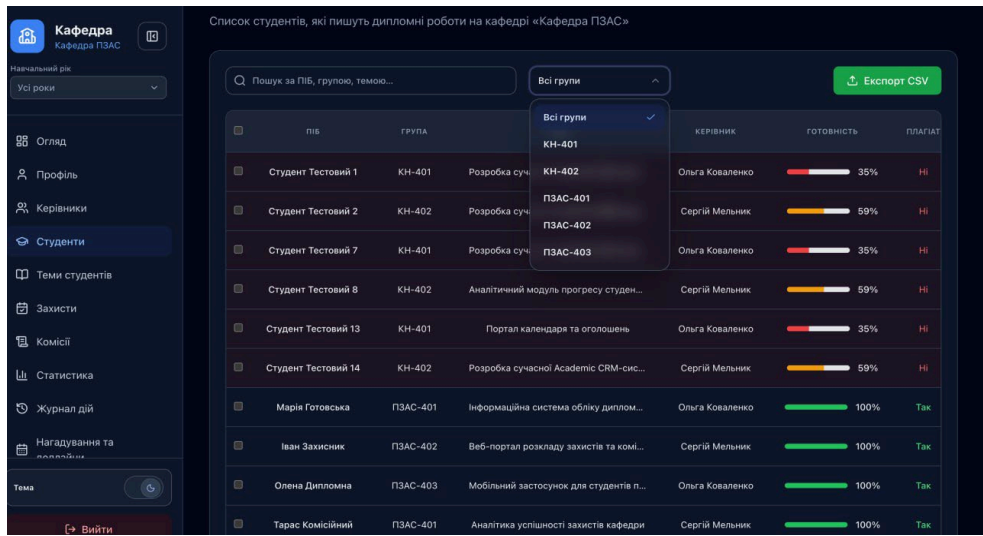


Рисунок 4.3 - Адміністративна панель кафедри для перегляду студентів і застосування фільтрів

Короткий висновок: інтерфейс системи відповідає основним вимогам зручності, оскільки функції структуровано відповідно до ролей користувачів, а основні дані подаються у вигляді таблиць, карток, календарів, статусів і повідомлень.

4.3 Оцінювання ефективності автоматизації процесу управління дипломними проєктами

Ефективність розробленого веб-сервісу визначається тим, наскільки система зменшує кількість ручних дій, підвищує прозорість контролю та забезпечує централізоване збереження інформації. У традиційному підході дані про дипломні роботи можуть зберігатися в таблицях, електронній пошті, месенджерах, локальних файлах або хмарних сховищах. Це ускладнює пошук актуальної інформації та контроль виконання завдань.

У розробленій системі основні процеси було перенесено в єдиний сервіс. Завдяки цьому користувачі працюють з актуальними даними в межах одного веб-сервісу. Для оцінювання впливу автоматизації на процес супроводу проєктів виконано порівняння традиційного підходу та функціональних можливостей веб-сервісу. Результати такого порівняння наведено в таблиці 4.5.

Таблиця 4.5 - Порівняльна характеристика традиційного підходу та автоматизованого процесу управління дипломними проєктами

Процес	Традиційний підхід	Реалізація у веб-сервісі	Очікуваний ефект
Погодження теми	Через повідомлення або окремі документи	Тема має окремий статус погодження	Підвищення прозорості процесу
Адміністративний контроль	Таблиці та ручна перевірка	Панель адміністратора кафедри	Єдиний контроль інформації

Кінець таблиці 4.5

Процес	Традиційний підхід	Реалізація у веб-сервісі	Очікуваний ефект
Контроль етапів	Ручний контроль керівником	Етапи, підзавдання, статуси та прогрес	Спрощення моніторингу
Робота з документами	Передавання документів через зовнішні засоби обміну повідомленнями	Завантаження документів у систему зі статусами	Підвищення надійності зберігання документів
Коментарі керівника	У різних каналах комунікації	Коментарі прив'язуються до документів або етапів	Збереження історії обговорень
Дедлайни	Окремі календарі або усні домовленості	Події та нагадування в межах системи	Зменшення ризику пропуску строків

Для наочного відображення відмінностей між традиційним та автоматизованим підходами побудовано діаграму, наведену на рисунку 4.4.

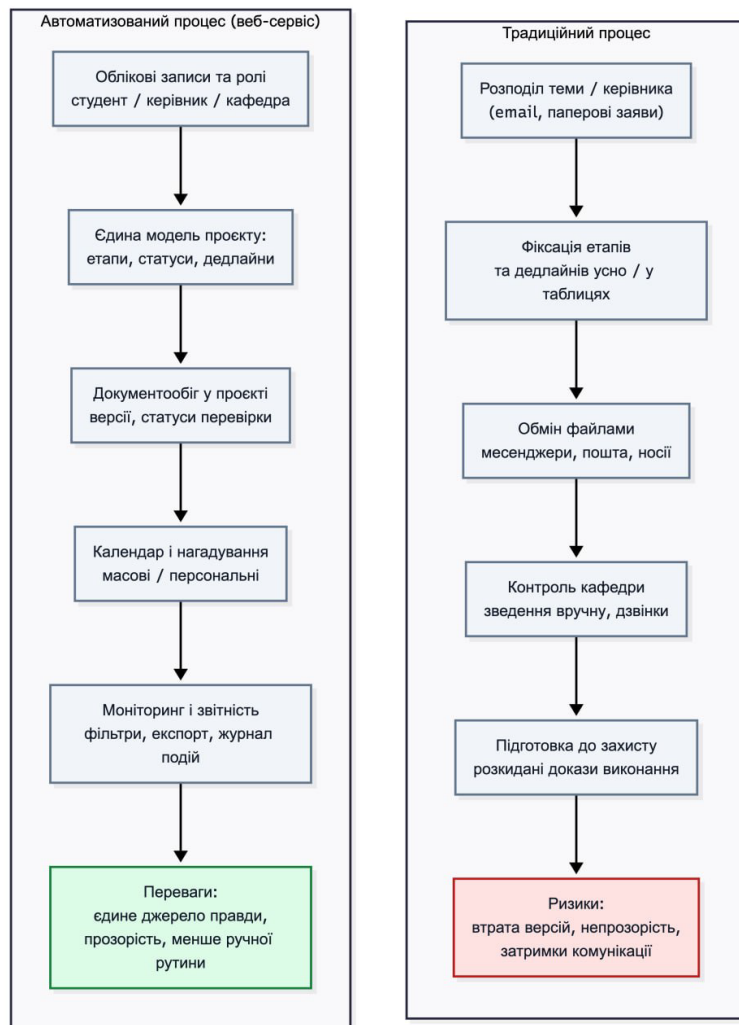


Рисунок 4.4 - Діаграма порівняння традиційного та автоматизованого процесу управління дипломними проєктами

Дані рисунка 4.4 підтверджують, що автоматизація забезпечує більш структуроване управління дипломними проєктами та покращує контроль основних процесів.

Автоматизація також підвищує контрольованість процесу. У системі фіксуються ролі користувачів, зв'язки між студентом і керівником, статуси документів, етапи виконання та календарні події. Це дозволяє швидше визначати поточний стан дипломної роботи та приймати організаційні рішення. Основні напрями автоматизації та результати їх впровадження для користувачів системи узагальнено в таблиці 4.6.

Таблиця 4.6 - Основні напрями автоматизації процесів та їхній практичний результат для користувачів системи

Напрямы автоматизації	Результат для користувачів
Централізація даних	Усі ключові відомості зберігаються в єдиній системі
Рольовий доступ	Користувач бачить лише релевантні функції та дані
Контроль дедлайнів	Важливі терміни відображаються в календарі та нагадуваннях
Документообіг	Файли мають статуси, версії та коментарі
Комунікація	Зауваження і повідомлення зберігаються в контексті проєкту
Адміністрування	Кафедра отримує інструменти контролю студентів, керівників і тем

Наведені в таблиці 4.6 результати демонструють, що використання системи спрощує доступ до інформації, допомагає своєчасно контролювати виконання робіт та робить процес роботи з документами більш зручним.

