

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Інформаційна система обліку та моніторингу наукової діяльності аспірантів кафедри»

Виконав студент групи ПЗ-22-1
ЯСТРЕБОВА Анна Віталіївна

Керівник роботи:
ВАКАЛЮК Тетяна Анатоліївна

Рецензент
ФАНТ Микола Олександрович

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна Вакалюк

«27» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ЯСТРЕБОВА Анна Віталіївна**

Керівник роботи: **ВАКАЛЮК Тетяна Анатоліївна**

Тема роботи: **«Інформаційна система обліку та моніторингу наукової діяльності аспірантів кафедри»**,

затверджена Наказом закладу вищої освіти від **«27» лютого 2026 р., №92/с**

Вихідні дані для роботи: **розробка інформаційної системи PhD Science Tracker для цифровізації та автоматизації роботи з аспірантами в університеті**. Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Вакалюк Т.А.	27.02.2026р.	27.02.2026р.
2	Вакалюк Т.А.	24.04.2026р.	24.04.2026р.
3	Вакалюк Т.А.	02.05.2026р.	02.05.2026р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра присвячена створенню інформаційного середовища для автоматизації наукових процесів в університеті. Проєкт представлений програмним комплексом «PhD Science Tracker» та доданою до нього пояснювальною запискою обсягом 89 сторінок. Записка містить 32 графічних матеріалів та 28 таблиць, що детально розкривають архітектурні рішення.

Головною метою роботи є спрощення управлінської діяльності адміністрації закладу шляхом цифровізації супроводу здобувачів наукових ступенів. Система забезпечує комплексний облік показників аспірантів, сприяє ефективному контролю за виконанням індивідуальних планів, дозволяє оперативно формувати звіти та здійснювати прозорий моніторинг кожного етапу підготовки дисертації, починаючи з моменту обрання теми і закінчуючи фінальним захистом.

У ході проєкту було проведено детальне обґрунтування технічних вимог, виконано порівняльний аналіз існуючих рішень у галузі систем управління освітою та підібрано оптимальний технологічний стек, що гарантує високу продуктивність, масштабованість та надійність захисту даних користувачів.

КЛЮЧОВІ СЛОВА: ЦИФРОВІЗАЦІЯ ОСВІТИ, АСПІРАНТУРА, УПРАВЛІННЯ НАУКОВИМИ ПРОЦЕСАМИ, ІНФОРМАЦІЙНА СИСТЕМА, МОНІТОРИНГ НАУКОВОЇ ДІЯЛЬНОСТІ.

					ІПЗ.КР.Б-121-26-ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Інформаційна система обліку та моніторингу наукової діяльності аспірантів кафедри	Літ.	Арк.	Аркуші
Розроб.		Ястребова А.В.						
Керівник		Вакалюк Т.А.					3	87
Рецензент		Фант М.О.				Житомирська політехніка, група ІПЗ-22-1		
Зав.каф.		Вакалюк Т.А.						

ABSTRACT

The qualification work is dedicated to the creation of an information environment designed to automate scientific processes at the university. The project includes the "PhD Science Tracker" software complex and an accompanying explanatory note totaling 89 pages. The note contains 32 graphic materials and 28 tables that provide a detailed overview of the architectural solutions implemented.

The main objective of the work is to streamline the administrative management of the institution by digitizing the support process for research degree candidates. The system facilitates comprehensive tracking of doctoral students' progress, enables effective monitoring of individual study plans, allows for the efficient generation of reports, and ensures transparent oversight of every stage of dissertation preparation, starting from the selection of the research topic to the final defense.

During the project, a detailed justification of technical requirements was performed, a comparative analysis of existing education management systems was conducted, and an optimal technology stack was selected to ensure high performance, scalability, and robust user data protection.

KEYWORDS: DIGITALIZATION OF EDUCATION, POSTGRADUATE STUDIES, SCIENTIFIC PROCESS MANAGEMENT, INFORMATION SYSTEM, MONITORING OF SCIENTIFIC ACTIVITY

					ИПЗ.КР.Б-121-26-ПЗ	Арк.
						4

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ОБЛІКУ ТА МОНІТОРИНГУ НАУКОВОЇ ДІЯЛЬНОСТІ АСПІРАНТІВ КАФЕДРИ.....	9
1.1. Постановка задачі	9
1.2. Аналіз аналогів програмного продукту	12
1.3. Вибір архітектури вебплатформи	16
1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	20
Висновки до першого розділу	23
РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ОБЛІКУ ТА МОНІТОРИНГУ НАУКОВОЇ ДІЯЛЬНОСТІ АСПІРАНТІВ КАФЕДРИ.....	25
2.1. Визначення варіантів використання та структури системи	25
2.2. Розробка структури бази даних	33
2.3. Проєктування та реалізація алгоритмів роботи системи	49
2.4. Реалізація інформаційної системи	53
Висновки до другого розділу	57
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ	59
3.1. Порядок встановлення та налаштування параметрів системи	59
3.2. Структура інтерфейсу	62
3.3. Тестування роботи програмного комплексу	80
Висновки до третього розділу	83
ВИСНОВКИ	85
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	88
ДОДАТКИ	91

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API - Application Programming Interface (прикладний програмний інтерфейс)

CQRS - Command Query Responsibility Segregation (розподіл відповідальності між командами та запитам)

DDD - Domain-Driven Design (предметно-орієнтоване проектування)

DTO - Data Transfer Object (об'єкт передавання даних)

HTTP - HyperText Transfer Protocol (протокол передавання гіпертексту)

JWT - JSON Web Token (веб-токен у форматі JSON)

ORM - Object-Relational Mapping (об'єктно-реляційне відображення)

RBAC - Role-Based Access Control (рольова модель управління доступом)

REST - Representational State Transfer (передавання представницького стану)

SPA - Single Page Application (односторінковий застосунок)

UUID - Universally Unique Identifier (універсальний унікальний ідентифікатор)

LMS - Learning Management System (система управління навчанням)

CRIS - Current Research Information System (система управління науковою інформацією)

ВСТУП

Актуальність теми. Зумовлена процесами цифрової трансформації вищої освіти в Україні та необхідністю адаптації механізмів підготовки наукових кадрів до сучасних вимог. Згідно з новими стандартами підготовки докторів філософії, важливою складовою ефективності аспірантів є виконання індивідуального плану. Індивідуальний план включає публікаційну активність, апробацію результатів та педагогічну практику. Традиційні паперові методи або використання різних файлових сховищ стають неефективними при такому стрімкому зростанні обсягів звітної документації та посилених вимог до академічної доброчесності.

Спеціалізована система ведення звітності дозволить автоматизувати процеси збору та обробки даних, забезпечити контроль виконання індивідуальних планів та створити єдиний цифровий простір для накопичення та зберігання наукових здобутків кафедри. Така система сприяє оптимізації документообігу, мінімізації помилок при атестації та підвищенню якості управління науковою діяльністю.

Метою кваліфікаційної роботи є проектування та розробка веб-орієнтованої інформаційної системи для автоматизації процесів звітування та контролю успішності аспірантів кафедри.

Встановлена мета обумовлює наступні **завдання**:

- Проведення аналізу предметної області та наявного програмного забезпечення для управління навчальним процесом;
- Визначення архітектури, рольової моделі та узагальненої структури платформи;
- Обґрунтування та вибір сучасних засобів і технологій реалізації системи;
- Програмна реалізація вебдодатка «PhD Science Tracker».

Об'єктом дослідження в роботі виступає процес інформаційного супроводу та моніторингу виконання індивідуальних планів наукової роботи аспірантів.

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						7

Предметом дослідження є методи та програмні засоби створення веб-орієнтованих систем для автоматизації звітності у закладах вищої освіти. Методологічну основу розробки складають методи системного аналізу, об'єктно-орієнтованого проектування, моделювання реляційних баз даних, а також сучасні підходи до побудови клієнт-серверних архітектур.

Методи дослідження. У роботі використано аналіз предметної області та існуючих аналогів систем обліку наукової діяльності, метод варіантів використання для формалізації вимог, об'єктно-орієнтоване проектування і доменно-орієнтоване моделювання, метод скінченних автоматів для проектування логіки переходів станів.

Практичне значення. Розроблена система може бути впроваджена на кафедрах університету для автоматизації обліку наукової діяльності аспірантів, що замінює ручне ведення таблиць Excel і паперових звітів.

Апробація роботи. За напрямом кваліфікаційної роботи здійснено публікацію двох тез доповідей, які увійшли до матеріалів науково-практичних конференцій:

1. Ястребова А.В. Необхідність розробки вебсистеми з оптимізації управління науковою роботою здобувачів ступеня доктор філософії. Тези доповідей XVI Міжнародної науково-технічній конференції "Інформаційно-комп'ютерні технології". Житомир, 2026.

2. Ястребова А.В., Вакалюк Т.А. Використання патерна Specification для верифікації бізнес-обмежень у складних інформаційних системах. Тези доповідей XVI Міжнародної наукової конференції «Сучасні інформаційні технології – 2026». Одеса, 2026. С. 274

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ОБЛІКУ ТА МОНІТОРИНГУ НАУКОВОЇ ДІЯЛЬНОСТІ АСПІРАНТІВ КАФЕДРИ

1.1. Постановка задачі

Підготовка наукових кадрів вищої кваліфікації в Україні останніми роками зазнала помітної трансформації, причому вимоги до здобувачів ступеня доктора філософії суттєво розширилися внаслідок упровадження положень профільного законодавства про вищу освіту та стандартів підготовки докторів філософії [1]. Чинні нормативні документи передбачають, що освітньо-наукова програма не обмежується написанням дисертаційного тексту, а охоплює також публікаційну активність у виданнях, які індексуються міжнародними наукометричними платформами Scopus та Web of Science, апробацію отриманих результатів на конференціях, проходження педагогічної практики та складання кваліфікаційних іспитів [2].

Кафедри університетів, на яких ведеться підготовка аспірантів, змушені адмініструвати значний масив інформації щодо кожного здобувача ступеня доктора філософії. Зокрема, необхідно відстежувати поточний стан виконання індивідуального плану кожного аспіранта, контролювати своєчасність подання семестрових звітів та проходження семестрових атестацій, вести облік опублікованих наукових праць і участі у конференціях, фіксувати поетапний прогрес написання дисертаційного тексту, а також документувати всі адміністративні рішення, пов'язані зі змінами в академічному статусі здобувачів.

Практика показує, що на переважній більшості кафедр усі ці завдання вирішуються за допомогою неспеціалізованих засобів, а саме паперових журналів, таблиць Microsoft Excel, що надсилаються електронною поштою, або зберігаються у хмарних файлових сховищах, а також переписки в месенджерах, де важливі документи легко губляться серед побутових повідомлень. Такий підхід призводить до фрагментарності даних і ускладнює

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						9

формування зведеної картини щодо наукових здобутків усіх аспірантів кафедри, а особливо у передатестаційний та звітний періоди.

Виявлена проблема зумовлює необхідність створення спеціалізованої інформаційної системи, яка б консолідувала в єдиному цифровому середовищі весь документообіг і комунікації, пов'язані з підготовкою аспірантів. Для чіткого формулювання завдання розробки необхідно визначити коло учасників системи та перелік функцій, які вона має виконувати.

Система обліку та моніторингу наукової діяльності аспірантів кафедри передбачає чотири категорії користувачів із різним рівнем доступу та набором дозволених операцій.

Аспірант є основним джерелом даних у системі. Він самостійно веде своє наукове портфоліо, реєструючи опубліковані наукові статті та матеріали конференцій із зазначенням типу видання, наукометричних баз індексування та посилань; подає заявки на участь у наукових заходах; заповнює та подає на розгляд керівника семестрові звіти; відстежує власний трекер завдань та прогрес виконання дисертаційної роботи. Крім того, аспірант має доступ до свого електронного індивідуального плану, де він може бачити поточний стан виконання кожного пункту.

Науковий керівник здійснює кураторство над закріпленою за ним аспірантом. Його функціональні повноваження включають перегляд матеріалів, поданих аспірантами, рецензування та попереднє затвердження семестрових звітів із можливістю залишати структуровані зауваження, внесення коректив до індивідуальних планів здобувачів, моніторинг публікаційної та конференційної активності, а також перегляд зведеної аналітики по його аспірантах з ключовими показниками результативності.

Завідувач кафедри здійснює управлінський нагляд за підготовкою всіх аспірантів кафедри. До його компетенції належить фінальне офіційне затвердження семестрових звітів з прив'язкою до реквізитів протоколу засідання кафедри, формування та реєстрація наказів щодо академічних переміщень здобувачів (зарахування, переведення, відрахування, академічні

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						10

відпустки), управління даними щодо дисертаційних досліджень, включаючи результати перевірки на плагіат і допуск до захисту, а також доступ до консолідованої аналітики по підрозділу.

Адміністратор системи забезпечує технічне та структурне налаштування платформи. Зокрема, він управляє довідниками кафедр, спеціальностей і навчальних програм, реєструє нових користувачів та призначає їм ролі, має доступ до журналу аудиту дій, здійснює нагляд за цілісністю даних та технічний моніторинг системи.

На підставі аналізу потреб визначених груп користувачів сформовано такий перелік функціональних вимог до проєктованої інформаційної системи.

Система має забезпечувати захищену автентифікацію користувачів на основі облікових записів електронної пошти з розмежуванням прав доступу відповідно до призначеної ролі. При цьому кожен користувач повинен мати доступ виключно до тих даних і функцій, що відповідають його ролі та кафедральній приналежності.

Підсистема управління індивідуальними планами має підтримувати електронне формування структурованого плану кожного аспіранта із розбивкою на семестри та типи активностей, відображення поточного статусу виконання кожного пункту плану, а також збереження повної хронології змін.

Модуль обліку наукових здобутків повинен забезпечувати реєстрацію публікацій із розширеними метаданими (тип видання, наукометричні бази, DOI, рік, співавтори), облік участі у наукових конференціях та семінарах із зазначенням ролі учасника, рівня заходу і дедлайнів, а також відстеження навчальних дисциплін з кредитами ECTS і фінальними оцінками.

Підсистема документообігу звітності повинна реалізовувати повний цикл погодження семестрових звітів.

Модуль управління дисертаційними дослідженнями має відстежувати поетапний прогрес підготовки дисертації, фіксувати результати перевірки на академічну доброчесність, зберігати дані щодо передзахисту та захисту.

Аналітичний модуль повинен формувати для керівників та завідувача

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						11

кафедри зведені інформаційні панелі з ключовими показниками, враховуючи кількість публікацій у наукометричних базах, відсоток виконання індивідуальних планів, статистику конференційної активності. Для адміністративного персоналу необхідна консолідована аналітика по всій установі.

До нефункціональних вимог проєктованої системи належать такі. Безпека даних повинна забезпечуватись через механізми аутентифікації на основі JSON Web Token, шифрування паролів криптографічними хеш-функціями та автоматичне ведення журналу аудиту всіх операцій зі зміни даних. Система повинна мати адаптивний інтерфейс, придатний для роботи як на стаціонарних робочих станціях, так і на мобільних пристроях. Архітектура системи повинна забезпечувати можливість масштабування при збільшенні кількості кафедр і аспірантів

1.2. Аналіз аналогів програмного продукту

Перед тим, як перейти до проєктування власного програмного продукту, необхідно провести системний огляд наявних на ринку рішень, що так чи інакше можуть застосовуватись для автоматизації процесів підготовки та моніторингу наукової діяльності аспірантів. Це дозволить не лише виявити недоліки існуючих підходів, але й запозичити їхні найкращі практики при проєктуванні власної системи. Для аналізу обрано чотири категорії програмного забезпечення, що найчастіше залучаються з подібною метою, а саме системи управління навчанням, універсальні системи управління проєктами, хмарні офісні пакети та спеціалізовані платформи для обліку наукових досліджень.

Найбільш поширеною системою управління навчанням у вищих навчальних закладах України та світу є Moodle, яка розроблена за моделлю вільного програмного забезпечення з відкритим кодом [3]. Платформа має відкриту ліцензію для академічного використання і надає широкий набір функцій для організації дистанційного навчання. Цей інструментарій охоплює модулі курсів, автоматизоване тестування, системи оцінювання, журнали

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						12

обліку успішності та засоби комунікації (рис. 1.1).