Короткий висновок: автоматизація процесу управління дипломними проєктами дозволяє зменшити залежність від розрізнених інструментів, підвищити прозорість контролю, централізувати документи й дедлайни та забезпечити структуровану взаємодію між студентами, керівниками й кафедрою.

4.4 Аналіз переваг розробленого веб-сервісу порівняно з традиційним підходом

Результати оцінювання ефективності автоматизації, наведені в підрозділі 4.3, підтверджують доцільність використання розробленого веб-сервісу для супроводу дипломного проектування. Проте оцінювання ефективності відображає насамперед вплив автоматизації на виконання окремих процесів, тоді як практична цінність системи визначається також зручністю її використання, рівнем керованості даними та можливістю організації взаємодії між учасниками освітнього процесу.

Для більш детального аналізу розробленого рішення виконано порівняння веб-сервісу з традиційним підходом за низкою ключових критеріїв. Результати такого порівняння наведено в таблиці 4.7.

Таблиця 4.7 - Порівняльний аналіз традиційного підходу та розробленого веб-сервісу за ключовими критеріями

Критерій порівняння	Традиційний підхід	Розроблений веб-сервіс
Зберігання даних	Інформація зберігається у кількох незалежних джерелах	Дані зберігаються централізовано в базі даних
Контроль тем	Погодження виконується через повідомлення або документи	Тема має окремий статус погодження
Актуальність інформації	Залежить від ручного оновлення	Дані оновлюються в межах веб-сервісу

Кінець таблиці 4.7

Критерій порівняння	Традиційний підхід	Розроблений веб-сервіс
Робота з документами	Документи передаються за допомогою сторонніх каналів зв'язку	Для кожного документа ведеться облік його стану та супровідної інформації
Контроль дедлайнів	Терміни контролюються вручну	Дедлайни та події відображаються в календарі
Комунікація	Повідомлення розміщені в різних каналах	Коментарі прив'язуються до проєкту, етапу або документа
Адміністративний контроль	Потребує ручної перевірки таблиць і списків	Адміністратор кафедри має окрему панель керування
Розмежування доступу	Часто не формалізоване	Доступ визначається роллю користувача

Порівняння показує, що застосування розробленої системи дає змогу краще організувати роботу над дипломними проєктами та спростити керування пов'язаними процесами. На відміну від традиційного підходу, де інформація та комунікація можуть бути розподілені між різними інструментами, у розробленій системі всі основні процеси об'єднано в єдиному інформаційному середовищі.

Однією з ключових можливостей системи є отримання повної інформації про поточний стан дипломної роботи в одному місці. Користувачі можуть отримувати актуальну інформацію про виконання етапів роботи, статуси

документів, заплановані події та результати перевірок без необхідності звернення до кількох джерел даних.

Також веб-сервіс створює зрозумілий механізм співпраці між студентами, керівниками та представниками кафедри. Використання ролей користувачів, механізмів погодження та засобів контролю виконання завдань сприяє підвищенню прозорості процесу та зменшенню ймовірності організаційних помилок. Узагальнення основних переваг веб-сервісу та очікуваних результатів від їх використання наведено в таблиці 4.8.

Таблиця 4.8 - Основні переваги розробленого веб-сервісу та очікувані практичні результати від їх впровадження

Перевага веб-сервісу	Практичний результат
Єдина база даних	Зменшується дублювання та втрата інформації
Статуси документів і тем	Спрощується контроль погодження та перевірки
Календар і дедлайни	Підвищується своєчасність виконання етапів
Коментарі в контексті проекту	Зберігається історія зауважень і рішень
Рольова модель доступу	Забезпечується контроль прав користувачів
Адміністративна панель	Полегшується робота кафедри з великою кількістю студентів

Як видно з таблиці 4.8, найбільш вагомими перевагами системи є підвищення впорядкованості роботи з інформацією, покращення контролю виконання дипломних проектів та спрощення взаємодії між усіма учасниками процесу. Це дозволяє не лише зменшити організаційне навантаження, а й створити більш зручні умови для супроводу дипломного проектування.

Короткий висновок: порівняно з традиційним підходом розроблений веб-сервіс забезпечує більш ефективну організацію роботи з дипломними проєктами, покращує контроль виконання завдань, спрощує взаємодію між учасниками процесу та створює єдине інформаційне середовище для супроводу дипломного проєктування.

4.5 Перспективи подальшого розвитку та вдосконалення системи

Розроблена система уже забезпечує виконання основних процесів супроводу дипломного проєктування. У системі реалізовано роботу з дипломними проєктами, документами, календарними подіями та адміністративними кабінетами користувачів. Крім того, підтримуються сповіщення в режимі реального часу, базові аналітичні можливості та журналювання подій. Наявний функціонал дозволяє ефективно організовувати взаємодію між користувачами.

Водночас подальший розвиток системи є важливим етапом підвищення її практичної цінності. Перспективи вдосконалення пов'язані не лише з розширенням функціональних можливостей, а й із підвищенням зручності використання, рівня автоматизації процесів та інтеграції веб-сервісу в інформаційну інфраструктуру закладу вищої освіти.

Основні напрями розвитку веб-сервісу з позиції користувачів, їх очікувана користь та пріоритетність наведено в таблиці 4.9. Технічні напрями вдосконалення, спрямовані на підвищення надійності, безпеки та масштабованості системи, подано в таблиці 4.10. Сукупність запропонованих заходів формує послідовний план подальшого розвитку веб-сервісу.

Таблиця 4.9 - Перспективні напрями розвитку веб-сервісу: пріоритет і очікуваний результат

№	Напрямок розвитку	Що дасть користувачу	Пріоритет
1	Розширені сповіщення (email, push)	Нагадування не лише в системі, а й на пошту або телефон	Високий
2	AI-помічник для студента та керівника	Підказки щодо етапів, структури роботи, чернетки коментарів	Середній
3	Інтеграція з Moodle / обліковою системою ЗВО	Менше ручного введення студентів і груп	Середній
4	Синхронізація з Google Calendar / Outlook	Дедлайни та консультації в звичному календарі	Середній
5	Мобільний застосунок	Зручний доступ з телефону, push-повідомлення	Середній
6	Гейміфікація (рівні, badges)	Вища мотивація студентів без спрощення вимог	Низький
7	Балансування навантаження керівників	Рівномірний розподіл дипломників між викладачами	Низький

Дані таблиці 4.9 свідчать, що найбільш перспективними напрямками розвитку є розширення механізмів сповіщення, інтеграція із зовнішніми сервісами та впровадження інтелектуальних функцій підтримки користувачів. Реалізація зазначених можливостей дозволить підвищити зручність роботи із системою та рівень автоматизації процесів.

Таблиця 4.10 - Технічні напрями розвитку: пріоритет і мета

№	Технічний напрям	Навіщо потрібен	Пріоритет
1	Єдине керування станом на клієнті (Redux Toolkit / RTK Query)	Швидший інтерфейс, менше “мигання” даних	Середній
2	Хмарне сховище файлів замість папки uploads/	Надійніше зберігання при зростанні навантаження	Високий
3	Резервне копіювання в хмару	Захист від втрати даних	Високий
4	Обов’язкова 2FA для адміністраторів	Більший захист облікових записів	Високий
5	Redis для Socket.IO та обмеження запитів	Стабільна робота при більшій кількості користувачів	Середній

Порівняння напрямів розвитку, наведених у таблицях 4.9 та 4.10, свідчить, що подальше вдосконалення системи доцільно здійснювати поетапно. У короткостроковій перспективі пріоритетними є заходи, спрямовані на підвищення безпеки та надійності роботи системи, а в подальшому - розвиток інтелектуальних функцій, інтеграція із зовнішніми сервісами та покращення користувацького досвіду.

Короткий висновок: перспективи розвитку веб-сервісу пов’язані з розширенням функціональних можливостей, підвищенням рівня безпеки та забезпеченням масштабованості системи. Реалізація запропонованих напрямів дозволить підвищити ефективність супроводу дипломного проєкування та розширити можливості практичного використання розробленого веб-сервісу.

ВИСНОВКИ ДО РОЗДІЛУ

У четвертому розділі було проведено оцінювання результатів роботи та аналіз ефективності розробленого веб-сервісу для управління дипломними проєктами. Було перевірено працездатність основних модулів системи, зокрема автентифікації, ролей, дипломних проєктів, документів, календарних подій, нагадувань та адміністративного керування.

Було встановлено, що інтерфейс системи відповідає основним вимогам зручності використання. Функціональність розподілено відповідно до ролей користувачів, а інформація подається у вигляді структурованих блоків, таблиць, календарів, статусів і повідомлень.

Оцінювання ефективності автоматизації показало, що розроблений веб-сервіс дозволяє замінити розрізнені інструменти єдиним середовищем для управління дипломними роботами. Це забезпечує централізоване збереження даних, контроль етапів, прозору роботу з документами, керування дедлайнами та збереження історії комунікації.

Порівняно з традиційним підходом розроблена система має переваги у вигляді рольового доступу, єдиної бази даних, адміністративної панелі, календаря подій, статусів перевірки та структурованого документообігу.

Отже, розроблений веб-сервіс виконує поставлені завдання та може бути використаний як основа для подальшого розвитку інформаційної системи супроводу дипломного проєктування у закладах вищої освіти.

					ІАЛЦ.467200.003 ПЗ	Арк.
						117
Зм.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

Під час опрацювання теми дипломного проєкту було створено веб-сервіс для управління дипломними проєктами. Розроблена система призначена для автоматизації процесів, пов'язаних із вибором і погодженням тем, супроводом етапів виконання дипломної роботи, роботою з документами, контролем дедлайнів, комунікацією між учасниками та адмініструванням даних на рівні кафедри.

У першому розділі було досліджено предметну область управління дипломними проєктами у закладах вищої освіти. Було визначено основних учасників процесу: студента, наукового керівника, адміністратора кафедри та адміністративні ролі вищого рівня. Також було проаналізовано проблеми традиційного супроводу дипломних робіт, серед яких розрізнене зберігання даних, ручний контроль строків, складність відстеження прогресу та відсутність єдиного середовища для документів і коментарів.

У другому розділі було обґрунтовано вибір архітектурного підходу та технологічного стеку. Для реалізації системи було обрано клієнт-серверну архітектуру, яка забезпечує розподіл відповідальності між інтерфейсом користувача, серверною логікою та базою даних. Клієнтську частину було реалізовано з використанням React, TypeScript і Vite. Серверну частину було побудовано на основі Node.js та Express.js. Для збереження даних було використано MongoDB у поєднанні з Mongoose.

У третьому розділі було описано проєктування та реалізацію веб-сервісу. Центральною сутністю системи визначено дипломний проєкт, до якого прив'язуються тема, опис, статус погодження, студент, керівник, етапи, підзавдання, документи, календарні події, нагадування та коментарі. Було реалізовано функціонал для освітніх користувачів системи. Для кожної ролі

					ІАЛЦ.467200.003 ПЗ	Арк.
						118
Зм.	Арк.	№ докум.	Підпис	Дата		

було передбачено окремі сценарії роботи, відповідні права доступу та інтерфейс.

У четвертому розділі було проведено оцінювання результатів роботи системи. Було перевірено працездатність основних модулів: автентифікації, керування ролями, дипломними проєктами, документами, календарними подіями, дедлайнами, нагадуваннями та адміністративним функціоналом. Було встановлено, що система забезпечує виконання основних вимог, сформованих на етапі аналізу предметної області.

Практичне значення роботи полягає в тому, що розроблений веб-сервіс дозволяє замінити розрізнені інструменти єдиним цифровим середовищем для управління дипломними проєктами. Система забезпечує централізоване збереження даних, контроль етапів виконання, роботу з документами, керування дедлайнами, збереження коментарів і підтримку адміністративного контролю.

Таким чином, мету дипломної роботи було досягнуто. Було розроблено веб-сервіс для управління дипломними проєктами, який відповідає поставленим функціональним і нефункціональним вимогам та може бути використаний як основа для подальшого розвитку інформаційної системи супроводу дипломного проєктування у закладах вищої освіти.

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		119

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sommerville I. Software Engineering. 10th ed. Boston: Pearson, 2016. 816 p.
2. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner's Approach. 9th ed. New York: McGraw-Hill Education, 2020. 672 p.
3. Richards M., Ford N. Fundamentals of Software Architecture: An Engineering Approach. Sebastopol: O'Reilly Media, 2020. 422 p.
4. Fowler M. Patterns of Enterprise Application Architecture. Boston: Addison-Wesley, 2002. 560 p.
5. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston: Addison-Wesley, 1994. 395 p.
6. Banks A., Porcello E. Learning React: Modern Patterns for Developing React Apps. 2nd ed. Sebastopol: O'Reilly Media, 2020. 310 p.
7. Haverbeke M. Eloquent JavaScript: A Modern Introduction to Programming. 3rd ed. San Francisco: No Starch Press, 2018. URL: <https://eloquentjavascript.net/> (Дата звернення: 20.04.2026).
8. React Documentation. URL: <https://react.dev/learn> (Дата звернення: 21.04.2026).
9. TypeScript Documentation. URL: <https://www.typescriptlang.org/docs/> (Дата звернення: 22.04.2026).
10. Vite Documentation. URL: <https://vite.dev/guide/> (Дата звернення: 23.04.2026).
11. Node.js Documentation. URL: <https://nodejs.org/en/docs> (Дата звернення: 24.04.2026).
12. Express.js Documentation. URL: <https://expressjs.com/> (Дата звернення: 25.04.2026).

					ІАЛЦ.467200.003 ПЗ	Арк.
						120
Зм.	Арк.	№ докум.	Підпис	Дата		

13. MongoDB Manual. URL: <https://www.mongodb.com/docs/manual/> (Дата звернення: 26.04.2026).
14. Mongoose Documentation. URL: <https://mongoosejs.com/docs/> (Дата звернення: 27.04.2026).
15. Socket.IO Documentation. URL: <https://socket.io/docs/v4/> (Дата звернення: 28.04.2026).
16. JSON Web Token Introduction. URL: <https://jwt.io/introduction> (Дата звернення: 29.04.2026).
17. Jones M., Bradley J., Sakimura N. JSON Web Token (JWT). RFC 7519. URL: <https://www.rfc-editor.org/rfc/rfc7519> (Дата звернення: 30.04.2026).
18. OWASP Top 10 Web Application Security Risks. URL: <https://owasp.org/www-project-top-ten/> (Дата звернення: 01.05.2026).
19. Helmet.js Documentation. URL: <https://helmetjs.github.io/> (Дата звернення: 02.05.2026).
20. MDN Web Docs. HTTP Overview. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview> (Дата звернення: 03.05.2026).
21. Axios Documentation. URL: <https://axios-http.com/docs/intro> (Дата звернення: 04.05.2026).
22. FullCalendar Documentation. URL: <https://fullcalendar.io/docs> (Дата звернення: 05.05.2026).
23. Tailwind CSS Documentation. URL: <https://tailwindcss.com/docs> (Дата звернення: 06.05.2026).
24. Chart.js Documentation. URL: <https://www.chartjs.org/docs/latest/> (Дата звернення: 07.05.2026).
25. Multer Documentation. URL: <https://github.com/expressjs/multer> (Дата звернення: 08.05.2026).
26. Nodemailer Documentation. URL: <https://nodemailer.com/about/> (Дата звернення: 09.05.2026).

					ІАЛЦ.467200.003 ПЗ	Арк.
						121
Зм.	Арк.	№ докум.	Підпис	Дата		

- 27.Nielsen J. Usability 101: Introduction to Usability. Nielsen Norman Group.
URL: <https://www.nngroup.com/articles/usability-101-introduction-to-usability/> (Дата звернення: 10.05.2026).
- 28.MongoDB University. Data Modeling Introduction. URL: <https://learn.mongodb.com/> (Дата звернення: 11.05.2026).
- 29.Git Documentation. URL: <https://git-scm.com/doc> (Дата звернення: 11.05.2026).
- 30.NGINX Documentation. URL: <https://nginx.org/en/docs/> (Дата звернення: 11.05.2026).
31. Atlassian. Trello Guide. URL: <https://trello.com/guide> (Дата звернення: 12.05.2026).
- 32.Docker Documentation. URL: <https://docs.docker.com/> (Дата звернення: 12.05.2026).