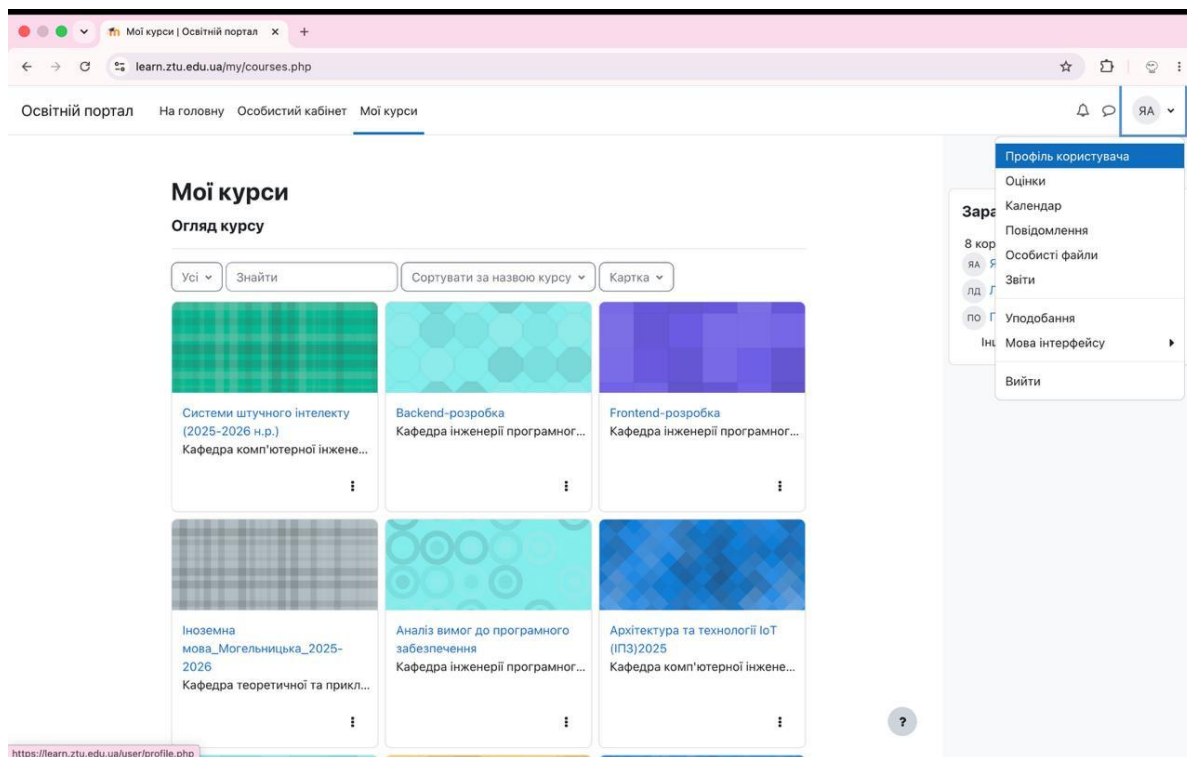


Рис. 1.1. Інтерфейс системи дистанційного навчання Moodle

Система Moodle є відкритою платформою з розвиненою спільнотою розробників та великою кількістю додаткових модулів і плагінів. Вона успішно справляється із завданнями організації масового дистанційного навчання для великих груп студентів. Проте архітектура цієї системи фундаментально побудована на концепції навчального курсу як центральної сутності, де кожен здобувач освіти проходить однаковий набір завдань і оцінюється за уніфікованими критеріями. Така модель є принципово непридатною для супроводу аспірантської підготовки, оскільки на цьому рівні кожен здобувач реалізує унікальний дослідницький проєкт, а взаємодія з науковим керівником має глибоко індивідуалізований характер. Відповідно жорстке прив'язування до навчальних дисциплін, відсутність розширених наукових профілів здобувачів і неможливість ведення довготривалих індивідуальних планів роблять пряме застосування Moodle для управління підготовкою докторів філософії безперспективним.

Іншою поширеною практикою серед академічної спільноти є спроба адаптувати універсальні системи управління проєктами на зразок популярної

платформи Notion для відстеження наукового прогресу здобувачів [4] . Цей інструмент вирізняється сучасним візуальним інтерфейсом, високою гнучкістю індивідуального налаштування і підтримкою інтерактивних дошок канбан, що робить його дуже привабливим для особистого планування щоденної наукової роботи (рис. 1.2). Ця платформа функціонує як універсальний робочий простір і дозволяє користувачам самостійно будувати довільні реляційні бази даних, вести розгалужені текстові нотатки, планувати поточні завдання та здійснювати спільне розгалужене редагування документів.

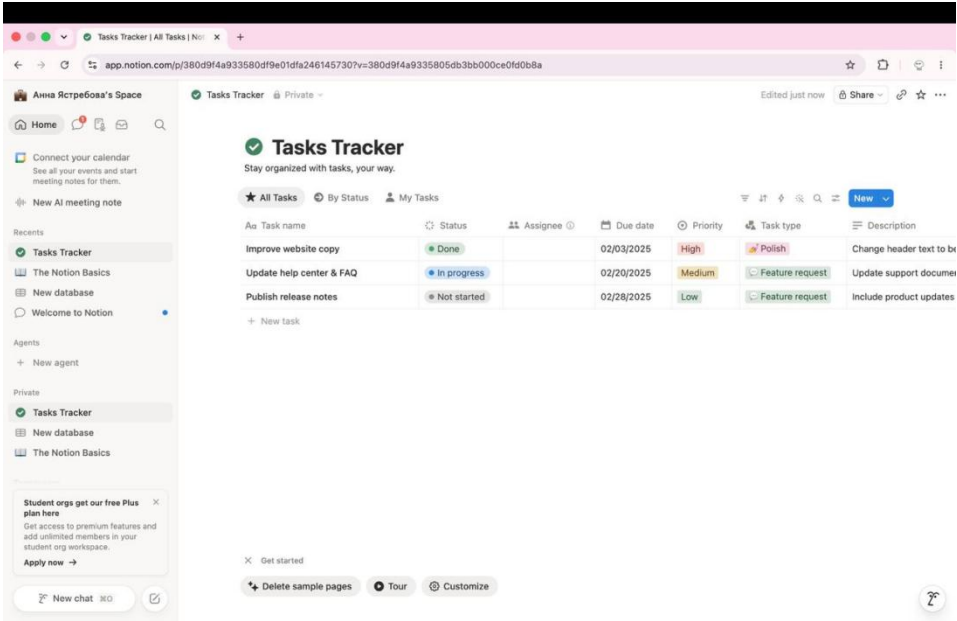
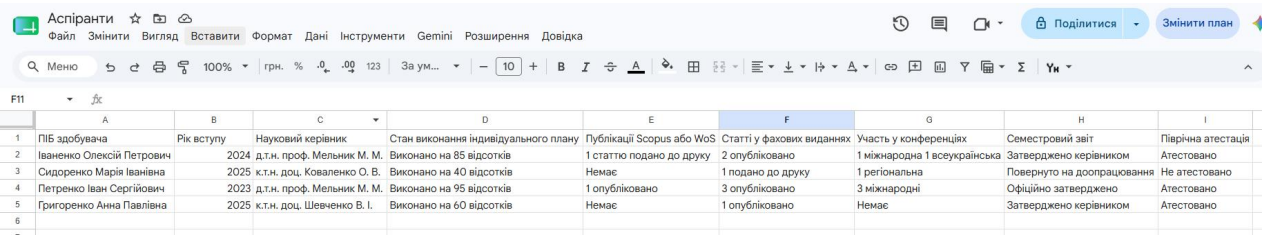


Рис. 1.2. Приклад використання Notion

Завдяки такій інструментальній свободі деякі аспіранти використовують Notion як персональний цифровий науковий щоденник для фіксації проміжних результатів досліджень. Однак варто враховувати, що Notion є суто персональним або командним корпоративним інструментом і не має жодної вбудованої академічної рольової моделі для управління великою групою дослідників у межах випускової кафедри. Спеціалізованого функціоналу для послідовного погодження чи офіційного затвердження документів у цій системі не передбачено, а для налаштування навіть елементарних сповіщень про терміни здачі звітності потрібні складні сторонні інтеграції через зовнішні сервіси.

Найбільш масово поширеним підходом у закладах вищої освіти

залишається використання хмарних офісних пакетів, до яких передусім відносять сервіси Google Workspace або Microsoft 365, для ведення академічної звітності здобувачів [5]. За такої організації завідувач кафедри або відповідальний методист створює електронну таблицю Excel чи таблицю Google зі списком аспірантів і окремими стовпцями для кожного пункту індивідуального плану (рис. 1.3).



	A	B	C	D	E	F	G	H	I
1	ПІБ здобувача	Рік вступу	Науковий керівник	Стан виконання індивідуального плану	Публікації Scopus або WoS	Статті у фахових виданнях	Участь у конференціях	Семестровий звіт	Піврічна атестація
2	Іванченко Олександр Петрович	2024	д.т.н. проф. Мельник М. М.	Виконано на 85 відсотків	1 стаття подано до друку	2 опубліковано	1 міжнародна 1 всеукраїнська	Затверджено керівником	Атестовано
3	Сидоренко Марія Іванівна	2025	д.т.н. доц. Коваленко О. В.	Виконано на 40 відсотків	Немає	1 подано до друку	1 регіональна	Повернуто на доопрацювання	Не атестовано
4	Петренко Іван Сергійович	2023	д.т.н. проф. Мельник М. М.	Виконано на 95 відсотків	1 опубліковано	3 опубліковано	3 міжнародні	Офіційно затверджено	Атестовано
5	Григоренко Анна Павлівна	2025	д.т.н. доц. Шевченко В. І.	Виконано на 60 відсотків	Немає	1 опубліковано	Немає	Затверджено керівником	Атестовано
6									

Рис. 1.3. Фрагмент зведеної електронної таблиці для контролю наукового прогресу аспірантів

Головним недоліком такого підходу є повна відсутність повноцінної реляційної бази даних та належної автоматизованої системної логіки валідації інформації. Зазначені таблиці з часом неминуче перетворюються на важко читабельний масив даних без чіткої структури, причому дублювання інформації стає регулярним явищем, а поточна версійність файлів після внесення численних редагувань фактично не піддається контролю. У разі значного збільшення кількості аспірантів або при зміні відповідального працівника кафедри така система стрімко деградує до стану, коли оперативно відновити актуальний перебіг подій стає вкрай складно. Крім того жодних автоматизованих нотифікацій про наближення дедлайнів чи алгоритмічних перевірок коректності заповнення даних ці базові офісні пакети не забезпечують, що зумовлює потребу у постійному ручному моніторингу з боку адміністрації університету.

Результати аналізу узагальнено у порівняльній таблиці (табл. 1.1), де кожна платформа оцінена за критеріями відповідності вимогам, сформульованим у підрозділі.

Порівняльний аналіз аналогів програмного продукту

Назва функції	LMS (Moodle)	Трекери (Notion)	Хмарні офісні пакети
Індивідуальний план аспіранта як базова сутність	Ні	Частково	Ні
Рольова модель (аспірант / керівник / завідувач)	Частково	Ні	Ні
Погодження та затвердження звітів	Ні	Ні	Ні
Облік публікацій (Scopus/WoS)	Ні	Ні	Частково
Облік конференцій та апробацій	Ні	Частково	Частково
Трекінг дисертаційного прогресу	Ні	Частково	Ні
Кадрові накази та протоколи	Ні	Ні	Ні
Аналітична панель для завідувача	Обмежено	Обмежено	Ні
Відкритість / вартість для кафедри	Безкоштовно	Freemium	Freemium

Як випливає з проведеного аналізу, жодна з розглянутих платформ не задовольняє комплексу вимог, визначених у попередньому підрозділі. LMS-системи ефективні для масового навчання, але не розраховані на індивідуалізований дослідницький процес. Трекери завдань та хмарні офісні пакети забезпечують певну гнучкість, але позбавлені системної логіки, рольового розмежування і засобів академічного документообігу. CRIS-системи орієнтовані на аналітику публікаційної активності університету в цілому, але не вирішують завдань управління підготовкою аспірантів на рівні кафедри. Це обґрунтовує доцільність розробки спеціалізованої веб-орієнтованої системи, що враховуватиме специфічний контекст українського академічного середовища та вимоги чинного законодавства

1.3. Вибір архітектури вебплатформи

Під час проєктування системи для управління аспірантською підготовкою на рівні кафедри варто розглянути три основні архітектурні підходи, серед яких класична монолітна архітектура, мікросервісна архітектура та проміжний варіант у вигляді модульного моноліту [6]. Кожен із

них має власні сильні та слабкі сторони, які потрібно зважити в контексті конкретних вимог проєкту, очікуваного масштабу й наявних ресурсів на розробку та подальшу підтримку, аби обрати рішення, оптимальне водночас і для розробника, і для майбутніх користувачів.

Класична монолітна архітектура передбачає, що вся серверна частина застосунку розробляється як єдиний розгортуваний артефакт, у якому компоненти тісно поєднані й виконуються в межах одного процесу [7]. Головною перевагою такого підходу є простота розробки, тестування та розгортання, адже весь код зосереджено в одній базі, що полегшує налагодження й трасування запиту від користувача через усі шари до бази даних. Розробникові не доводиться розбиратися зі складною топологією мережевої взаємодії між сервісами, оскільки всі залежності перебувають локально, а саме розгортання зводиться до копіювання одного виконуваного артефакту на сервер.

Водночас монолітна архітектура має відомі недоліки при масштабуванні систем та залученні великих команд розробників. Кожне оновлення якої-небудь окремої функції вимагає перерозгортання всього застосунку, що збільшує ризик неконтрольованих впливів однієї частини коду на іншу та ризик введення регресій. При значному зростанні розміру кодової бази моноліт може стати складним для розуміння та підтримки, особливо якщо розроблено кількома незалежними командами без чіткої домену відповідальності. Однак для невеликих проєктів в межах однієї кафедри з обмеженими ресурсами на розробку такі недоліки не є критичними, оскільки система розроблюється однією людиною [8]. До того ж, універсальність монолітної архітектури означає, що легко можна додати будь-яку нову функцію без необхідності проектування нових сервісів.

Мікросервісна архітектура передбачає декомпозицію системи на набір незалежних сервісів, кожен з яких відповідає за певну функцію програми, розгортається окремо і комунікує з іншими через мережевий протокол або брокер повідомлень [9]. Мікросервіси забезпечують незалежне

масштабування окремих компонентів, коли можна запустити більше екземплярів популярного сервісу без скалювання всіх інших, технологічну гетерогенність, коли різні сервіси можуть бути написані різними мовами програмування, та ізоляцію відмов, коли збій одного сервісу не обов'язково призводить до збою всієї системи. Однак їхнє впровадження суттєво підвищує складність інфраструктури через необхідність налаштування шлюзів прикладного програмного інтерфейсу, сервісів виявлення та навантажувального балансування, розподіленого трасування запитів та складнішої оркестрації розгортань. Крім того, розроблювачі повинні мати глибоке розуміння нюансів розподілених систем та потенційних проблем, які можуть виникнути при комунікації між незалежними процесами.

Клієнтська частина системи концептуально виділена в окремий односторінковий застосунок на базі бібліотеки React, який взаємодіє із серверною частиною виключно за допомогою прикладного програмного інтерфейсу передачі стану репрезентації. Такий підхід забезпечує розмежування відповідальності між візуальним фронтендом і прихованою бізнес-логікою бекенду, полегшує процеси паралельної розробки та дозволяє надалі легко доповнити клієнт мобільним застосунком без жодних змін у серверній частині.

Архітектура розроблюваної системи реалізує класичну клієнт-серверну модель взаємодії [10]. Серверна частина відіграє роль провайдера даних і бізнес-логіки шляхом надання відповідного інтерфейсу доступу. Клієнтська частина виступає безпосереднім споживачем цього інтерфейсу та відповідає за відображення інформації і взаємодію з користувачем. Вибір архітектурного стилю REST зумовлений його визнаною світовою простотою, універсальністю та повною стандартизованою підтримкою у всіх сучасних браузерях [11]. Кожен логічний ресурс системи ідентифікується унікальною мережевою адресою, тоді як стандартні методи протоколу HTTP визначають конкретний тип виконуваної операції.