					ІАЛЦ.467200.003 ПЗ	Арк.
						122
Зм.	Арк.	№ докум.	Підпис	Дата		

ДОДАТОК А

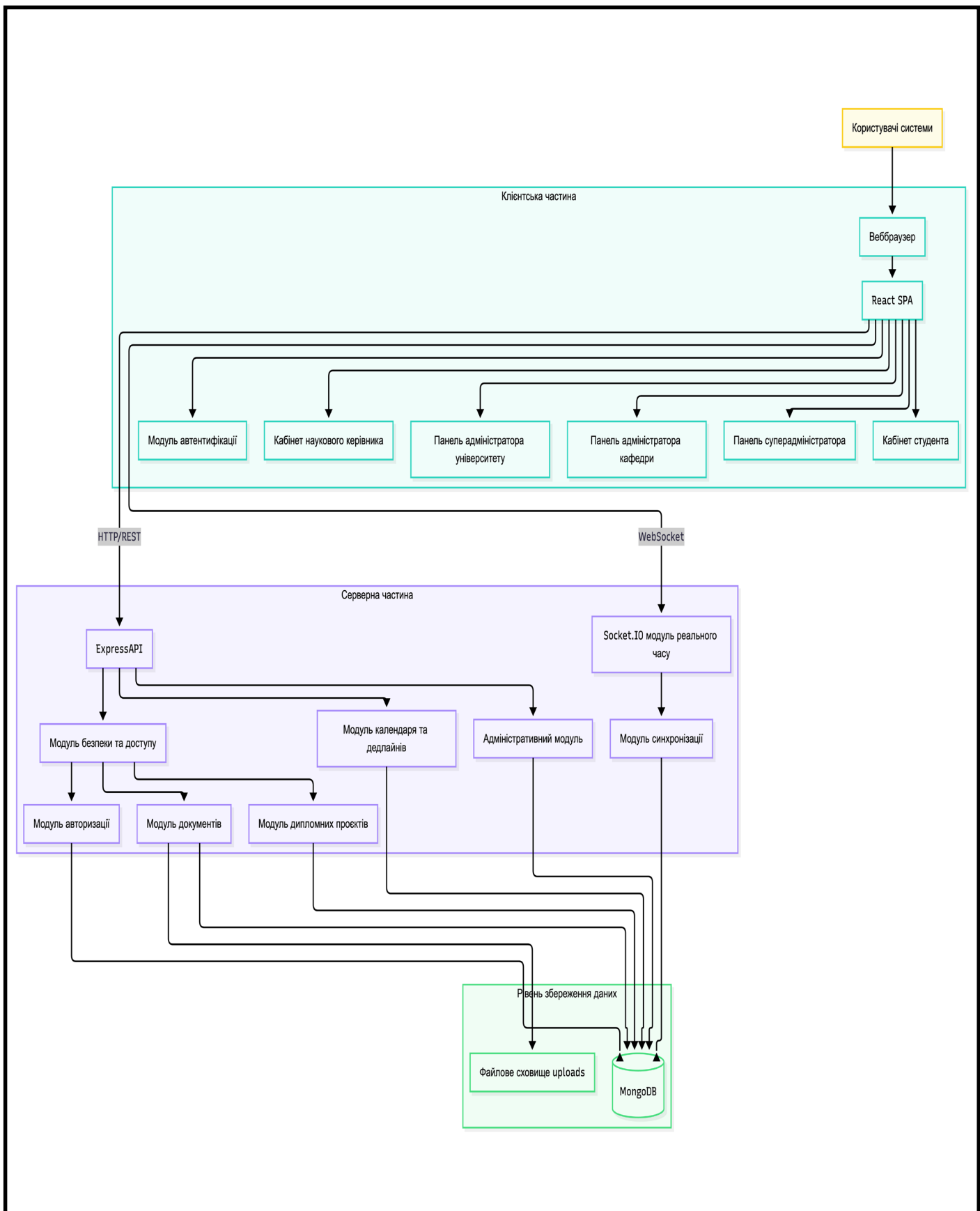
Веб-сервіс для управління дипломними проектами

Структурна схема системи

ІАЛЦ.467200.004 Д1

Аркушів 1

Київ 2026 р



					ІАЛЦ.467200.004 Д1			
					Веб-сервіс для управління дипломними проєктами Структурна схема системи			
Зм.	Арк.	№ докум.	Підпис	Дата				
Розробив		Хвиць М. С.						
Перевірив		Ковальчук О. М.						
Т. Контр.								
					КПІ ім. Ігоря Сікорського, ФІОТ, ІМ-21			
Н. Контр.		Гончаренко О. О.						
Затвердив		Волокита А. М.						
					Літ. Маса Масш.			
					Аркуш 1 Аркушів 1			

ДОДАТОК Б

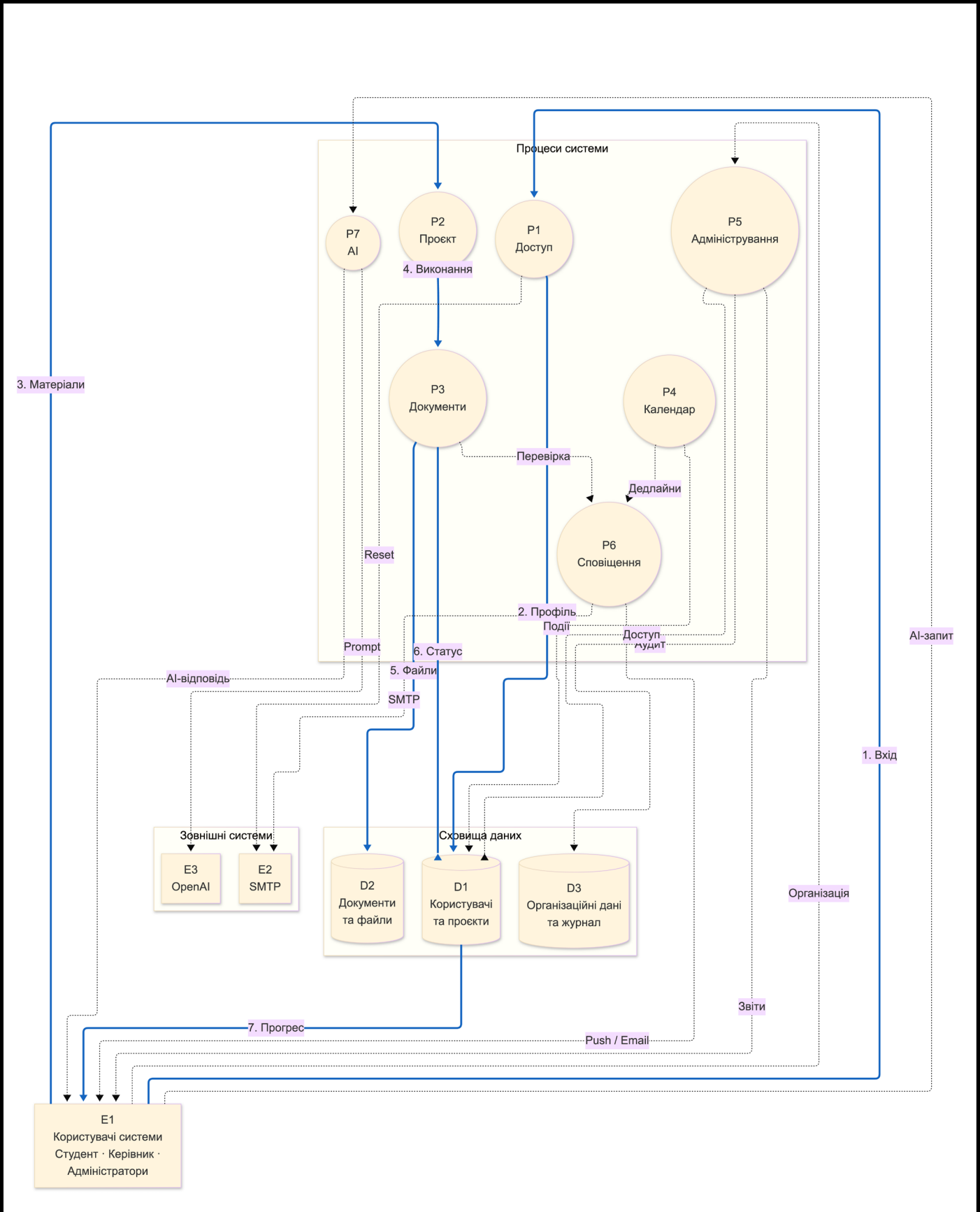
Веб-сервіс для управління дипломними проектами