Серверна частина системи побудована відповідно до суворих принципів

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						18

Чистої Архітектури з трьома чітко розмежованими та ізольованими шарами, до яких відносяться доменний, застосунковий та інфраструктурний рівні. Доменний шар є серцем системи і містить у собі ключові бізнес-сутності, об'єкти-значення та описи доменних подій, які детально моделюють предметну область управління науковим процесом без жодної прив'язки до конкретних технологій зберігання [12]. Ці агрегати виступають коренями власних меж транзакційної узгодженості і відповідають ключовим концептам академічного середовища.

Застосунковий шар безпосередньо реалізує всі можливі сценарії використання системи та координує логіку взаємодії між чистими доменними об'єктами та зовнішніми інструментами інфраструктури. Кожен такий сценарій відповідає за виконання однієї атомарної бізнес-операції на кшталт подання звіту або реєстрації нової наукової публікації. Завдяки механізму впровадження залежностей застосунковий шар взаємодіє з інфраструктурою виключно через абстрактні програмні інтерфейси [13].

Інфраструктурний шар знаходиться на самому зовнішньому контурі архітектури і вміщує конкретні технічні реалізації всіх абстракцій [14]. Тут розміщено класи для роботи з реляційною базою даних, модулі генерації криптографічних токенів безпеки та підсистеми аудиту. Важливою архітектурною особливістю є те, що цей шар залежить від внутрішніх рівнів бізнес-логіки, але не навпаки, що гарантує стійкість системи до зміни технологій у майбутньому.

Серверна частина організована у вісімнадцять предметно-орієнтованих модулів, які повністю охоплюють усі можливі аспекти життєдіяльності та звітності аспірантури. Кожен такий модуль є повністю самодостатньою алгоритмічною одиницею, яка самостійно визначає власні контролери доступу, внутрішні сервіси для обробки логіки та класи роботи з базою даних. З іншими частинами коду модулі спілкуються лише через суворо типізовані інтерфейси.

Багаторівнева система авторизації користувачів реалізована на основі

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						19

гнучкої рольової моделі доступу. Кожен вхідний мережевий запит послідовно проходить через складний захисний ланцюжок автоматичних перевірок, який включає валідацію криптографічного підпису токена доступу, підтвердження наявності необхідної ролі в ієрархії та фінальну перевірку прав власності на модифікацію конкретного об'єкта. Глобальний перехоплювач аудиту автоматично фіксує всі мутаційні операції в незмінному журналі із зазначенням ідентифікатора актора, типу зміненої сутності та точної часової мітки.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Вибір технологічного стека значною мірою визначає успіх реалізації програмного проєкту, тому він має спиратися на об'єктивне зіставлення можливостей кожної технології з функціональними вимогами системи, очікуваним масштабом застосування та наявними ресурсами на розробку й підтримку. Під час добору інструментів для системи управління аспірантською підготовкою розглядалися різні варіанти, проте вирішальними критеріями стали зрілість і стабільність технології, наявність активної спільноти розробників, якість документації та природна сумісність з обраною архітектурою. Додатковою перевагою стало застосування єдиної мови програмування TypeScript і на стороні сервера, і на стороні клієнта, що забезпечує наскрізну типізацію переданих між частинами системи даних та зменшує кількість помилок ще на етапі компіляції.

Платформа NestJS виступає прогресивним серверним фреймворком, який спонукає до модульної архітектури [15]. Це дозволяє максимально природно реалізувати розглянуті принципи Чистої Архітектури. Код організовується у вигляді ізольованих модулів, кожен з яких інкапсулює окрему ділянку предметної галузі, а контейнер інверсії управління автоматично постачає необхідні залежності в місця їхнього використання, завдяки чому компоненти залишаються слабо зв'язаними між собою та легко піддаються модульному тестуванню. Фреймворк також містить глибоко

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						20

інтегровану підтримку інструментів криптографічного захисту, конвеєри автоматичної валідації вхідних даних та шини подій для асинхронної обробки складних сценаріїв. Валідація відхиляє некоректні запити ще до того, як вони потраплять до бізнес-логіки, а шина подій дає змогу виконувати супутні дії, наприклад сповіщення наукового керівника про зміну статусу аспіранта, у фоновому режимі, не блокуючи основний хід виконання запиту.

Інструмент React являє собою сучасну бібліотеку для побудови користувацьких інтерфейсів на основі концепції компонентного підходу [16]. Таке рішення дозволяє інженеру конструювати складні візуальні інтерфейси шляхом композиції невеликих незалежних блоків коду. Кожен компонент описує свій фрагмент інтерфейсу декларативно, а бібліотека самостійно обчислює мінімально необхідні зміни та застосовує їх до сторінки, що особливо цінно для системи з чотирма типами користувачів, де спільні елементи інтерфейсу описуються один раз і повторно застосовуються в кабінетах аспіранта, завідувача кафедри, наукового керівника та адміністратора. Для ефективного управління асинхронним станом даних застосовується спеціалізована бібліотека React Query, яка бере на себе рутину з декларативного отримання інформації, розумного кешування запитів у пам'яті браузера та фонового оновлення інформації для підтримки її актуальності [17].

Надійне зберігання інформації забезпечується використанням потужної об'єктно-реляційної системи управління базами даних PostgreSQL [18]. Її вибір обумовлений тим, що реляційна модель є найбільш природною для моделювання предметної галузі навчального закладу з її складними багаторівневими зв'язками, а відношення між аспірантами, кафедрами, керівниками та наказами виражаються через зовнішні ключі й обмеження цілісності. Зазначена база даних повністю відповідає вимогам транзакційної цілісності та має вбудовану підтримку оптимізованих типів для безпечного збереження динамічних неструктурованих даних наказів. Зокрема, тип JSONB дозволяє зберігати наказ зі змінним набором полів як цілісний документ і

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						21

водночас будувати за його вмістом індекси для швидкого пошуку. Програмна взаємодія з таблицями бази реалізована за допомогою інструменту об'єктно-реляційного відображення TypeORM, який дозволяє зручно описувати структуру бази даних мовою програмування та автоматично генерувати необхідні міграції [19]. Сутності оголошуються у вигляді класів з відповідними анотаціями, а механізм міграцій фіксує кожну зміну схеми як окремий версіонований крок, який можна застосувати або скасувати.

Механізм аутентифікації базується на використанні сучасного галузевого стандарту безпечних веб-токенів із застосуванням схеми подвійного життєвого циклу. При успішному вході сервер видає короткоживучий токен доступу для виконання запитів та довгоживучий токен для фонових оновлень активної сесії. Такий розподіл обмежує вікно можливої компрометації, водночас не примушуючи користувача повторно вводити облікові дані. Токени містять у підписаному вигляді ідентифікатор користувача та закріплену за ним роль, завдяки чому охоронці доступу розмежовують повноваження між аспірантом, завідувачем кафедри, науковим керівником та адміністратором без звернення до бази даних під час кожного запиту. Паролі користувачів безповоротно хешуються з використанням стійкого алгоритму та унікальних сольових значень, що унеможливорює компрометацію паролів навіть у випадку витіку дампу бази даних.

Для забезпечення безперебійного розгортання та стабільної продуктивної роботи серверної частини інформаційної системи висуваються чіткі мінімальні вимоги до апаратного забезпечення цільової платформи. Обладнання повинно являти собою фізичний сервер або виділену хмарну інстанцію, яка обладнана щонайменше двоядерним процесором із базовою тактовою частотою від півтора гігагерца та містить оперативну пам'ять обсягом не менше двох гігабайт. Необхідний обсяг постійного дискового простору становить від двадцяти гігабайт для безпечного розміщення всіх даних, включно з базою, файлами застосунку, журналами роботи та резервними копіями. Програмним середовищем сервера обов'язково має

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						22

виступати операційна система сімейства Linux стабільних випусків, що зумовлено її надійністю за тривалої безперервної роботи та ощадливим використанням системних ресурсів. Для забезпечення стійкої роботи системи в періоди пікових навантажень під час сесійного періоду рекомендується використовувати серверне обладнання підвищеної продуктивності або хмарні потужності відповідного класу.

Висновки до першого розділу

У першому розділі випускної кваліфікаційної роботи проведено детальний аналіз предметної галузі, досліджено наявні програмні рішення та обґрунтовано ключові архітектурні й технологічні підходи для реалізації інформаційної системи обліку та моніторингу наукової діяльності аспірантів кафедри.

У результаті постановки задачі визначено чотири основні категорії користувачів системи, до яких належать аспірант, науковий керівник, завідувач кафедри та адміністратор, а також сформовано вичерпний перелік функціональних і нефункціональних вимог. Встановлено, що розроблений програмний продукт повинен забезпечувати гнучкий рольовий доступ, підтримувати повноцінний електронний документообіг звітності з реалізацією повного життєвого циклу погодження документів, автоматизувати облік наукових здобутків, формувати аналітичну звітність та гарантувати високу захищеність даних на основі використання безпечних токенів доступу і підсистеми незмінного аудиту операцій.

Проведений порівняльний аналіз найближчих аналогів переконливо засвідчив, що жоден із розглянутих класів програмних рішень, серед яких системи управління навчанням, корпоративні платформи управління проєктами, хмарні офісні пакети та глобальні інституційні системи управління дослідженнями, не відповідає повною мірою вимогам до супроводу підготовки здобувачів за українськими нормативними стандартами. Кожен із проаналізованих інструментів здатний охопити лише певну частину необхідного функціоналу, що безперечно підтверджує доцільність розробки

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						23

спеціалізованого програмного продукту, адаптованого під конкретні управлінські потреби університету.

Для логічної архітектурної організації інформаційної системи обрано модульну монолітну архітектуру з виділеним клієнтським односторінковим застосунком, що взаємодіє з серверним ядром за допомогою прикладного програмного інтерфейсу. Серверна частина побудована на суворих принципах Чистої Архітектури, яка передбачає розмежування коду на доменний, застосунковий та інфраструктурний шари. Така концептуальна організація проєкту дозволяє забезпечити чистоту алгоритмів бізнес-логіки, її незалежність від мінливих технологій зберігання даних та високу зручність подальшого розширення функціональних можливостей.

Комплексний технологічний стек, який включає мову програмування TypeScript, прогресивний серверний фреймворк NestJS, клієнтську бібліотеку React, систему управління базами даних PostgreSQL та інструмент TypeORM, був обраний як оптимальна сукупність зрілих і добре задокументованих технологій. Цей набір інструментів природно підтримує реалізацію визначеної архітектури моноліту, забезпечує високі показники безпеки і надійності системи, а також повністю відповідає мінімальним вимогам до апаратного забезпечення типового серверного обладнання закладу вищої освіти. Отримані аналітичні результати та зроблені висновки даного розділу слугують надійним інженерним підґрунтям для безпосереднього переходу до фази проєктування архітектурних діаграм і практичної розробки програмного забезпечення у наступних розділах кваліфікаційної роботи.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ОБЛІКУ ТА МОНІТОРИНГУ НАУКОВОЇ ДІЯЛЬНОСТІ АСПІРАНТІВ КАФЕДРИ

2.1. Визначення варіантів використання та структури системи

Моделювання варіантів використання (use case modeling) є невід'ємним етапом проєктування інформаційних систем і дозволяє зафіксувати функціональні вимоги у формі, зрозумілій як розробникам, так і замовникам. Методологічною основою для цього розділу слугує уніфікована мова моделювання UML, діаграми варіантів використання якої наочно відображають відносини між акторами і функціями системи. Кожна така діаграма містить діючу особу, що позначається стилізованою фігуркою, межу системи у вигляді прямокутника та сукупність варіантів використання, зображених еліпсами, причому асоціативні лінії між діючою особою і варіантами використання вказують, які функції є доступними для відповідної ролі [20].

У розроблюваній системі обліку та моніторингу наукової діяльності аспірантів виокремлено чотири діючі особи, що відповідають чотирьом ролям користувачів, а саме аспіранта, наукового керівника, завідувача кафедри та адміністратора. Оскільки сукупна кількість варіантів використання є значною, для збереження читабельності побудовано окрему діаграму для кожної діючої особи. Слід зауважити, що варіант використання, пов'язаний з автентифікацією, є спільним для всіх ролей, оскільки доступ до будь-якого функціоналу системи можливий лише після успішного входу, тому він повторюється на кожній діаграмі.

Діаграма варіантів використання для діючої особи «Аспірант» наведена на рис. 2.1. Аспірант є основним постачальником первинних даних у системі, тому коло доступних йому функцій є найширшим серед усіх ролей і охоплює дев'ять варіантів використання.

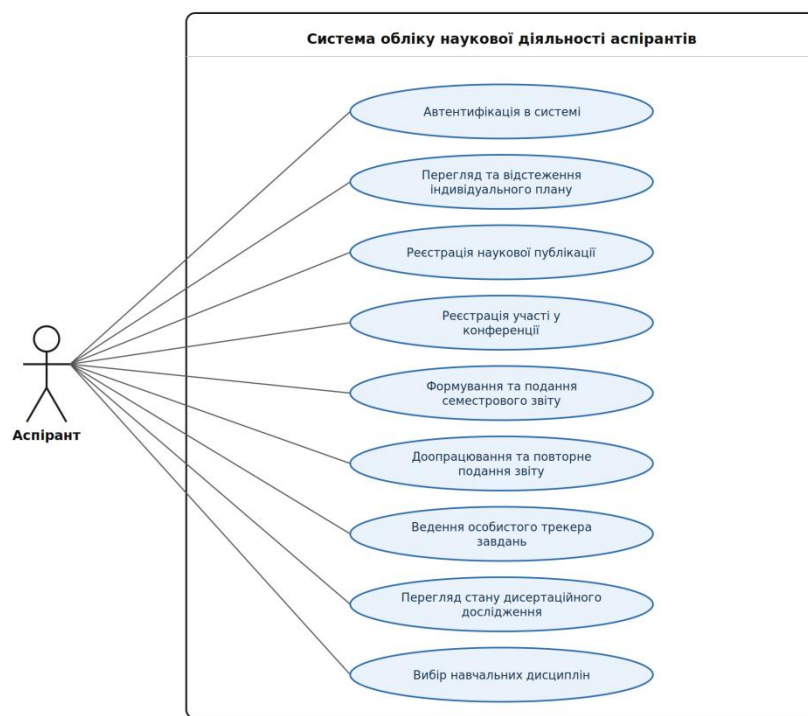


Рис. 2.1. Діаграма варіантів використання для актора «Аспірант»

Як видно з наведеної на рис. 2.1 діаграми, робота аспіранта із системою розпочинається з варіанту використання «Автентифікація в системі», що відкриває доступ до персонального кабінету. Після входу аспірант отримує можливість переглядати та відстежувати власний індивідуальний план, оновлюючи відсоток виконання його пунктів, а також реєструвати наукові публікації із зазначенням їхніх метаданих і фіксувати участь у конференціях. Окрему групу функцій становить робота зі звітністю, що представлена варіантами використання «Формування та подання семестрового звіту» і «Допрацювання та повторне подання звіту», які разом реалізують повний цикл взаємодії аспіранта з керівником щодо погодження звітів. Додатково аспірант веде особистий трекер завдань для самоорганізації, переглядає поточний стан свого дисертаційного дослідження в режимі лише читання та формує власну навчальну складову, обираючи дисципліни з каталогу освітньої програми. Така сукупність функцій повністю покриває потреби здобувача у самостійному веденні наукового портфолію та звітуванні перед кафедрою.

Діаграма варіантів використання для діючої особи «Науковий керівник»

зображена на рис. 2.2. Функції цієї ролі зосереджено навколо контролю та супроводу закріпленої за керівником групи аспірантів і охоплюють шість варіантів використання



Рис. 2.2. Діаграма варіантів використання для актора «Науковий керівник»

Згідно з наведеною на рис. 2.2 діаграмою, після автентифікації науковий керівник передусім отримує доступ до перегляду зведеної статистики своєї наукової групи, що дає змогу швидко оцінити загальний стан справ за всіма підопічними. Варіант використання «Моніторинг виконання індивідуальних планів» дозволяє керівникові не лише спостерігати за поступом кожного аспіранта, а й безпосередньо коригувати пункти планів, тоді як варіант використання «Рецензування та попереднє затвердження звіту» реалізує ключову контрольну функцію, у межах якої керівник перевіряє поданий звіт, залишає відгук і ухвалює рішення про його попереднє затвердження або повернення на допрацювання. Для зручності аналізу окремого здобувача передбачено варіант використання «Єдиний моніторинг аспіранта», що збирає всю інформацію про обраного підопічного в одному місці, а варіант використання «Моніторинг публікаційної та конференційної активності» надає зведені таблиці наукових результатів усієї групи з можливістю їх фільтрування й експорту.

Діаграма варіантів використання для діючої особи «Завідувач кафедри» подана на рис. 2.3. Ця роль зосереджує управлінські та адміністративні повноваження на рівні всього підрозділу й охоплює шість варіантів використання.

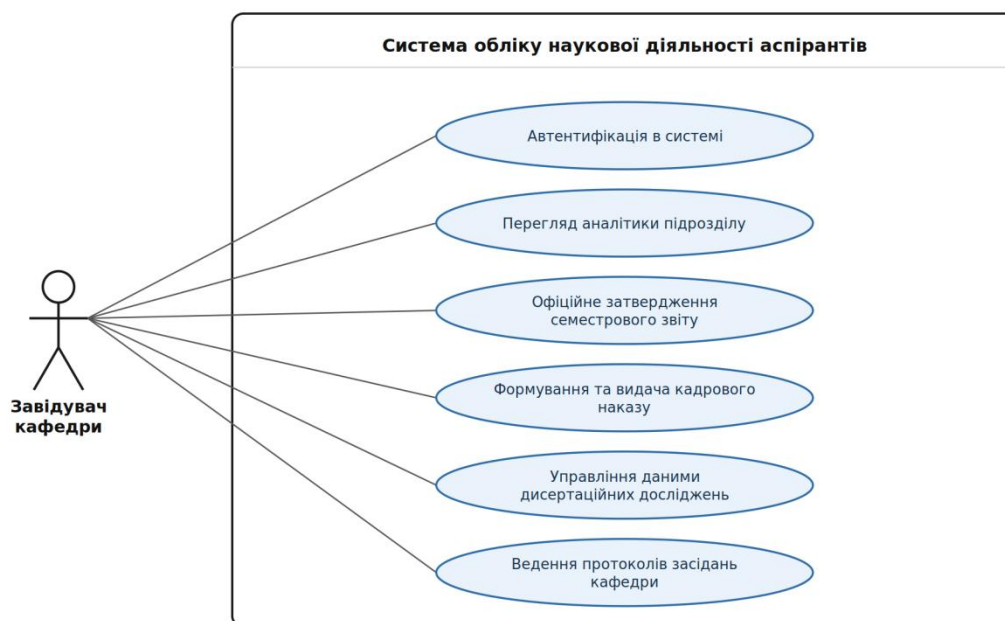


Рис. 2.3. Діаграма варіантів використання для актора «Завідувач кафедри»

Як ілюструє рис. 2.3, після входу до системи завідувач кафедри отримує доступ до перегляду аналітики підрозділу, де відображається консолідована статистика по всіх аспірантах та перелік справ, що потребують реагування. Центральне місце серед його повноважень посідає варіант використання «Офіційне затвердження семестрового звіту», у межах якого завідувач остаточно затверджує звіт із прив'язкою до реквізитів протоколу засідання кафедри, унаслідок чого звіт набуває юридичної значущості. Варіант використання «Формування та видача кадрового наказу» дозволяє оформлювати накази щодо зарахування, переведення, надання академічної відпустки чи відрахування аспірантів із автоматичним застосуванням відповідних змін до їхніх профілів. Завідувач також керує даними дисертаційних досліджень, фіксуючи етапи підготовки, результати антиплагіатної перевірки та реквізити захисту.

Діаграма варіантів використання для діючої особи «Адміністратор»

наведена на рис. 2.4. Адміністратор відповідає за технічне та структурне налаштування платформи, тому його функції мають переважно конфігураційний і наглядний характер та охоплюють п'ять варіантів використання.



Рис. 2.4. Діаграма варіантів використання для актора «Адміністратор»

Відповідно до наведеної на рис. 2.4 діаграми, після автентифікації адміністратор отримує доступ до управління обліковими записами користувачів, що передбачає їх реєстрацію, редагування, призначення ролей та деактивацію. Варіант використання «Ведення довідників системи» охоплює керування переліками кафедр, спеціальностей, освітньо-наукових програм і навчальних дисциплін, які становлять структурну основу для роботи решти модулів. Через варіант використання «Моніторинг та адміністрування системи» адміністратор має змогу переглядати вміст таблиць бази даних і незмінний журнал аудиту, що забезпечує прозорість та підзвітність усіх дій у системі. Варіант використання «Загальне затвердження звітів та управління наказами» наділяє адміністратора повноваженнями завідувача кафедри без обмеження за підрозділом, що створює резервний механізм виконання адміністративних функцій на випадок відсутності призначеного завідувача.

Для систематизації описаних варіантів використання й узагальнення всіх функцій системи незалежно від ролей у таблиці 2.1 наведено зведений перелік основних сценаріїв використання із зазначенням діючої особи,

найменування сценарію та його стислого формулювання. Така таблична форма доповнює графічні діаграми та слугує єдиним довідником функціональних можливостей системи на етапі її подальшого проєктування

Таблиця 2.1

Основні сценарії використання

Основний актор	Найменування	Формулювання
Аспірант	Автентифікація в системі	Вхід до системи за допомогою електронної пошти та пароля для отримання доступу до персонального кабінету відповідно до призначеної ролі
	Перегляд та відстеження індивідуального плану	Перегляд структурованого переліку завдань за семестрами й типами активностей та оновлення відсотка виконання кожного пункту плану
	Реєстрація наукової публікації	Унесення відомостей про опубліковану працю із зазначенням типу, видання, року, співавторів та наукометричних баз індексування
	Реєстрація участі у конференції	Унесення даних про науковий захід із зазначенням його рівня, ролі учасника, дедлайнів подання матеріалів та статусу участі
	Формування та подання семестрового звіту	Заповнення навчальної, наукової та дисертаційної складових звіту й передавання його на розгляд науковому керівнику
	Допрацювання та повторне подання звіту	Унесення правок до поверненого керівником звіту згідно з відгуком та повторне подання його на погодження
	Ведення особистого трекера завдань	Створення та відстеження довільних персональних завдань із зазначенням дедлайну й пріоритету для самоорганізації роботи
	Перегляд стану дисертаційного дослідження	Перегляд у режимі лише читання поточного етапу підготовки дисертації, результатів антиплагіатної перевірки та реквізитів захисту
	Вибір навчальних дисциплін	Додавання дисциплін навчальної програми до власного плану та перегляд набраних кредитів ECTS і отриманих оцінок

Науковий керівник	Автентифікація в системі	Вхід до системи за обліковим записом для отримання доступу до кабінету наукового керівника
	Перегляд статистики наукової групи	Перегляд зведених показників по всіх підопічних аспірантах на головній панелі кабінету керівника
	Моніторинг виконання індивідуальних планів	Перегляд та коригування пунктів індивідуальних планів закріплених за керівником аспірантів
	Рецензування та попереднє затвердження звіту	Перевірка поданого звіту, написання відгуку та попереднє затвердження або повернення звіту на допрацювання
	Єдиний моніторинг аспіранта	Отримання повної зведеної картини по обраному підопічному в єдиному розділі з вкладками за напрямками діяльності
	Моніторинг публікаційної та конференційної активності	Перегляд зведених таблиць публікацій і конференцій усієї наукової групи з фільтрацією, сортуванням та експортом даних
	Автентифікація в системі	Вхід до системи за обліковим записом для отримання доступу до кабінету завідувача кафедри
	Перегляд аналітики підрозділу	Перегляд консолідованої статистики кафедри та переліку справ, що потребують термінового реагування
	Офіційне затвердження семестрового звіту	Остаточне затвердження звіту аспіранта із прив'язкою до реквізитів протоколу засідання кафедри
	Формування та видача кадрового наказу	Створення наказів щодо зарахування, переведення, відпустки чи відрахування та застосування змін до профілю аспіранта
Завідувач кафедри	Управління даними дисертаційних досліджень	Унесення відомостей про етапи підготовки, антиплагіат, передзахист, допуск та результати захисту дисертацій
	Ведення протоколів засідань кафедри	Реєстрація протоколів засідань кафедри із зазначенням порядку денного та прийнятих рішень
Адміністратор	Автентифікація в системі	Вхід до системи за обліковим записом для отримання доступу до адміністративного кабінету

Управління обліковими записами користувачів	Реєстрація, редагування, призначення ролей та деактивація облікових записів усіх користувачів системи
Ведення довідників системи	Управління довідниками кафедр, спеціальностей, освітньо-наукових програм та навчальних дисциплін
Моніторинг та адміністрування системи	Доступ до перегляду таблиць бази даних та незмінного журналу аудиту мутаційних операцій
Загальне затвердження звітів та управління наказами	Затвердження звітів і видача наказів для аспірантів будь-якої кафедри без обмеження за підрозділом

Подані на рис. 2.1 – 2.4 діаграми разом зі зведеною таблицею 2.1 формують повну функціональну модель системи з погляду її користувачів і слугують підґрунтям для подальшого проектування структури бази даних та програмних модулів, що реалізують описані сценарії взаємодії.

Структура системи PhD Science Tracker реалізується за архітектурним патерном клієнт-сервер із відокремленими серверним і клієнтським застосунками [21]. На серверному рівні система організована як модульний моноліт із 16 функціональними модулями NestJS. Клієнтський рівень являє собою React SPA із захищеними маршрутами. Між рівнями розташований RESTful HTTP API з JWT-автентифікацією.

Серверні модулі організовані за доменними контекстами: AuthModule відповідає за автентифікацію і управління JWT; UserModule управляє обліковими записами; StudentModule і SupervisorModule зберігають профілі відповідних акторів; DepartmentModule управляє кафедрами і програмами; IndividualPlanModule реалізує логіку індивідуальних планів; ReportModule реалізує життєвий цикл звітів; PublicationModule і ConferenceModule ведуть облік наукових здобутків; DissertationModule відстежує дисертаційні дослідження; OrdersModule реалізує кадровий документообіг; AttestationModule і EducationModule управляють атестаціями та навчальними

дисциплінами; PersonalTaskModule реалізує особисті трекеři завдань; AuditLogModule реалізує журналювання; FileModule управляє завантаженими файлами.

Клієнтська частина структурована навколо єдиного Dashboard-компонента з бічною панеллю навігації. Маршрути поділяються на чотири групи за роллю: маршрути аспіранта (my-plan, my-publications, my-conferences, my-education, my-dissertation, my-task-tracker, attestations, my-reports), маршрути керівника (students, plans-monitoring, publications-monitoring, conferences-monitoring, education-monitoring, integrated-monitoring, supervisor-reports), маршрути завідувача (department-students, orders, dissertations, department-reports), маршрути адміністратора (users, supervisors, departments, specialties, programs, courses, database, audit-logs, theses). Доступ до кожного маршруту перевіряється функцією canAccessDashboardRoute перед відображенням компонента.

2.2. Розробка структури бази даних

Проектування структури бази даних є одним із визначальних етапів побудови надійної інформаційної системи, адже від якості схеми безпосередньо залежать цілісність збережених відомостей, швидкодія запитів та простота внесення майбутніх змін.

Вибір системи керування базами даних PostgreSQL зумовлений кількома міркуваннями, безпосередньо пов'язаними з особливостями предметної галузі. Передусім ця система повністю відповідає вимогам транзакційної моделі щодо атомарності, узгодженості, ізольованості та довговічності, що є критичним під час виконання складених операцій на кшталт одночасного видання наказу та зміни статусу аспіранта, які мають виконуватися як єдине ціле або не виконуватися взагалі. Крім того, вбудована підтримка типу універсального унікального ідентифікатора без додаткових розширень дозволяє генерувати глобально унікальні ключі на рівні застосунку, не покладаючись на автоінкремент бази, а тип JSONB у поєднанні з узагальненими інвертованими індексами забезпечує ефективний пошук за

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						33

вмістом напівструктурованих полів.

Загалом схема бази даних налічує двадцять чотири таблиці. Структурно таблиці можна умовно поділити на кілька груп, а саме на довідкові таблиці організаційної структури, таблиці облікових записів і ролей, таблиці академічного процесу, таблиці адміністративного діловодства та службові таблиці підтримки роботи системи.

Основу організаційної структури формує таблиця departments, яка призначена для збереження відомостей про кафедри університету, зокрема їхній унікальний код, назву та посилання на користувача-завідувача через відповідне поле headId, що детально відображено у таблиці 2.2.

Таблиця 2.2

Опис таблиці departments

Назва поля	Тип	Опис
<i>1.</i>	<i>2.</i>	<i>3.</i>
id	uuid (PK)	Унікальний ідентифікатор кафедри
code	varchar(50)	Унікальний код кафедри
name	varchar(255)	Назва кафедри
headId	uuid (FK)	Посилання на користувача-завідувача кафедри
createdAt	timestamp	Дата та час створення запису

Таблиця specialties зберігає академічні спеціальності з їхніми кодами й назвами та пов'язує кожну спеціальність із кафедрою через departmentId та з освітньою програмою через programId, як продемонстровано у таблиці 2.3.

Таблиця 2.3

Опис таблиці specialties

Назва поля	Тип	Опис
<i>1.</i>	<i>2.</i>	<i>3.</i>
id	uuid (PK)	Унікальний ідентифікатор спеціальності
code	varchar(20)	Код спеціальності
name	varchar(255)	Назва спеціальності
departmentId	uuid (FK)	Кафедра, до якої належить спеціальність

1.	2.	3.
programId	uuid (FK)	Посилання на освітньо-наукову програму
createdAt	timestamp	Дата та час створення запису
updatedAt	timestamp	Дата та час останнього оновлення запису

Таблиця programs описує освітньо-наукові програми підготовки докторів філософії й містить, окрім назви та рівня освіти, перелік мов викладання та допустимі межі кредитів, що детально описано у таблиці 2.4.

Таблиця 2.4

Опис таблиці programs

Назва поля	Тип	Опис
id	integer (PK)	Унікальний ідентифікатор програми
name	varchar(255)	Назва освітньо-наукової програми
description	text	Опис програми
degreeLevel	varchar(50)	Рівень освіти (за замовчуванням PhD)
Languages	jsonb	Перелік мов викладання
educationalEctsMin	integer	Мінімальна кількість кредитів освітньої складової
educationalEctsMax	integer	Максимальна кількість кредитів освітньої складової
electiveCreditsMinPercent	integer	Мінімальний відсоток вибіркового кредитів
requiresSemiAnnualAttestation	boolean	Ознака потреби в семестровій атестації
specialtyId	uuid (FK)	Посилання на спеціальність
createdAt	timestamp	Дата та час створення запису
updatedAt	timestamp	Дата та час останнього оновлення запису

Таблиця cohorts описує набори, під якими слід розуміти групи аспірантів, що навчаються за однією програмою й вступили в один період, і містить дати початку та завершення строку навчання, що дозволяє автоматично визначати академічні роки набору, а її структура представлена у таблиці 2.5.

Таблиця 2.5.

Опис таблиці cohorts

Назва поля	Тип	Опис
<i>1.</i>	<i>2.</i>	<i>3.</i>
id	uuid (PK)	Унікальний ідентифікатор спеціальності
name	varchar(255)	Назва групи
programId	uuid (FK)	Посилання на освітньо-наукову програму
startDate	date	Початок навчання
expiritionDate	date	Кінець навчання групи
createdAt	timestamp	Дата та час створення запису
updatedAt	timestamp	Дата та час останнього оновлення запису

Центральною сутністю підсистеми автентифікації та авторизації є таблиця `users`, навколо якої побудовано керування доступом. У ній зберігаються електронна пошта як унікальний ідентифікатор для входу, хеш пароля, повне ім'я та роль користувача, що набуває одного з чотирьох значень переліку, причому детальну структуру наведено у таблиці 2.6.