Функціональна схема системи

ІАЛЦ.467200.005 Д2

Аркушів 1

Київ 2026 р



					ІАЛЦ.467200.005 Д2			
					Веб-сервіс для управління дипломними проектами Функціональна схема системи	Літ.	Маса	Масш.
Зм.	Арк.	№ докум.	Підпис	Дата				
Розробив		Хвиць М. С.						
Перевірив		Ковальчук О. М.						
Т. Контр.						Аркуш 1	Аркушів 1	
Н. Контр.		Гончаренко О. О.				КПІ ім. Ігоря Сікорського, ФІОТ, ІМ-21		
Затвердив		Волокита А. М.						

ДОДАТОК В

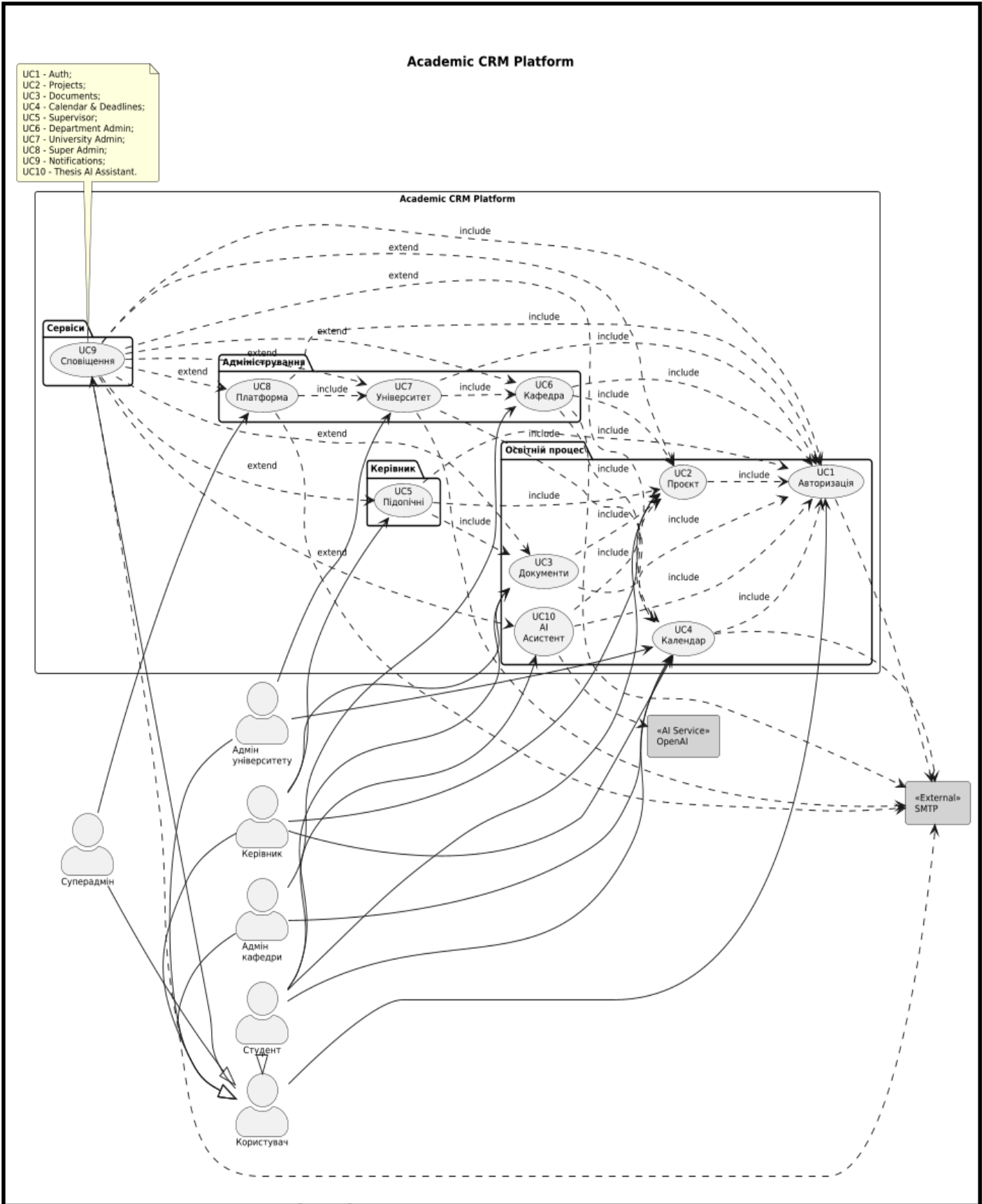
Веб-сервіс для управління дипломними проектами

Принципова схема системи

ІАЛЦ.467200.006 ДЗ

Аркушів 1

Київ 2026 р



					ІАЛЦ.467200.006 ДЗ		
					Веб-сервіс для управління дипломними проектами Принципова схема системи		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Хвиць М. С.			Літ.		
Перевірив		Ковальчук О. М.			Маса		
Т. Контр.					Масш.		
					Аркуш 1		Аркушів 1
Н. Контр.		Гончаренко О. О.			КПІ ім. Ігоря Сікорського, ФІОТ, ІМ-21		
Затвердив		Волокита А. М.					

ДОДАТОК Г

Веб-сервіс для управління дипломними проектами

Фрагменти програмного коду

ІАЛЦ.467200.007 Д4

Аркушів 14

Київ 2026 р

```
// Fragment from server/server.js
// Initializing the Express server and connecting API routes.
const app = express();
const server = http.createServer(app);
const io = new Server(server, { cors: { origin: true, credentials: true } });

// Basic security middleware and request body parsing
app.use(helmet());
app.use(cors({ origin: true, credentials: true }));
app.use(express.json({ limit: "2mb" }));
app.use(cookieParser());

// REST API
app.use("/api/auth", authRoutes);
app.use("/api/projects", projectRoutes);
app.use("/api/documents", documentRoutes);
app.use("/api/supervisor", supervisorRoutes);
app.use("/api/admin", adminRoutes);
app.use("/api/ai", aiRoutes);

export { app, server, io };

// Fragment from server/middleware/authMiddleware.js
// middleware protect: JWT validation and user upload.
export const protect = async (req, res, next) => {
  const header = req.headers.authorization;
  if (!header?.startsWith("Bearer")) {
    return res.status(401).json({
      success: false,
      message: "Немає авторизації, токен відсутній",
    });
  }

  try {
    const token = header.split(" ")[1];
    const decoded = jwt.verify(token, process.env.JWT_SECRET);

    if (decoded.typ && decoded.typ !== "access") {
      return res.status(401).json({
        success: false,
        message: "Невірний тип токена",
      });
    }

    req.user = await User.findById(decoded.id).select("-passwordHash");
    if (!req.user) {
      return res.status(401).json({
        success: false,
        message: "Користувача не знайдено",
      });
    }

    if (req.user.isActive === false) {
      return res.status(403).json({
        success: false,
        message: "Обліковий запис деактивовано",
      });
    }
  }
}
```