Таблиця 2.6

Опис таблиці users

Назва поля	Тип	Опис
id	uuid (PK)	Унікальний ідентифікатор користувача
email	varchar(255)	Електронна пошта, що слугує логіном (унікальна)
passwordHash	varchar(255)	Хешований пароль користувача (bcrypt)
fullName	varchar(255)	Повне ім'я користувача
role	enum	Роль користувача: admin, supervisor, department_head, student
departmentId	uuid (FK)	Кафедра, до якої належить користувач
isActive	boolean	Ознака активності облікового запису
lastLoginAt	timestamp	Дата та час останнього входу в систему
createdAt	timestamp	Дата та час створення запису
updatedAt	timestamp	Дата та час останнього оновлення запису

Таблиця supervisors розширює обліковий запис відомостями наукового керівника, посилаючись на користувача через унікальне поле userId у

співвідношенні один до одного та зберігаючи науковий ступінь, кафедру, статус і граничну кількість підопічних, що детально відображено у таблиці 2.7.

Таблиця 2.7

Опис таблиці supervisors

Назва поля	Тип	Опис
1.	2.	3.
id	uuid (PK)	Ідентифікатор профілю
userId	uuid (FK)	Посилання на обліковий запис (унікальне, зв'язок один-до-одного)
academicTitle	varchar(255)	Науковий ступінь та вчене звання
specialization	text	Наукова спеціалізація керівника
status	enum	Статус керівника
maxStudents	integer	Гранична кількість аспірантів
isProfileComplete	boolean	Ознака повноти заповнення профілю
createdAt	timestamp	Дата та час створення запису
updatedAt	timestamp	Дата та час останнього оновлення запису

Таблиця phd_students аналогічно розширює профіль аспіранта, пов'язуючись із обліковим записом через userId, із науковим керівником через supervisorId та з набором через cohortId, і зберігає рік вступу, орієнтовну дату захисту й статус згідно з таблицею 2.8.

Таблиця 2.8.

Опис таблиці phd_students

Назва поля	Тип	Опис
id	uuid (PK)	Унікальний ідентифікатор профілю аспіранта
enrollmentYear	integer	Рік вступу до аспірантури
supervisorId	uuid (FK)	Посилання на наукового керівника
defensePlannedAt	timestamp	Орієнтовна запланована дата захисту
status	enum	Статус: активний, у відпустці, відрахований, випущений
cohortId	uuid (FK)	Група
isProfileComplete	boolean	Ознака повноти заповнення профілю
createdAt	timestamp	Дата та час створення запису
updatedAt	timestamp	Дата та час останнього оновлення запису

Найбільшу за обсягом групу становлять таблиці академічного процесу.

Таблиця individual_plans містить індивідуальні плани аспірантів із розбивкою

за академічним роком та семестром, фіксує загальний відсоток виконання й статус погодження, а також посилання на користувачів, що схвалили та завізували план, як показано у таблиці 2.9.

Таблиця 2.9

Опис таблиці individual_plans

Назва поля	Тип	Опис
id	uuid (PK)	Унікальний ідентифікатор плану
studentId	uuid (FK)	Посилання на аспіранта
supervisorId	uuid (FK)	Посилання на наукового керівника
academicYear	varchar(20)	Академічний рік
semester	enum	Семестр
status	enum	Статус плану
completionPct	integer	Відсоток виконання плану
description	text	Опис плану
approvedAt	timestamp	Дата затвердження плану
approvedBy	uuid (FK)	Користувач, що затвердив план
supervisorEndorsedAt	timestamp	Дата візування плану керівником
supervisorEndorsedBy	uuid (FK)	Керівник, що завізував план
createdAt	timestamp	Дата та час створення запису
updatedAt	timestamp	Дата та час останнього оновлення запису

Деталізацію плану забезпечує таблиця plan_tasks, у якій кожне завдання прив'язане до плану через planId і має назву, опис, статус, тип активності, порядковий номер, оцінений обсяг годин та граничну дату виконання, що продемонстровано у таблиці 2.10.

Таблиця 2.10

Опис таблиці plan_tasks

Назва поля	Тип	Опис
id	uuid (PK)	Унікальний ідентифікатор завдання
title	varchar(255)	Назва завдання
description	text	Опис завдання
status	enum	Статус (не розпочато, в процесі, виконано)
orderIndex	integer	Порядковий номер завдання у плані
estimatedHours	integer	Оцінений обсяг годин на виконання
dueDate	date	Гранична дата виконання
completedDate	date	Фактична дата виконання
completionPct	integer	Відсоток виконання завдання
type	enum	Тип активності (наукова, навчальна тощо)
createdAt	timestamp	Дата та час створення запису
updatedAt	timestamp	Дата та час останнього оновлення запису

Обговорення завдань ведеться в таблиці task_comments, кожен запис якої посилається на завдання та автора коментаря згідно зі структурою у таблиці 2.11.

Таблиця 2.11

Опис таблиці task_comments

Назва поля	Тип	Опис
id	uuid (PK)	Унікальний ідентифікатор коментаря
taskId	uuid (FK)	Посилання на завдання плану
authorId	uuid (FK)	Автор коментаря
content	text	Текст коментаря
createdAt	timestamp	Дата та час створення коментаря

Звітність здобувачів зберігається у двох пов'язаних таблицях. Таблиця reports містить семестрові звіти аспірантів із полями академічного року, семестру, текстового змісту, оцінки, обмеженої перевіркою діапазону, а також часовими мітками подання, попереднього та остаточного затвердження, відгуком керівника й реквізитами протоколу, що відображено у таблиці 2.12.

Таблиця 2.12

Опис таблиці reports

Назва поля	Тип	Опис
id	uuid (PK)	Унікальний ідентифікатор звіту
studentId	uuid (FK)	Посилання на аспіранта
supervisorId	uuid (FK)	Посилання на наукового керівника
semester	smallint	Номер семестру
academicYear	varchar(20)	Академічний рік
status	enum	Статус (чернетка, подано, погоджено, затверджено, повернено)
content	text	Текстовий зміст звіту
grade	integer	Оцінка від 0 до 100
submittedAt	timestamp	Дата подання звіту
supervisorApprovedAt	timestamp	Дата попереднього затвердження керівником
approvedAt	timestamp	Дата офіційного затвердження
rejectionReason	varchar(500)	Причина повернення звіту
supervisorOpinion	text	Відгук наукового керівника
protocolNumber	varchar(50)	Номер протоколу засідання
protocolDate	date	Дата протоколу засідання
createdAt	timestamp	Дата та час створення запису
updatedAt	timestamp	Дата та час останнього оновлення запису

Таблиця attestations фіксує семестрові атестації здобувачів, об'єднуючи посилання на звіт, офіційний план, керівника та протокол засідання й зберігаючи рішення кафедри, рекомендацію керівника та поточний статус атестації відповідно до таблиці 2.13.

Таблиця 2.13

Опис таблиці attestations

Назва поля	Тип	Опис
1.	2.	3.
id	uuid (PK)	Унікальний ідентифікатор атестації

1.	2.	3.
studentId	uuid (FK)	Посилання на аспіранта
academicYear	varchar(20)	Академічний рік
semester	enum	Семестр
status	enum	Статус атестації
reportId	uuid (FK)	Посилання на семестровий звіт
officialPlanId	uuid (FK)	Посилання на офіційний індивідуальний план
studentSummary	text	Узагальнення, надане аспірантом
supervisorId	uuid (FK)	Посилання на наукового керівника
supervisorOpinion	text	Відгук наукового керівника
supervisorRecommendation	varchar(100)	Рекомендація наукового керівника
supervisorReviewedAt	timestamp	Дата розгляду керівником
departmentDecision	text	Рішення кафедри
departmentComment	text	Коментар кафедри
protocolNumber	varchar(50)	Номер протоколу засідання
protocolDate	date	Дата протоколу засідання
submittedAt	timestamp	Дата подання атестації
attestedAt	timestamp	Дата проходження атестації
createdAt	timestamp	Дата та час створення запису
updatedAt	timestamp	Дата та час оновлення запису

Облік наукових результатів реалізовано у трьох таблицях. Таблиця publications зберігає метадані публікацій, серед яких назва, перелік авторів, тип публікації, ідентифікатор цифрового об'єкта, відомості про видання та статус індексування у наукометричних базах, що описано у таблиці 2.14.

Таблиця 2.14

Опис таблиці publications

Назва поля	Тип	Опис
1.	2.	3.
id	uuid (PK)	Унікальний ідентифікатор публікації

1.	2.	3.
studentId	uuid (FK)	Посилання на аспіранта
title	varchar(500)	Назва публікації
authors	text	Перелік авторів
type	enum	Тип публікації
doi	varchar(100)	Ідентифікатор цифрового об'єкта (DOI)
journalName	varchar(255)	Назва видання
journalNumber	varchar(50)	Номер видання
publishedYear	integer	Рік публікації
volume	varchar(100)	Том видання
pages	varchar(100)	Сторінки публікації
indexStatus	enum	Статус індексування у наукометричних базах
url	varchar(500)	Посилання на публікацію
createdAt	timestamp	Дата та час створення запису
updatedAt	timestamp	Дата та час останнього оновлення запису

Таблиця conference_participations фіксує участь у наукових заходах, зберігаючи назву конференції, місце й дати проведення, назву доповіді, тип участі та посилання на офіційний сайт, як показано у таблиці 2.15.

Таблиця 2.15

Опис таблиці conference_participations

Назва поля	Тип	Опис
1.	2.	3.
id	uuid (PK)	Унікальний ідентифікатор участі
studentId	uuid (FK)	Посилання на аспіранта
conferenceName	varchar(500)	Назва конференції
location	varchar(255)	Місце проведення
paperTitle	varchar(500)	Назва доповіді

1.	2.	3.
participationType	enum	Тип участі (доповідач, слухач тощо)
eventDate	date	Дата заходу
startDate	date	Дата початку заходу
endDate	date	Дата завершення заходу
url	varchar(500)	Посилання на офіційний сайт
notes	text	Додаткові примітки
createdAt	timestamp	Дата та час створення запису
updatedAt	timestamp	Дата та час останнього оновлення запису

Таблиця dissertations описує дисертаційні дослідження з обмеженням не більше однієї картки на аспіранта й містить тему, поточний етап підготовки, відсоток унікальності за результатами антиплагіатної перевірки, реквізити передзахисту, допуску до захисту та фінального захисту, а також посилання на повний текст дисертації й автореферату згідно з таблицею 2.16.

Таблиця 2.16

Опис таблиці dissertations

Назва поля	Тип	Опис
1.	2.	3.
id	uuid (PK)	Унікальний ідентифікатор дисертації
studentId	uuid (FK)	Посилання на аспіранта
topic	varchar(500)	Тема дисертації
description	text	Опис дослідження
currentStage	enum	Поточний етап підготовки
topicApprovedAt	timestamp	Дата затвердження теми
topicApprovedBy	uuid (FK)	Користувач, що затвердив тему
defenseScheduledAt	timestamp	Запланована дата захисту
defenseCompletedAt	timestamp	Дата проведення захисту
result	varchar(100)	Результат захисту

1.	2.	3.
defenseComments	text	Коментарі щодо захисту
plagiarismPercentage	numeric(5,2)	Відсоток унікальності за антиплагіатом
dissertationTextUrl	varchar(500)	Посилання на повний текст дисертації
abstractTextUrl	varchar(500)	Посилання на автореферат
preDefenseScheduledAt	timestamp	Запланована дата передзахисту
preDefenseCompletedAt	timestamp	Дата проведення передзахисту
preDefenseDecision	varchar(100)	Рішення передзахисту
preDefenseComments	text	Коментарі передзахисту
admissionStatus	varchar(100)	Статус допуску до захисту
admissionProtocolNumber	varchar(100)	Номер протоколу допуску
admissionProtocolDate	date	Дата протоколу допуску
admittedToDefenseAt	timestamp	Дата допуску до захисту
specializedCouncilName	varchar(255)	Назва спеціалізованої вченої ради
defenseVenue	varchar(255)	Місце проведення захисту
createdAt	timestamp	Дата та час створення запису
updatedAt	timestamp	Дата та час останнього оновлення запису

Навчальну складову підготовки обслуговують чотири таблиці. Таблиця courses є каталогом навчальних дисциплін із зазначенням назви, опису та кількості кредитів ECTS, що відображено у таблиці 2.17.

Таблиця 2.17

Опис таблиці courses

Назва поля	Тип	Опис
1.	2.	3.
id	uuid (PK)	Унікальний ідентифікатор дисципліни
name	varchar(255)	Назва дисципліни
description	text	Опис дисципліни

продовж.табл. 2.17

1.	2.	3.
ectsCredits	integer	Кількість кредитів ECTS
createdAt	timestamp	Дата та час створення запису
updatedAt	timestamp	Дата та час останнього оновлення запису

Таблиця enrollments фіксує записи аспірантів на дисципліни в розрізі академічного року й семестру, причому унікальний складений індекс запобігає повторному запису на ту саму дисципліну, як показано у таблиці 2.18.

Таблиця 2.18

Опис таблиці enrollments

Назва поля	Тип	Опис
id	uuid (PK)	Унікальний ідентифікатор запису на дисципліну
studentId	uuid (FK)	Посилання на аспіранта
courseId	uuid (FK)	Посилання на дисципліну
academicYear	varchar(20)	Академічний рік
semester	integer	Семестр
status	enum	Статус
createdAt	timestamp	Дата та час створення запису
updatedAt	timestamp	Дата та час останнього оновлення запису

Таблиця grades зберігає оцінки за пройденої дисципліни у співвідношенні один до одного із записом на дисципліну через унікальне поле enrollmentId, фіксуючи бал, оцінку за шкалою ECTS та дату іспиту згідно з таблицею 2.19.

Таблиця 2.19

Опис таблиці grades

Назва поля	Тип	Опис
1.	2.	3.
id	uuid (PK)	Унікальний ідентифікатор оцінки

1.	2.	3.
enrollmentId	uuid (FK)	Посилання на запис на дисципліну
score	integer	Бал за дисципліну
ectsGrade	varchar(2)	Оцінка за шкалою ECTS
comments	text	Коментарі до оцінки
examDate	date	Дата іспиту
createdAt	timestamp	Дата та час створення запису
updatedAt	timestamp	Дата та час останнього оновлення запису

Таблиця `course_group_assignments` закріплює дисципліни за когортами аспірантів, забезпечуючи централізоване формування навчальних планів наборів відповідно до структури у таблиці 2.20.

Таблиця 2.20

Опис таблиці `course_group_assignments`

Назва поля	Тип	Опис
id	uuid (PK)	Унікальний ідентифікатор закріплення
courseId	uuid (FK)	Посилання на дисципліну
cohortId	uuid (FK)	Посилання на набір (когорту)
academicYear	varchar(20)	Академічний рік
semester	integer	Семестр
createdAt	timestamp	Дата та час створення запису
updatedAt	timestamp	Дата та час останнього оновлення запису

Адміністративне діловодство представлене трьома таблицями. Таблиця `orders` реалізує концепцію двоетапного застосування кадрових рішень і містить поле `payload` типу `JSONB` для зберігання специфічних для кожного типу наказу даних, а поле `status` розмежовує чернетки від виданих наказів, що детально описано у таблиці 2.21.

Таблиця 2.21

Опис таблиці orders

Назва поля	Тип	Опис
id	uuid (PK)	Унікальний ідентифікатор наказу
orderNumber	varchar(50)	Номер наказу
date	date	Дата наказу
type	enum	Тип наказу
status	enum	Статус (чернетка / видано)
studentId	uuid (FK)	Посилання на аспіранта
payload	jsonb	Додаткові дані наказу залежно від його типу
description	text	Опис наказу
fileIds	jsonb	Перелік прикріплених файлів
createdAt	timestamp	Дата та час створення запису
updatedAt	timestamp	Дата та час останнього оновлення запису

Завершують схему службові таблиці, що підтримують роботу системи. Таблиця `personal_tasks` зберігає особисті завдання аспірантів для самоорганізації, які можуть бути необов'язково пов'язані з пунктом плану чи звітом через відповідні зовнішні ключі, що продемонстровано у таблиці 2.22.