					ІАЛЦ.467200.007 Д4			
Зм.	Арк.	№ докум.	Підпис	Дата				
Розробив	Хвиць М. С.				Веб-сервіс для управління дипломними проектами Фрагменти програмного коду	Літ.	Аркуш	Аркушів
Перевірів	Ковальчук О. М.						1	14
Реценз.	Омельченко В. В.					КПІ ім. Ігоря Сікорського, ФІОТ, ІМ-21		
Н. Контр.	Гончаренко О. О.							
Затвердив	Волокита А. М.							

```

const tokenVersion = Number(decoded.tv || 0);
if (tokenVersion !== Number(req.user.authTokenVersion || 0)) {
  return res.status(401).json({
    success: false,
    message: "Сесію завершено. Увійдіть повторно",
  });
}

// Checking the active session in the DB (revoke / expiry)
if (decoded.jti) {
  const session = await UserSession.findOne({
    tokenId: String(decoded.jti),
    userId: req.user._id,
    revokedAt: null,
    expiresAt: { $gt: new Date() },
  }).select("_id");

  if (!session) {
    return res.status(401).json({
      success: false,
      message: "Сесію завершено або термін дії минув",
    });
  }
}

next();
} catch {
  return res.status(401).json({
    success: false,
    message: "Токен недійсний",
  });
}
};

// Fragment from server/controllers/auth/sessionController.js
// user login handler (email + password → JWT + session).
export const loginUser = async (req, res) => {
  try {
    const { email, password } = req.body;

    const user = await User.findOne({ email });
    if (!user) {
      return res.status(404).json({ message: "Користувача не знайдено" });
    }
    if (user.isActive === false) {
      return res.status(403).json({ message: "Обліковий запис деактивовано" });
    }

    const isMatch = await bcrypt.compare(password, user.passwordHash);
    if (!isMatch) {
      return res.status(400).json({ message: "Невірний пароль" });
    }

    // Create access + refresh pair and record session
    const tokenId = crypto.randomUUID();
    const refreshToken = signRefreshToken(user, tokenId);
    const accessToken = signAccessToken(user, tokenId);

    await UserSession.create({
      userId: user._id,
      tokenId,
      ipAddress: req.ip || "",
      userAgent: String(req.headers["user-agent"] || ""),
      expiresAt: new Date(Date.now() + 30 * 24 * 60 * 60 * 1000),
      lastSeenAt: new Date(),
    });
  }
};

```

					ІАЛЦ.467200.007 Д4	Арк.
						2
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    });

    const csrfToken = setAuthCookies(req, res, refreshToken);

    return res.json({
      success: true,
      token: accessToken,
      csrfToken,
      user: {
        id: user._id,
        fullName: user.fullName,
        email: user.email,
        role: normalizeRole(user.role),
        department: user.department || "",
        university: user.university || ""
      },
    });
  } catch (error) {
    console.error(error);
    return res.status(500).json({ message: "Помилка сервера" });
  }
};

// Fragment from server/models/Project.js
// diploma project model: stages, progress.
const stageSchema = new mongoose.Schema({
  title: { type: String, required: true },
  deadline: { type: Date, required: true },
  status: {
    type: String,
    enum: ["todo", "in-progress", "done", "overdue"],
    default: "todo",
  },
  progress: { type: Number, default: 0 },
  subtasks: [subtaskSchema],
});

const projSchema = new mongoose.Schema(
  {
    theme: { type: String, required: true },
    studentId: { type: mongoose.Schema.Types.ObjectId, ref: "User" },
    supervisorId: { type: mongoose.Schema.Types.ObjectId, ref: "User" },
    status: {
      type: String,
      enum: ["В роботі", "На перевірці", "Захищено", "Архів"],
      default: "В роботі",
    },
    overallProgress: { type: Number, default: 0 },
    stages: [stageSchema],
    reminders: [
      {
        id: String,
        text: String,
        createdAt: Date,
        authorRole: String,
      },
    ],
  },
  { timestamps: true },
);

```

					ІАЛЦ.467200.007 Д4	Арк.
						3
Зм.	Арк.	№ докум.	Підпис	Дата		

```

// Fragment from server/utils/rbac.js
// role-based access control (RBAC): rights based on roles.

export const PERMISSIONS = {
  PROJECT_DEPARTMENT_READ: "project:department:read",
  DOCUMENT_REVIEW: "document:review",
  SUPERVISOR_MANAGE: "supervisor:manage",
  SYNC_AUDIT_READ: "sync:audit:read",
};

// Matrix "role → allowed actions"
export const ROLE_PERMISSIONS = {
  SuperAdmin: Object.values(PERMISSIONS),
  UniversityAdmin: [
    PERMISSIONS.PROJECT_DEPARTMENT_READ,
    PERMISSIONS.DOCUMENT_REVIEW,
    PERMISSIONS.SYNC_AUDIT_READ,
  ],
  DepartmentAdmin: [
    PERMISSIONS.PROJECT_DEPARTMENT_READ,
    PERMISSIONS.DOCUMENT_REVIEW,
    PERMISSIONS.SYNC_AUDIT_READ,
  ],
  Supervisor: [
    PERMISSIONS.DOCUMENT_REVIEW,
    PERMISSIONS.SUPERVISOR_MANAGE,
  ],
  Student: [],
};

export const hasPermission = (role, permission) =>
  Boolean(ROLE_PERMISSIONS[normalizeRole(role)]?.includes(permission));

/** Middleware для маршруту: router.get("/", protect,
requirePermission(PERMISSIONS.X), handler) */
export const requirePermission = (permission) => (req, res, next) => {
  if (req.user && hasPermission(req.user.role, permission)) {
    return next();
  }
  return res.status(403).json({
    success: false,
    message: "Недостатньо прав доступу",
  });
};

// Fragment from server/controllers/project/projectWorkflowController.js
// Change subtask status and recalculate project progress.
export const toggleSubtask = async (req, res) => {
  try {
    const { projectId, stageId, subtaskId } = req.params;

    const project = await Project.findById(projectId)
      .populate("studentId", "fullName")
      .populate("supervisorId", "fullName");

    if (!project || !userCanAccessProject(req.user, project)) {
      return res.status(403).json({ message: "Немає доступу до проекту" });
    }

    // After defense, the student cannot change the assignment
    if (
      normalizeRole(req.user.role) === "Student" &&
      ["Захищено", "Апхів"].includes(String(project.status))
    ) {

```

					ІАЛЦ.467200.007 Д4	Арк.
						4
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        return res.status(409).json({
            code: "PROJECT_LOCKED",
            message: "Проект завершено. Редагування недоступне.",
        });
    }

    const stage = project.stages.id(stageId);
    const subtask = stage?.subtasks.id(subtaskId);
    if (!stage || !subtask) {
        return res.status(404).json({ message: "Етап або завдання не знайдено" });
    }

    // The student cannot change the assignment accepted by the supervisor
    if (
        normalizeRole(req.user.role) === "Student" &&
        subtask.reviewStatus === "accepted"
    ) {
        return res.status(409).json({
            code: "SUBTASK_LOCKED",
            message: "Завдання вже прийнято керівником.",
        });
    }

    subtask.completed = !subtask.completed;
    subtask.updatedAt = new Date();
    subtask.updatedById = req.user._id;

    // Stage progress = fraction of subtasks completed
    const total = stage.subtasks.length;
    const done = stage.subtasks.filter((s) => s.completed).length;
    stage.progress = total > 0 ? Math.round((done / total) * 100) : 0;
    stage.status =
        done === 0 ? "todo" : done === total ? "done" : "in-progress";

    // Overall project progress
    let allTasks = 0;
    let allDone = 0;
    for (const s of project.stages) {
        allTasks += s.subtasks.length;
        allDone += s.subtasks.filter((t) => t.completed).length;
    }
    project.overallProgress =
        allTasks > 0 ? Math.round((allDone / allTasks) * 100) : 0;

    recalculateProjectStatus(project);
    await project.save();
    await emitSystemSync(req, project);

    return res.json({
        success: true,
        overallProgress: project.overallProgress,
        stageProgress: stage.progress,
    });
} catch (error) {
    console.error(error);
    return res.status(500).json({ message: "Помилка сервера" });
}
};

// Fragment from server/controllers/project/myProjectController.js
// Notification delivery: saving to inbox + push via Socket.io.

/**
 * @param {object} project - Project document