Таблиця 2.22

Опис таблиці personal_tasks

Назва поля	Тип	Опис
1.	2.	3.
id	uuid (PK)	Унікальний ідентифікатор завдання
studentId	uuid (FK)	Посилання на аспіранта
title	varchar(255)	Назва завдання
description	text	Опис завдання
status	enum	Статус завдання
category	enum	Категорія завдання

1.	2.	3.
dueDate	date	Гранична дата виконання
relatedPlanId	uuid (FK)	Пов'язаний індивідуальний план
relatedReportId	uuid (FK)	Пов'язаний звіт
createdAt	timestamp	Дата та час створення запису
updatedAt	timestamp	Дата та час оновлення

Таблиця files зберігає метадані завантажених файлів разом із посиланням на власника, а вміст самих файлів розташовується у файловій системі, що наведено у таблиці 2.23.

Таблиця 2.23

Опис таблиці files

Назва поля	Тип	Опис
id	uuid (PK)	Унікальний ідентифікатор файлу
ownerId	uuid (FK)	Посилання на власника файлу
originalName	varchar(255)	Оригінальна назва файлу
filename	varchar(255)	Назва файлу у сховищі
contentType	varchar(100)	MIME-тип файлу
size	bigint	Розмір файлу в байтах
path	varchar(500)	Шлях до файлу у сховищі
createdAt	timestamp	Дата та час створення запису
updatedAt	timestamp	Дата та час останнього оновлення

Таблиця audit_logs веде незмінний журнал усіх мутаційних операцій, фіксуючи виконавця, тип і ідентифікатор зміненої сутності, вид дії та тіло запиту, що детально відображено у таблиці 2.24.

Таблиця 2.24

Опис таблиці audit_logs

Назва поля	Тип	Опис
1.	2.	3.
id	uuid (PK)	Унікальний ідентифікатор запису журналу

1.	2.	3.
actorId	uuid (FK)	Посилання на користувача-виконавця дії
entityType	varchar(100)	Тип зміненої сутності
entityId	uuid	Ідентифікатор зміненої сутності
action	enum	Вид дії
payload	jsonb	Тіло запиту зі станом сутності
createdAt	timestamp	Дата та час виконання дії

Службова таблиця migrations веде реєстр застосованих до схеми міграцій, що продемонстровано у таблиці 2.25.

Таблиця 2.25

Опис таблиці migrations

Назва поля	Тип	Опис
id	serial (PK)	Унікальний ідентифікатор міграції
timestamp	bigint	Часова мітка застосування міграції
name	varchar	Назва міграції

Реляційні зв'язки між таблицями формують чітку ієрархію, у якій таблиця departments виступає батьківською для користувачів, керівників і спеціальностей, таблиця users є батьківською для профілів керівників та аспірантів, а таблиця phd_students виступає центральним вузлом, до якого прив'язано індивідуальні плани, звіти, атестації, публікації, конференції, дисертації, записи на дисципліни, накази та особисті завдання.

2.3. Проєктування та реалізація алгоритмів роботи системи

Алгоритм є формальним описом послідовності кроків для вирішення конкретної задачі [22]. У системі PhD Science Tracker реалізовано п'ять ключових алгоритмів, що разом забезпечують основну функціональну логіку платформи. До них належать алгоритм автентифікації на основі двох токенів, скінченний автомат статусів звіту аспіранта, скінченний автомат статусів самого аспіранта, алгоритм транзакційної видачі кадрового наказу і алгоритм агрегації аналітичних даних для дашбордів. Кожен із цих алгоритмів виконує

самостійну роль і разом вони утворюють основу бізнес-логіки системи.

Алгоритм автентифікації базується на стандарті JSON Web Token, що визначений у специфікації RFC 7519 [23]. Суть підходу полягає у тому, що після успішного входу сервер видає користувачу два токени одночасно. Перший токен називається access token і має короткий термін дії, а саме 15 хвилин. Другий токен називається refresh token і має термін дії 7 діб. Використання двох токенів замість одного є кращою практикою, оскільки короткий термін дії access token обмежує потенційну шкоду у разі його перехоплення злоумисником.

Access token є підписаним рядком, що містить ідентифікатор користувача і його роль. Сервер перевіряє підпис токена за секретним ключем без звернення до бази даних, що робить авторизацію кожного запиту дуже швидкою. Клієнт передає access token у заголовок Authorization до кожного запиту. Refresh token зберігається на клієнті, а на сервері зберігається лише його хеш. При отриманні запиту на оновлення сервер перевіряє підпис наданого refresh token і порівнює його з хешем у базі. Якщо перевірка успішна, генерується нова пара токенів і старий хеш замінюється новим.

Паролі зберігаються у базі даних виключно у вигляді хешу, що обчислений алгоритмом bcrypt з коефіцієнтом складності 12 [24]. При вході в систему введений пароль хешується і порівнюється зі збереженим хешем. Bcrypt навмисно є повільним алгоритмом, що робить атаки перебором на вкрадену базу даних практично неможливими. Якщо введені облікові дані не відповідають жодному запису, сервер повертає однакову загальну помилку незалежно від того, чи не існує електронна пошта, чи неправильний пароль, щоб не розкривати зайву інформацію.

Оновлення сесії виконується автоматично і є непомітним для користувача. Якщо черговий запит клієнта повертає помилку з кодом 401, клієнтський інтерцептор автоматично надсилає запит на оновлення токенів і після отримання нових токенів повторює початковий запит. Завдяки цьому користувач продовжує роботу без переривання, хоча його access token вже

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						50

закінчився. При виході з системи хеш refresh token видаляється з бази, що робить старі токени недійсними навіть якщо вони ще не закінчились.

Скінченний автомат є математичною моделлю, що описує систему з кінцевою кількістю станів і чіткими правилами переходів між ними [25]. Статус звіту аспіранта реалізовано як такий автомат з базовими станами, а саме DRAFT, SUBMITTED, SUPERVISOR_APPROVED, REVISION і APPROVED. Автомат гарантує, що процес погодження звіту завжди проходить у правильній послідовності і жоден крок не може бути пропущений або виконаний в неправильному порядку.

Стан DRAFT є початковим станом нового звіту. У цьому стані аспірант заповнює три розділи звіту і може зберігати зміни скільки завгодно разів. Коли аспірант готовий подати звіт, система перевіряє, що всі три розділи заповнені, і виконує перехід до стану SUBMITTED. Якщо хоча б один розділ порожній, перехід блокується.

У стані SUBMITTED аспірант більше не може редагувати текст звіту. Науковий керівник розглядає звіт і може виконати один з двох переходів. При позитивному рішенні звіт переходить до стану SUPERVISOR_APPROVED. При поверненні на допрацювання звіт переходить до стану REVISION, а поле коментаря стає обов'язковим.

У стані REVISION аспірант знову отримує доступ до редагування звіту і може виправити зауваження. Після внесення змін аспірант повторно подає звіт, що переводить його назад у стан SUBMITTED. Цей цикл може повторюватись будь-яку кількість разів. Стан SUPERVISOR_APPROVED є проміжним і означає передачу звіту на офіційне затвердження завідувачу кафедри. Завідувач вказує номер протоколу засідання і дату, після чого звіт переходить до кінцевого стану APPROVED і більше не може бути змінений.

Перед виконанням будь-якого переходу система звіряється з матрицею дозволених переходів. Якщо запитуваний перехід відсутній у матриці для поточного стану, операція відхиляється і сервер повертає помилку. Це повністю виключає ситуації некоректних змін стану незалежно від того, хто

надсилає запит.

Академічний статус аспіранта також реалізовано як скінченний автомат з станами, а саме `active`, `on_leave`, `expelled` і `graduated`. Статус `active` є початковим і означає, що аспірант навчається. Стани `expelled` і `graduated` є кінцевими і не передбачають подальших переходів.

Прив'язка зміни статусу до наказу є навмисним рішенням, що забезпечує документування кожної зміни. Кожен наказ зберігає тип, підставу, дати і підпис завідувача. Завдяки цьому у системі зберігається повна і незмінна хронологія академічних статусів кожного аспіранта, що є важливим для офіційного діловодства кафедри.

Наказ про надання академічної відпустки переводить аспіранта зі статусу `active` до статусу `on_leave`. Цей перехід можливий лише для аспіранта з поточним статусом `active`. Наказ про поновлення після академічної відпустки виконує зворотний перехід і повертає статус `active`. Наказ про відрахування переводить аспіранта до статусу `expelled`, що є одним із кінцевих станів автомата. Наказ про закінчення навчання переводить до статусу `graduated`, що також є кінцевим станом.

Видача кадрового наказу змінює одночасно два записи у базі даних, а саме статус самого наказу і статус аспіранта. Для забезпечення узгодженості цих змін операція виконується у транзакції PostgreSQL. Транзакція гарантує, що або обидва записи оновлені успішно, або жоден з них не змінений. Якщо при оновленні будь-якого запису виникає помилка, вся транзакція відкочується і база даних повертається до попереднього стану.

Перед виконанням видачі система перевіряє, що наказ ще не виданий. Якщо наказ вже має статус виданого, операція відхиляється з відповідною помилкою. Це запобігає повторному застосуванню того самого наказу, що могло б призвести до некоректного стану аспіранта. Після успішної видачі наказу у журнал аудиту автоматично записується відповідний запис і аспіранту надсилається сповіщення про зміну його академічного статусу.

Алгоритм агрегації формує показники для головних панелей різних

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						52

ролей і виконує обчислення безпосередньо на рівні бази даних. Для аспіранта розраховується середній відсоток виконання плану поточного семестру як середнє арифметичне поля `completion_percentage` усіх пунктів плану. Також підраховується кількість публікацій і конференцій та їх розподіл за типами і базами індексування.

Для наукового керівника агрегуються показники по аспірантах, а саме загальна кількість аспірантів, кількість звітів у стані SUBMITTED і сумарна кількість публікацій Scopus і WoS за поточний рік. Для завідувача кафедри показники агрегуються на рівні всього підрозділу і включають кількість аспірантів за кожним статусом, публікації кафедри за рік, кількість звітів на затвердженні і розподіл навантаження між науковими керівниками.

2.4. Реалізація інформаційної системи

Реалізація системи PhD Science Tracker здійснена з використанням сучасного стеку технологій, що забезпечує надійність, безпеку і зручність подальшого розширення. Серверна частина розроблена на фреймворку NestJS версії 10 з використанням мови TypeScript [27]. Клієнтська частина розроблена на бібліотеці React 18 з інструментом збірки Vite 5 [28]. Вибір цих технологій обґрунтований у першому розділі роботи. Нижче описано деталі архітектурних рішень і конкретних технічних підходів, застосованих при реалізації кожної частини системи.

NestJS є фреймворком для Node.js, що забезпечує структуровану систему модулів і механізм ін'єкції залежностей. Суворий режим TypeScript вимагає явної типізації всіх змінних і параметрів, що дозволяє виявляти помилки на етапі компіляції. Архітектура серверної частини побудована відповідно до доменно-орієнтованого проєктування [29]. Код кожного модуля розділено на три шари, а саме доменний, прикладний і інфраструктурний, що залежать один від одного лише в одному напрямку.

Кожен функціональний домен реалізований як окремий модуль NestJS. Усього в системі є 9 модулів, серед яких AuthModule, StudentsModule, ReportsModule, PublicationsModule, ConferencesModule, PlanModule,

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						53

OrdersModule, NotificationsModule і AuditModule. Кожен модуль є самодостатньою одиницею, що явно оголошує свої компоненти і залежності. Такий поділ відповідає принципу єдиної відповідальності і спрощує розуміння і супроводження коду. Загальна діаграма класів наведена на рисунку 2.5.

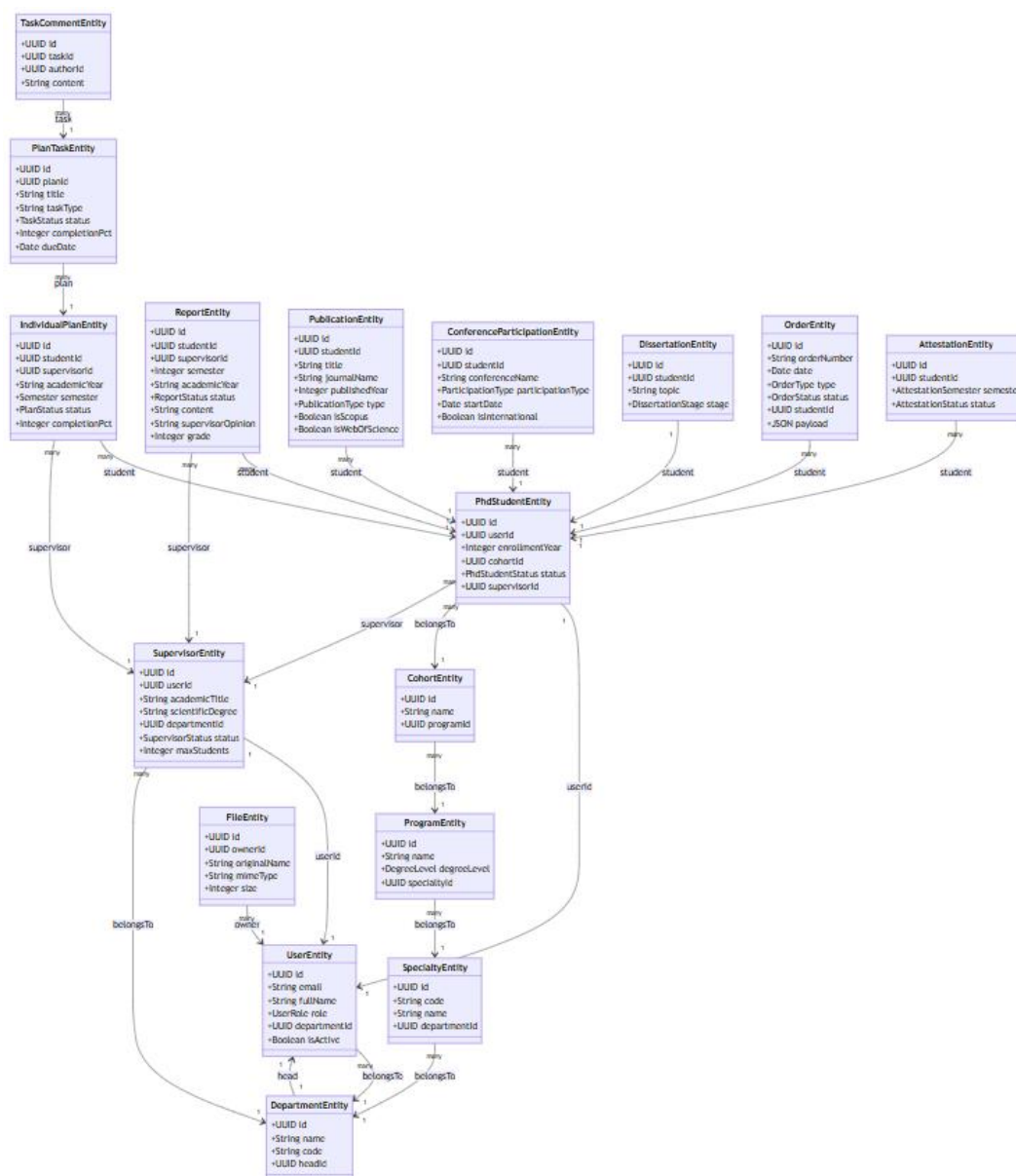


Рис. 2.5. Діаграма класів

Доменний шар містить клас агрегату і інтерфейс репозиторію. Клас агрегату інкапсулює бізнес-правила, що стосуються конкретної сутності, і не залежить від жодних зовнішніх бібліотек. Стан агрегату змінюється лише через публічні методи, що перед виконанням перевіряють бізнес-правила і інваріанти. Наприклад, метод подання звіту перевіряє заповненість усіх

розділів і допустимість переходу зі стану DRAFT, і лише потім застосовує зміну.