```

					ІАЛЦ.467200.007 Д4	Арк.
						5
Зм.	Арк.	№ докум.	Підпис	Дата		

```

* @param {object} req - Express request (for io)
* @param {{ id, type, title, text, createdAt }} item - message
*/
export const notifyProjectStudentsInbox = async (project, req, item) => {
  const studentIds = collectStudentUserIdsOnProject(project);
  if (!studentIds.length || !item?.id) return;

  const room = projectSocketRoom(project._id);
  const io = req.app.get("io");

  for (const uid of studentIds) {
    const user = await User.findById(uid).select(
      "studentInbox studentNotificationPrefs role",
    );
    if (!user || normalizeRole(user.role) !== "Student") continue;
    if (user.studentNotificationPrefs?.newEvents === false) continue;

    const row = {
      id: String(item.id),
      type: item.type || "admin",
      title: String(item.title || ""),
      text: String(item.text || "").slice(0, 5000),
      createdAt: item.createdAt || new Date(),
      read: false,
    };

    const prev = Array.isArray(user.studentInbox) ? user.studentInbox : [];
    user.studentInbox = [row, ...prev].slice(0, 150);
    user.markModified("studentInbox");
    await user.save();

    // Instant UI update via WebSocket
    if (io && room) {
      io.to(room).emit("student_inbox_push", {
        userId: uid,
        item: row,
      });
    }
  }
};

// Fragment from server/server.js
// WebSocket rooms for real-time updates (project, manager, department).
export function registerSocketHandlers(io) {
  io.on("connection", (socket) => {
    const currentUser = socket.data.user;

    // Student/supervisor subscribes to updates on a specific project
    socket.on("join_project_room", async (projectId) => {
      if (!projectId) return;

      const project = await Project.findById(projectId).select(
        "studentId supervisorId teamMembers",
      );
      if (!project || !userCanAccessProject(currentUser, project)) return;

      const room = projectSocketRoom(projectId);
      if (room) socket.join(room);
    });

    // Manager - personal reminder room
    socket.on("join_supervisor_room", async (supervisorId) => {
      const userId = currentUser?._id?.toString();
      if (userId === supervisorId || isSuperadmin(currentUser?.role)) {

```

					ІАЛЦ.467200.007 Д4	Арк.
						6
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        socket.join(`supervisor_${supervisorId}`);
    }
});

// Department admin - department broadcast room
socket.on("join_department_room", (departmentName) => {
    if (!departmentName) return;
    const canJoin =
        isSuperadmin(currentUser?.role) ||
        (isDepartmentAdmin(currentUser?.role) &&
            currentUser.department?.trim() === String(departmentName).trim());
    if (canJoin) socket.join(`department_${departmentName}`);
});

// University Admin - Institution-level room
socket.on("join_university_room", (universityName) => {
    const canJoin =
        isSuperadmin(currentUser?.role) ||
        (isUniversityAdmin(currentUser?.role) &&
            currentUser.university === universityName);
    if (canJoin) socket.join(`university_${universityName}`);
});
});
}

// Fragment from client/src/services/api.ts
// HTTP client with automatic addition of JWT to requests.
export const api = axios.create({
    baseURL: "/api",
    withCredentials: true,
    headers: { "Content-Type": "application/json" },
});

// Each API request receives a Bearer token and a CSRF header
api.interceptors.request.use((config) => {
    const token = getAccessToken();
    if (token && config.headers) {
        config.headers.Authorization = `Bearer ${token}`;
    }
    const csrf = getCsrftoken() || hydrateCsrftokenFromCookie();
    if (csrf && config.headers) {
        config.headers["x-csrf-token"] = csrf;
    }
    return config;
});

// When 401 - attempt to update access token via refresh cookie
api.interceptors.response.use(
    (response) => response,
    async (error) => {
        const original = error.config;
        if (error.response?.status === 401 && !original._retry) {
            original._retry = true;
            try {
                const res = await api.post("/auth/refresh");
                if (res.data?.token) {
                    setAccessToken(String(res.data.token));
                    if (res.data.csrfToken) setCsrftoken(String(res.data.csrfToken));
                    return api(original);
                }
            } catch {
                clearAuthTokens();
            }
        }
    }
);

```

					ІАЛЦ.467200.007 Д4	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		7

```

        return Promise.reject(error);
    },
);

// Fragment from client/src/features/auth/Auth.tsx
// User login form (React + TypeScript).
type AuthProps = {
    setRole: (role: UserRole) => void;
    setIsLoggedIn: (v: boolean) => void;
    setUsername: (v: string) => void;
    setUserEmail: (v: string) => void;
};

export function LoginForm({
    setRole,
    setIsLoggedIn,
    setUsername,
    setUserEmail,
}: AuthProps) {
    const [email, setEmail] = useState("");
    const [password, setPassword] = useState("");
    const [error, setError] = useState("");
    const [isLoading, setIsLoading] = useState(false);

    const handleLogin = async (e: React.FormEvent) => {
        e.preventDefault();
        setError("");
        setIsLoading(true);

        try {
            const res = await api.post("/auth/login", { email, password });

            if (res.data.success) {
                setAccessToken(String(res.data.token || ""));
                setCsrfToken(String(res.data.csrfToken || ""));

                const userRole = mapBackendRoleToUiRole(res.data.user);
                setRole(userRole);
                setUsername(res.data.user.fullName || "");
                setUserEmail(res.data.user.email || "");
                setIsLoggedIn(true);
            }
        } catch (err: unknown) {
            const axiosErr = err as {
                response?: { status?: number; data?: { message?: string } };
            };
            if (axiosErr.response?.status === 429) {
                setError("Забагато спроб входу. Спробуйте пізніше.");
            } else {
                setError(
                    axiosErr.response?.data?.message ||
                    "Помилка авторизації. Перевірте email і пароль.",
                );
            }
        } finally {
            setIsLoading(false);
        }
    };

    return (
        <form onSubmit={handleLogin}>
            <input
                type="email"
                value={email}
            />
        </form>
    );
}

```

					ІАЛЦ.467200.007 Д4	Арк.
						8
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        onChange={(e) => setEmail(e.target.value)}
        placeholder="Email"
        required
      />
      <input
        type="password"
        value={password}
        onChange={(e) => setPassword(e.target.value)}
        placeholder="Пароль"
        required
      />
      {error && <p role="alert">{error}</p>}
      <button type="submit" disabled={isLoading}>
        {isLoading ? "Вхід..." : "Увійти"}
      </button>
    </form>
  );
}

// Fragment from server/controllers/documentController.js
// Uploading a document to a project (Multer + MongoDB + volume uploads).
export const uploadDocument = async (req, res) => {
  try {
    await fs.mkdir(path.join(process.cwd(), "uploads"), { recursive: true });

    const { projectId } = req.params;
    const { name, type, visibility, description } = req.body;
    const file = req.file; // Multer middleware saved the file to disk

    if (!file) {
      return res.status(400).json({
        success: false,
        message: "Файл не завантажено",
      });
    }

    const project = await Project.findById(projectId).populate(
      "supervisorId",
      "university department",
    );
    if (!project) {
      return res.status(404).json({ message: "Проект не знайдено" });
    }
    if (!userCanAccessProject(req.user, project)) {
      return res.status(403).json({
        success: false,
        message: "Немає доступу до цього проекту",
      });
    }

    const vis =
      visibility === "draft" || visibility === "supervisor"
        ? visibility
        : "supervisor";

    const newDocument = {
      name: name || file.originalname,
      type: type || "other",
      description: String(description || "").slice(0, 2000).trim(),
      url: `/uploads/${file.filename}`,
      size: file.size,
      uploadedByRole:
        normalizeRole(req.user?.role) === "Supervisor" ? "Supervisor" : "Student",
      uploadedById: req.user._id,
      status: "pending",
    };
  }
};