При зміні стану агрегат генерує доменну подію, що описує факт, який відбувся. Ці події публікуються у шину подій після успішного збереження агрегату і дозволяють іншим модулям реагувати на зміни без прямих залежностей між собою. Наприклад, при публікації події ReportSubmittedEvent модуль сповіщень автоматично надсилає повідомлення науковому керівнику, а модуль аудиту записує відповідний запис у журнал.

Інтерфейс репозиторію в доменному шарі визначає набір методів для роботи зі сховищем без вказівки конкретної технології. Завдяки цьому бізнес-логіка в прикладному шарі не залежить від TypeORM або PostgreSQL і може бути протестована через підміну репозиторію тестовою заглушкою. Конкретна реалізація інтерфейсу знаходиться в інфраструктурному шарі і підключається через механізм ін'єкції залежностей NestJS.

Прикладний шар містить класи варіантів використання, де кожен клас реалізує одну конкретну операцію системи. Наприклад, SubmitReportUseCase відповідає виключно за подання звіту, ApproveReportUseCase за попереднє затвердження. Кожен клас завантажує агрегат через інтерфейс репозиторію, перевіряє права поточного користувача, викликає відповідний метод агрегату і зберігає результат. Такий підхід робить кожен варіант використання незалежним і добре читабельним.

Інфраструктурний шар реалізує взаємодію з базою даних через TypeORM. Кожна таблиця описана у вигляді TypeScript-класу з декораторами, що визначають колонки, типи даних і зв'язки між таблицями. Репозиторій в інфраструктурному шарі реалізує інтерфейс з доменного шару і перетворює сутності TypeORM на доменні агрегати і навпаки. Цей підхід називається патерн Data Mapper і повністю розділяє схему бази даних і доменну модель.

Захист від несанкціонованого доступу реалізований на трьох рівнях. Перший рівень є глобальним охоронцем JwtAuthGuard, що перевіряє підпис JWT-токена для кожного запиту до API. Маршрути, що не потребують

автентифікації, явно позначаються декоратором Public. Другий рівень є охоронцем RolesGuard, що порівнює роль поточного користувача з переліком дозволених ролей для конкретного маршруту і блокує запит при невідповідності.

Третій рівень є інтерцептором OwnershipCheckInterceptor, що захищає від атак типу IDOR. Суть атаки полягає у підміні ідентифікатора ресурсу в URL для доступу до чужих даних. Інтерцептор завантажує ресурс за ідентифікатором з URL і порівнює власника ресурсу з поточним користувачем. Якщо вони не збігаються, запит блокується з кодом 403. Адміністратори і завідувачі кафедри мають право доступу до ресурсів підопічних і тому проходять перевірку без обмежень.

Глобальний інтерцептор AuditInterceptor перехоплює всі успішно виконані запити методів POST, PATCH і DELETE і записує до таблиці audit_logs тип сутності, ідентифікатор запису, тип дії, ідентифікатор актора і IP-адресу. Журнал аудиту є незмінним і доступний лише адміністратору. Захист від SQL-ін'єкцій забезпечується параметризованими запитами TypeORM. Захист від XSS забезпечується React, що екранує всі рядкові значення при рендерингу. Захист від CSRF забезпечується передачею токена через заголовок Authorization, який браузер не надсилає автоматично при міжсайтових запитах [30].

Клієнтська частина є односторінковим застосунком, що виконується у браузері. Vite 5 забезпечує швидкий сервер розробки і ефективну виробничу збірку. Навігація реалізована через React Router v7, де маршрути утворюють три незалежні гілки залежно від ролі користувача. Захищені маршрути загорнуті у компонент ProtectedRoute, що при відсутності автентифікації або невідповідній ролі автоматично перенаправляє на сторінку входу.

Стан автентифікації управляється через React Context. Провайдер AuthProvider при завантаженні перевіряє наявність токена у локальному сховищі браузера і відновлює сесію без повторного входу. Хук useAuth надає компонентам доступ до об'єкта поточного користувача з ідентифікатором,

ім'ям, роллю і ідентифікатором кафедри. Ці дані використовуються для формування навігаційного меню і фільтрації контенту відповідно до ролі.

Завантаження даних з сервера реалізоване через TanStack Query v5. Хук useQuery приймає унікальний ключ і функцію запиту і повертає стан завантаження, дані або помилку. Кешування за ключем запобігає зайвим запитам при повторному відображенні компонента. Хук useMutation виконує операції запису і при успіху інвалідує кеш пов'язаних запитів, що автоматично оновлює відображені дані без ручної синхронізації стану.

Форми реалізовані через бібліотеку react-hook-form з валідацією схем zod. Схема zod описує тип і правила кожного поля, а при надсиланні форми бібліотека автоматично перевіряє всі поля і відображає підказки при помилках. Форма не надсилає запит до сервера до успішного проходження валідації. Обробка помилок від сервера реалізована через колбек onError хука useMutation, що відображає текст помилки у вигляді спливаючого повідомлення.

Висновки до другого розділу

У другому розділі здійснено комплексне проектування веб-системи PhD Science Tracker за трьома взаємозалежними напрямками: функціональним, архітектурою даних і алгоритмічним забезпеченням, з докладним описом технічної реалізації.

Розроблено й описано детальні сценарії взаємодії для чотирьох типів користувачів системи з позначенням вихідних умов, основного ходу дій та можливих відхилень. Впорядкована сукупність сценаріїв охоплює весь спектр функцій системи і служить основою для розроблення тестів. Представлено тришарову архітектуру: представницький рівень (клієнт), проміжний шар (REST API) й серверні компоненти, організовані за принципом доменно-керованого дизайну.

Побудовано структуру реляційної бази даних із повним описом параметрів і форматів основних об'єктів, аргументовано застосування JSONB для частково упорядкованої інформації та UUID для ідентифікаторів записів,

визначено індексну стратегію. Деталізовано зв'язки між таблицями.

Розроблено та детально викладено п'ять центральних алгоритмів: система автентифікації на основі двох токенів з механізмом автоматичного поновлення; стан звіту з переходами; двофазний процес обробки кадрових документів; механізм збирання й аналізу даних для аналітичних панелей. Наведено деталі розроблення як серверної, так і клієнтської компонент, включаючи структуру модулів.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ

3.1. Порядок встановлення та налаштування параметрів системи

Система PhD Science Tracker складається з двох частин, а саме серверної і клієнтської. Серверна частина є застосунком NestJS, що виконується у середовищі Node.js і спілкується з базою даних PostgreSQL. Клієнтська частина є застосунком React, що збирається інструментом Vite і відкривається у браузері користувача. Обидві частини розміщені в одному репозиторії і встановлюються незалежно одна від одної.

Перед встановленням системи на комп'ютері або сервері має бути встановлено кілька програм. По-перше, необхідний Node.js версії 18 або вище, оскільки серверна частина і інструмент збірки клієнта виконуються саме в цьому середовищі. По-друге, необхідна СУБД PostgreSQL версії 14 або вище, в якій зберігаються всі дані системи. По-третє, необхідний Git для завантаження вихідного коду з репозиторію. Наявність встановленого Node.js можна перевірити командою `node`, а PostgreSQL запускається як системна служба і після встановлення зазвичай стартує автоматично.

Версії Node.js нижче 18 не підтримуються, оскільки серверна частина використовує вбудовані можливості платформи, що з'явилися саме у цій версії. Рекомендується встановлювати Node.js через менеджер версій NVM, що дозволяє легко переключатись між версіями і не вимагає прав адміністратора для встановлення глобальних пакетів. Для PostgreSQL підходить будь-яка версія від 14 і вище, оскільки система використовує стандартні можливості SQL без специфічних функцій більш нових версій.

Вихідний код системи завантажується з репозиторію через Git. Після клонування репозиторію у вибраній директорії з'являється папка проекту, що містить дві підпапки, а саме `server` для серверної частини і `client` для клієнтської. Всередині кожної підпапки знаходиться файл `package.json`, що описує залежності відповідної частини. Решта файлів є вихідним кодом і конфігурацією застосунку.

Встановлення залежностей виконується окремо для серверної і клієнтської частин. Для серверної частини потрібно перейти у папку `server` і виконати команду `npm install`. Менеджер пакетів `npm` завантажить всі необхідні бібліотеки, що описані у файлі `package.json`, і розмістить їх у папці `node_modules`. Цей процес займає кілька хвилин залежно від швидкості інтернет-з'єднання. Після завершення у консолі з'являється повідомлення про кількість встановлених пакетів.

Для клієнтської частини аналогічним чином виконується команда `npm install` у папці `client`. Обсяг залежностей клієнтської частини менший, тому встановлення займає менше часу. Після встановлення залежностей обидві частини готові до налаштування конфігурації.

Перед запуском серверної частини необхідно створити базу даних у PostgreSQL. Для цього потрібно підключитись до PostgreSQL від імені адміністратора і виконати команди створення користувача і бази даних. Рекомендується створити окремого користувача бази даних спеціально для цього застосунку, а не використовувати адміністративний обліковий запис. Це є стандартною практикою безпеки, що обмежує можливі наслідки у разі компрометації облікових даних застосунку.

Назва бази даних, ім'я користувача і пароль можуть бути довільними і вибираються на розсуд адміністратора. Єдина вимога до пароля полягає у тому, що він має бути достатньо складним і не збігатись з паролями інших систем. Після створення бази даних і користувача їх назви вказуються у файлі конфігурації застосунку.

Конфігурація серверної частини задається через файл `.env`, що розміщується у кореневій папці серверної частини поруч з файлом `package.json`. Якщо файл відсутній, його необхідно створити вручну або скопіювати з прикладу `.env.example`, що знаходиться у репозиторії. Файл `.env` містить змінні оточення у форматі `назва=значення`, кожна на окремому рядку. Цей файл не потрапляє до репозиторію і не передається іншим розробникам, оскільки містить секретні дані.

Змінні DB_HOST, DB_PORT, DB_USERNAME, DB_PASSWORD і DB_DATABASE описують підключення до бази даних. Значення DB_HOST зазвичай є localhost при запуску на тій самій машині, де встановлений PostgreSQL. Значення DB_PORT є стандартним портом PostgreSQL 5432. Значення DB_USERNAME, DB_PASSWORD і DB_DATABASE мають відповідати тому, що було вказано при створенні бази даних у попередньому кроці.

Змінні JWT_SECRET і JWT_REFRESH_SECRET є секретними ключами для підписання токенів автентифікації. Вони мають бути довгими випадковими рядками довжиною мінімум 32 символи. Для кожного розгортання системи ці значення мають бути унікальними і зберігатись у таємниці. Якщо ці ключі стануть відомі стороннім особам, вони зможуть підробляти токени автентифікації і отримувати несанкціонований доступ до системи. Змінна PORT визначає порт, на якому слухатиме серверний застосунок, і за замовчуванням дорівнює 3000.

Для клієнтської частини налаштовується лише одна змінна оточення, а саме VITE_API_URL, що вказує адресу серверного API. При локальному запуску це значення зазвичай є http://localhost з номером порту серверної частини. Ця змінна вшивається у збірку клієнта під час компіляції і визначає, куди клієнт надсилатиме запити.

Після налаштування конфігурації необхідно створити структуру бази даних через TypeORM-міграції. Міграції є спеціальними скриптами, що створюють таблиці, індекси і обмеження відповідно до схеми системи. Для виконання міграцій у папці server запускається команда npm run migration:run. При успішному виконанні у консолі виводиться перелік застосованих міграцій і повідомлення про завершення. Якщо виникає помилка підключення до бази, необхідно перевірити правильність параметрів у файлі .env.

Після виконання міграцій база даних містить порожні таблиці зі схемою. Для того щоб одразу мати можливість увійти в систему і перевірити її роботу, виконується скрипт-сід командою npm run seed:run. Цей скрипт заповнює базу

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						61

тестовими обліковими записами для кожної ролі, а також зразковими довідниковими даними, а саме кафедрами, спеціальностями і освітніми програмами. Завдяки цьому система одразу готова до демонстрації і тестування без ручного введення даних.

Скрипт-сід створює чотири тестові облікові записи. Перший є обліковим записом адміністратора з роллю ADMIN, другий є обліковим записом завідувача кафедри з роллю DEPARTMENT_HEAD, третій є обліковим записом наукового керівника з роллю SUPERVISOR і четвертий є обліковим записом аспіранта з роллю STUDENT. Кожен обліковий запис має тимчасовий пароль, що вказаний у документації. Ці паролі необхідно змінити після першого входу в систему.

Для запуску серверної частини у режимі розробки у папці server виконується команда `npm run start : dev`. У цьому режимі сервер автоматично перезапускається при зміні будь-якого файлу вихідного коду, що зручно при розробці і налагодженні. При успішному запуску у консолі з'являється повідомлення про те, що NestJS застосунок запущено і слухає на вказаному порту. Сервер готовий до прийому запитів і можна переходити до запуску клієнтської частини.

Для запуску клієнтської частини у папці client виконується команда `npm run dev`. Vite запускає сервер розробки і відображає адресу, за якою відкривається застосунок у браузері, зазвичай це `http://localhost` з портом 5173. Відкривши цю адресу у браузері, користувач бачить сторінку авторизації системи PhD Science Tracker. Клієнтська частина також підтримує гаряче оновлення і автоматично відображає зміни у вихідному коді без перезавантаження сторінки.

3.2. Структура інтерфейсу

Структура інтерфейсу вебсистеми обліку та моніторингу наукової діяльності аспірантів розроблена з урахуванням принципів зручності використання, логічного розподілу функціональних модулів та забезпечення роботи користувачів з різними рівнями доступу. Система включає декілька

ключових модулів, кожен з яких відповідає за певну область управління науково-освітнім процесом.

Кожен з представлених нижче екранів реалізує чітко визначений функціональний блок і є частиною єдиної навігаційної структури адміністративної панелі. Перехід між модулями здійснюється через бічне навігаційне меню, яке залишається видимим на всіх сторінках системи та дозволяє переключатися між розділами без втрати контексту поточної роботи. Наведені нижче рисунки ілюструють основні екрани вебсистеми з детальним поясненням елементів інтерфейсу, доступних дій та призначення кожного компонента в загальній архітектурі продукту.

Модуль управління користувачами (рис. 3.1) представляє структурований реєстр усіх учасників системи з можливістю пошуку, редагування та фільтрування за ролями та статусами. Таблична форма подання дозволяє адміністратору отримувати повний огляд активності кожного користувача і контролювати права доступу.

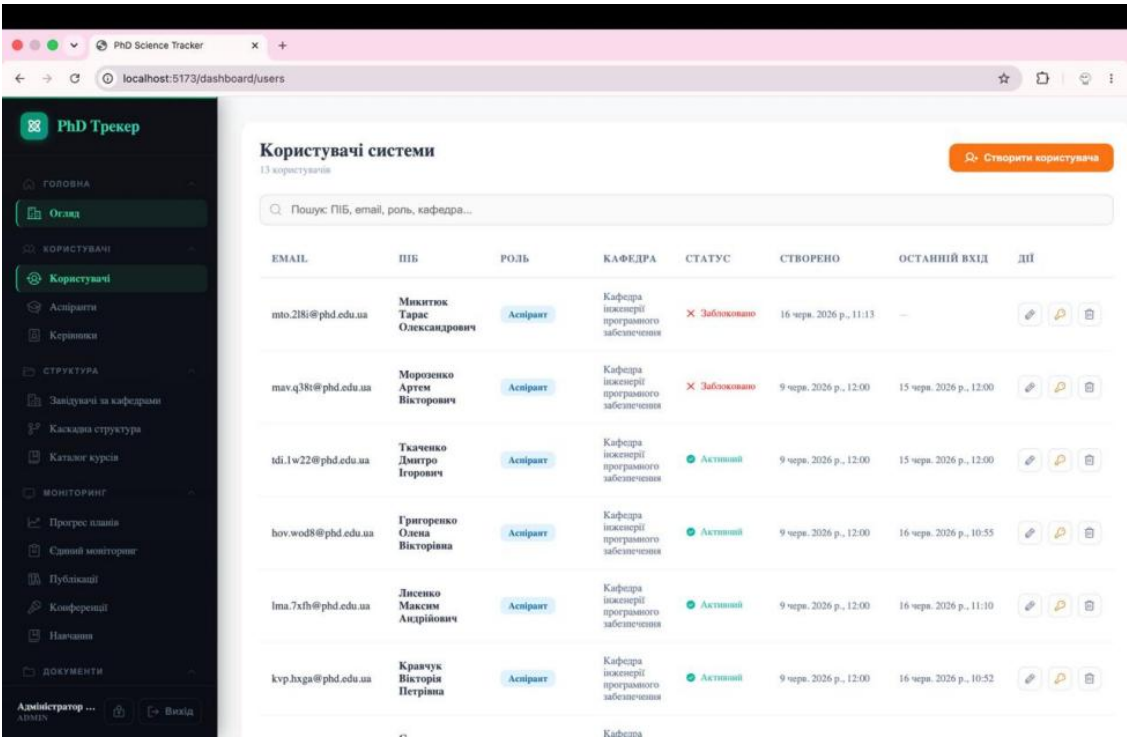


Рис. 3.1. Модуль управління користувачами

Розділ наукових керівників (рис. 3.2) показує список закріплених за системою викладачів у форматі карток з візуалізацією їхнього навантаження,

статусу активності та профільної спеціалізації, що полегшує розподіл аспірантів між керівниками.