```

					ІАЛЦ.467200.007 Д4	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		9

```

        visibility: vis,
        uploadedAt: new Date(),
    };

    project.documents.push(newDocument);
    recalculateProjectStatus(project);
    await project.save();

    await emitSystemSync(req, project._id, {
        actionType: "document.uploaded",
        summary: `Завантажено: ${newDocument.name}`,
    });

    return res.status(201).json({
        success: true,
        document: newDocument,
    });
} catch (error) {
    console.error(error);
    return res.status(500).json({ message: "Помилка завантаження" });
}
};

// Fragment from server/controllers/documentController.js
// document review by the manager (status + comment)
export const reviewDocument = async (req, res) => {
    try {
        const { projectId, documentId } = req.params;
        const { status, commentText } = req.body;

        const project = await Project.findById(projectId).populate(
            "supervisorId",
            "university department",
        );
        if (!project) {
            return res.status(404).json({ message: "Проект не знайдено" });
        }
        if (!userCanReviewProjectDocuments(req.user, project)) {
            return res.status(403).json({
                success: false,
                message: "Немає прав на перевірку документа",
            });
        }

        const document = project.documents.id(documentId);
        if (!document) {
            return res.status(404).json({ message: "Документ не знайдено" });
        }

        if (status) {
            document.status = status; // pending | approved | rejected
        }

        if (commentText?.trim()) {
            document.comments.push({
                author: req.user.fullName || "Керівник",
                authorId: req.user._id,
                text: commentText.trim(),
                timestamp: new Date(),
            });
        }

        recalculateProjectStatus(project);
        await project.save();
    }
};

```

					ІАЛЦ.467200.007 Д4	Арк.
						10
Зм.	Арк.	№ докум.	Підпис	Дата		

```

// Notification to the student inbox
const studentId = project.studentId?.toString();
if (studentId) {
  const student = await User.findById(studentId).select("studentInbox");
  if (student) {
    const item = {
      id: `doc-review-${documentId}-${Date.now()}`,
      type: "feedback",
      title: "Керівник перевірів документ",
      text: `${document.name}: ${status || "коментар"}`,
      createdAt: new Date(),
      read: false,
    };
    student.studentInbox = [item, ...(student.studentInbox || [])].slice(0, 150);
    student.markModified("studentInbox");
    await student.save();

    const io = req.app.get("io");
    if (io) {
      io.to(projectSocketRoom(projectId)).emit("student_inbox_push", {
        userId: studentId,
        item,
      });
    }
  }
}

await emitSystemSync(req, project._id, {
  actionType: "document.reviewed",
  summary: `Перевірено: ${document.name}`,
});

return res.json({ success: true, document });
} catch (error) {
  console.error(error);
  return res.status(500).json({ message: "Помилка сервера" });
}
};

```

```

// Fragment from server/utils/projectAccess.js
// Checking access to the project by role and organizational scope.
// Student - presenter or co-author (teamMembers)
export const isStudentProjectMember = (project, userId) => {
  const uid = String(userId || "");
  if (!uid || !project) return false;
  if (project.studentId?.toString() === uid) return true;
  return (project.teamMembers || []).some(
    (m) => m.userId?.toString() === uid,
  );
};

const supervisorOrgFromProject = (project) => {
  const s = project?.supervisorId;
  if (!s || typeof s !== "object") {
    return { university: "", department: "" };
  }
  return {
    university: String(s.university || "").trim(),
    department: String(s.department || "").trim(),
  };
};

```

					ІАЛЦ.467200.007 Д4	Арк.
						11
Зм.	Арк.	№ докум.	Підпис	Дата		

```

// Admin sees the project only within their university / department
export const adminMatchesProjectScope = (user, project) => {
  if (!user || !project) return false;
  if (isSuperadmin(user.role)) return true;

  const { university: pu, department: pd } = supervisorOrgFromProject(project);

  if (isUniversityAdmin(user.role)) {
    const u = String(user.university || "").trim();
    return Boolean(u && pu && pu === u);
  }

  if (isDepartmentAdmin(user.role)) {
    const u = String(user.university || "").trim();
    const d = String(user.department || "").trim();
    return Boolean(u && d && pu && pd && pu === u && pd === d);
  }

  return false;
};

// Full access: team student · project manager · scope admin
export const userCanAccessProject = (user, project) => {
  if (!project || !user) return false;
  const uid = user._id?.toString();

  if (isStudentProjectMember(project, uid)) return true;

  const supervisorId =
    project.supervisorId?._id?.toString() ||
    project.supervisorId?.toString?.() ||
    "";

  if (normalizeRole(user.role) === "Supervisor" && supervisorId === uid) {
    return true;
  }

  return adminMatchesProjectScope(user, project);
};

// Document verification - project manager or admin only
export const userCanReviewProjectDocuments = (user, project) => {
  if (!project || !user) return false;
  const uid = user._id?.toString();
  const supervisorId =
    project.supervisorId?._id?.toString() ||
    project.supervisorId?.toString?.() ||
    "";

  if (normalizeRole(user.role) === "Supervisor" && supervisorId === uid) {
    return true;
  }

  return adminMatchesProjectScope(user, project);
};

// Snippet from server/utils/emitSystemSync.js and
server/controllers/syncController.js
// synchronization event log + real-time push to project participants.

// After changing the project - log entry + WebSocket event to clients.
export const emitSystemSync = async (req, projectLike, auditMeta = {}) => {
  const projectId = projectLike?._id || projectLike?.id || projectLike;
  if (!projectId || !req?.io) return;

```

					ІАЛЦ.467200.007 Д4	Арк.
						12
Зм.	Арк.	№ докум.	Підпис	Дата		

```

const id = projectId.toString();
const actionType = auditMeta.actionType || "project.change";
const summary = auditMeta.summary || "";

// Real-time: update UI without reloading the page
const room = projectSocketRoom(id);
if (room) {
  req.io.to(room).emit("project_sync_needed", {
    projectId: id,
    actionType,
    summary,
    timestamp: Date.now(),
  });
}

const project = await Project.findById(projectId).populate(
  "supervisorId",
  "department university",
);
if (!project) return;

const supervisorId = project.supervisorId?._id?.toString();
if (supervisorId) {
  req.io.to(`supervisor_${supervisorId}`).emit("supervisor_sync_needed", {
    projectId: id,
    actionType,
    summary,
  });
}

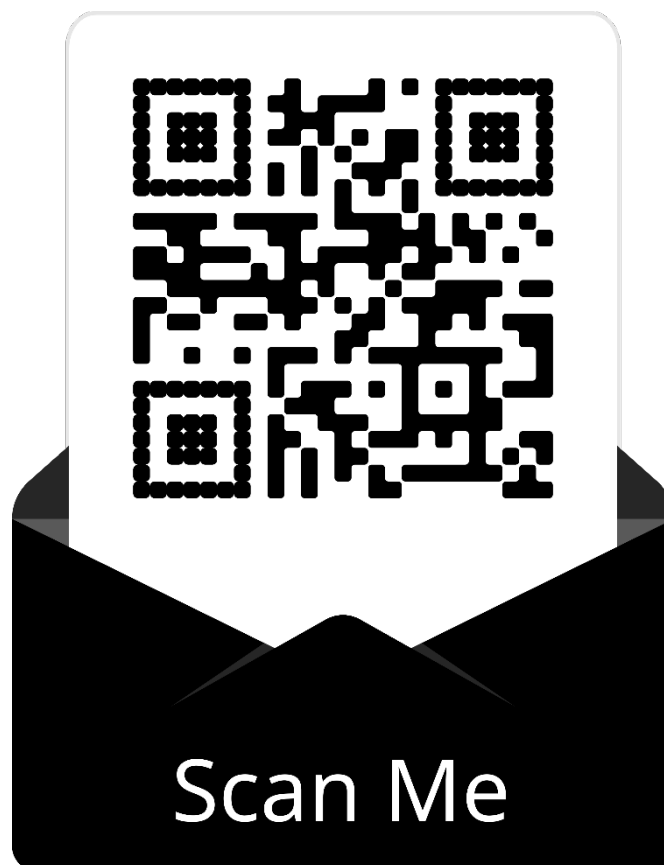
// Persistent log for the Change History tab
await SyncEvent.create({
  projectId,
  actorUserId: req.user?._id || null,
  actorRole: req.user?.role || "",
  actorName: req.user?.fullName || "",
  actionType,
  summary,
  department: project.supervisorId?.department || "",
  university: project.supervisorId?.university || "",
});
};

```

					ІАЛЦ.467200.007 Д4	Арк.
						13
Зм.	Арк.	№ докум.	Підпис	Дата		

Повний вихідний код розробленого програмного забезпечення
розміщено в репозиторії за посиланням:

https://github.com/makskhv21/Academic_CRM_Platform



					ІАЛЦ.467200.007 Д4	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		14