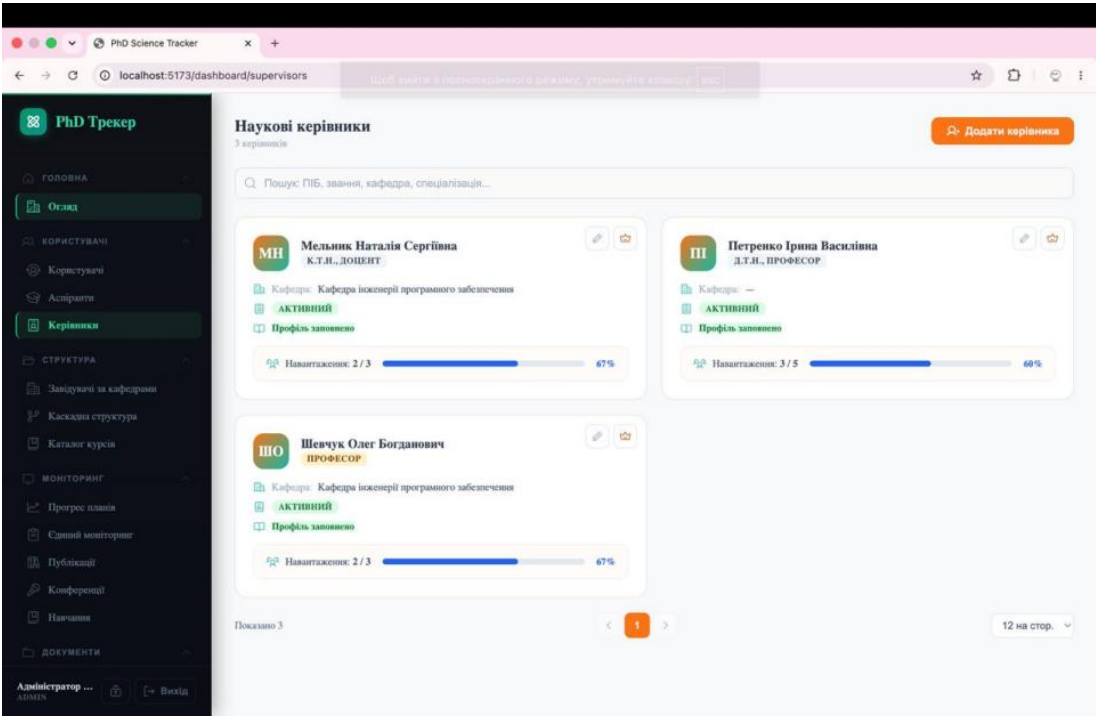


Рис. 3.2. Розділ управління науковими керівниками у вигляді карток з індикаторами навантаження

Реєстр аспірантів (рис. 3.3) представляє портфоліо всіх здобувачів науково-освітнього рівня, організовано у форматі карток з фільтрацією за статусами прогресу. На сторінці реалізовано розділення за ключовими статусами: активні, ті, що отримали академічну випуску, випущені та відраховані. Кожна картка відображає базову інформацію, закріплену кафедру, спеціальність, рік вступу та наукового керівника. Функціональні кнопки «Доброочесність» та «Відрахування» дозволяють позначити динаміку в навчально-науковій діяльності, а індикатор «Профіль заповнено» показує повноту інформації про аспіранта в системі.

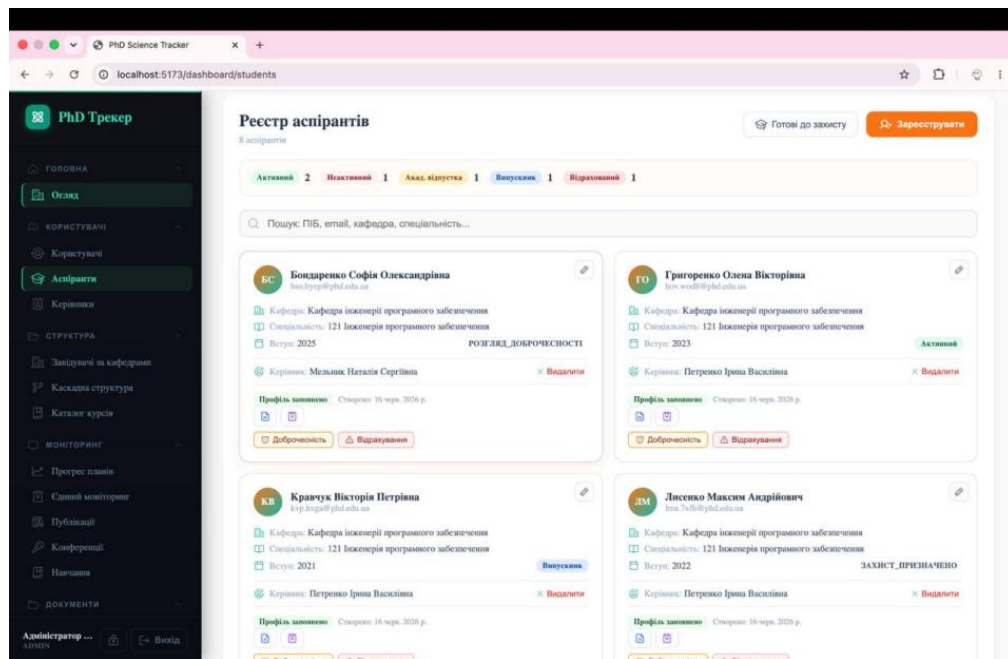


Рис. 3.3. Реєстр аспірантів

Управління кафедрою (рис. 3.4) дозволяє адміністратору переглядати та призначати завідувачів.

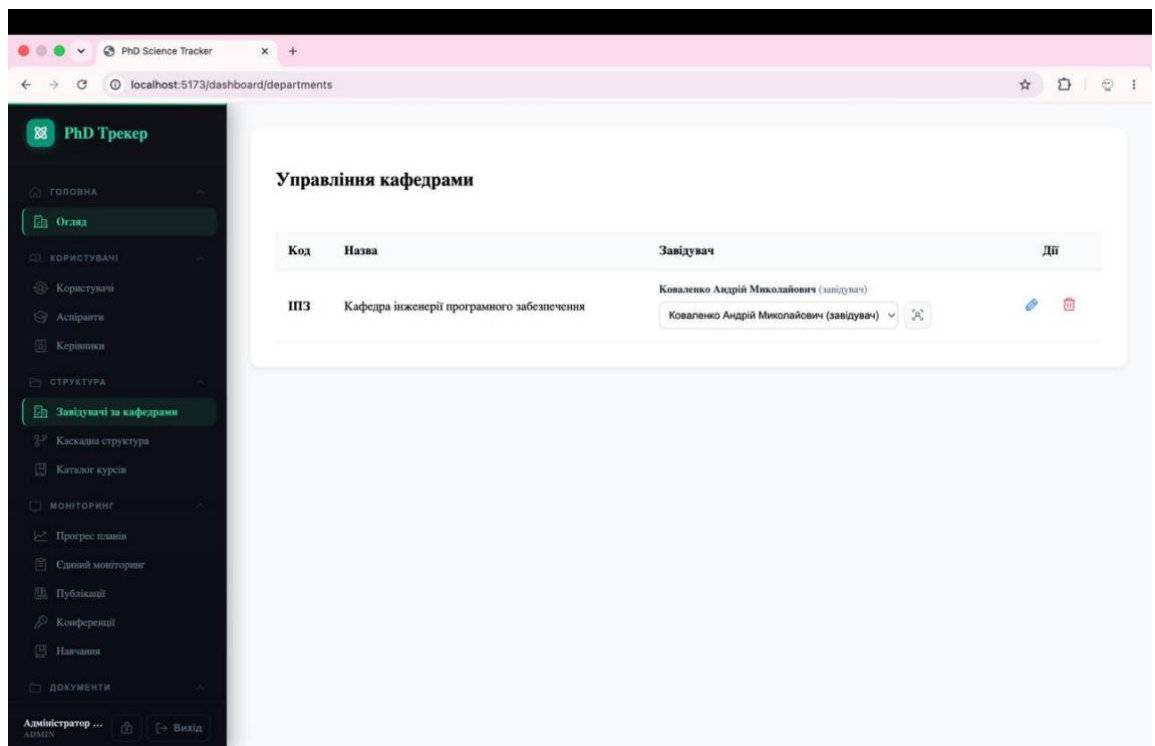


Рис. 3.4. Сторінка для призначення завідувачів кафедри

Каскадна структура (рис. 3.5) розширює функціонал управління кафедрами, надаючи детальний перегляд багаторівневої організації: від кафедри через спеціальності та освітні програми до груп, з можливістю редагування на кожному рівні.

Каталог дисциплін (рис. 3.7) служить довідковою базою всіх навчальних курсів з позначенням кредитів ЄКТС, типів курсів та опису змісту, необхідною для формування навчальних планів підготовки аспірантів.

НАЗВА	ECTS	ОПИС	ТИП	ДІ
Іноземна мова академічного спрямування обов'язк.	6	Академічне письмо та презентація результатів	Загальна	[іконки]
Інтелектуальний аналіз даних	5	Методи машинного навчання та аналізу даних	Фах. іспит	[іконки]
Методологія та організація наукових досліджень обов'язк.	5	Методи й інструменти наукового дослідження	Загальна	[іконки]
Педагогіка та психологія вищої школи	5	Підготовка до педагогічної практики	Загальна	[іконки]
Сучасні технології розробки програмного забезпечення	5	Поглиблене вивчення інженерії ПЗ	Фах. іспит	[іконки]
Філософія науки обов'язк.	5	Філософські засади наукового пізнання	Загальна	[іконки]

Рис. 3.7. Каталог дисциплін

Єдиний моніторинг (рис. 3.8) реалізує двопанельний інтерфейс пошуку та детального перегляду прогресу кожного аспіранта, включаючи розподілене за вкладками відображення планів, звітів (рис. 3.9), публікацій, конференцій та навчань зі можливістю залишення рекомендацій і коментарів науковим керівником прямо в системі.

затвердження науковим керівником і погодження завідувачем кафедри.

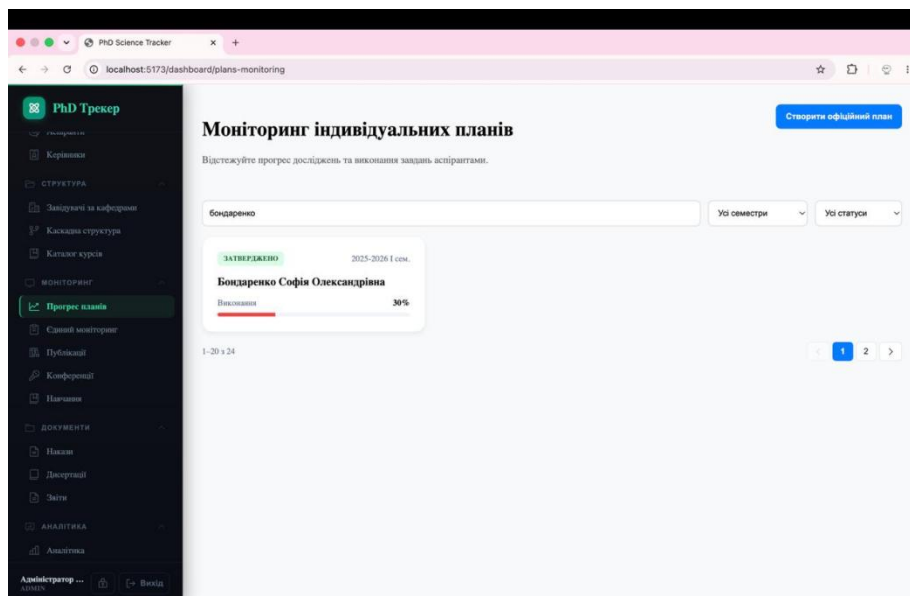


Рис. 3.10. Сторінка для відслідковування прогресу індивідуальних планів.

Модуль моніторингу навчальної складової (рис. 3.11) показує реєстр успішності аспірантів із деталізацією результатів у модальному вікні, де фіксуються підсумкові бали та оцінки ECTS, що спрощує контроль за виконанням освітньої програми.

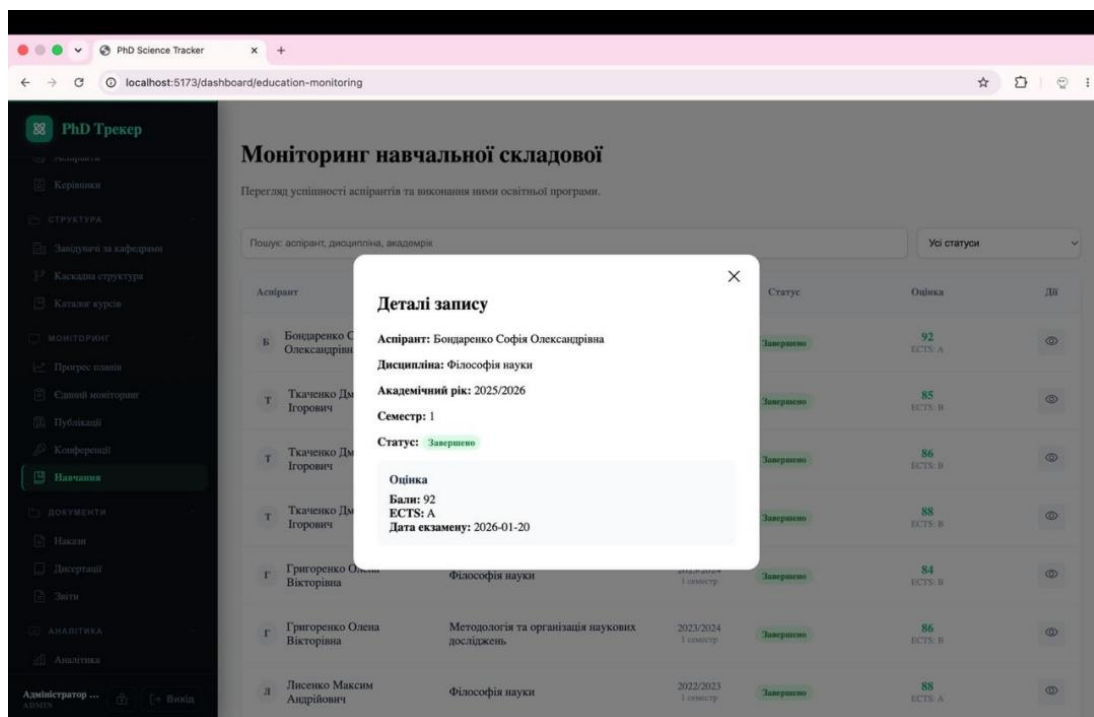


Рис. 3.11. Модуль моніторингу навчальної складової

Модуль наказів (рис. 3.12) показує інструмент формування офіційних розпоряджень у форматі модального вікна з вбудованою системою перевірки

статусів аспіранта, що автоматично блокує суперечливі дії та мінімізує помилки в діловодстві.

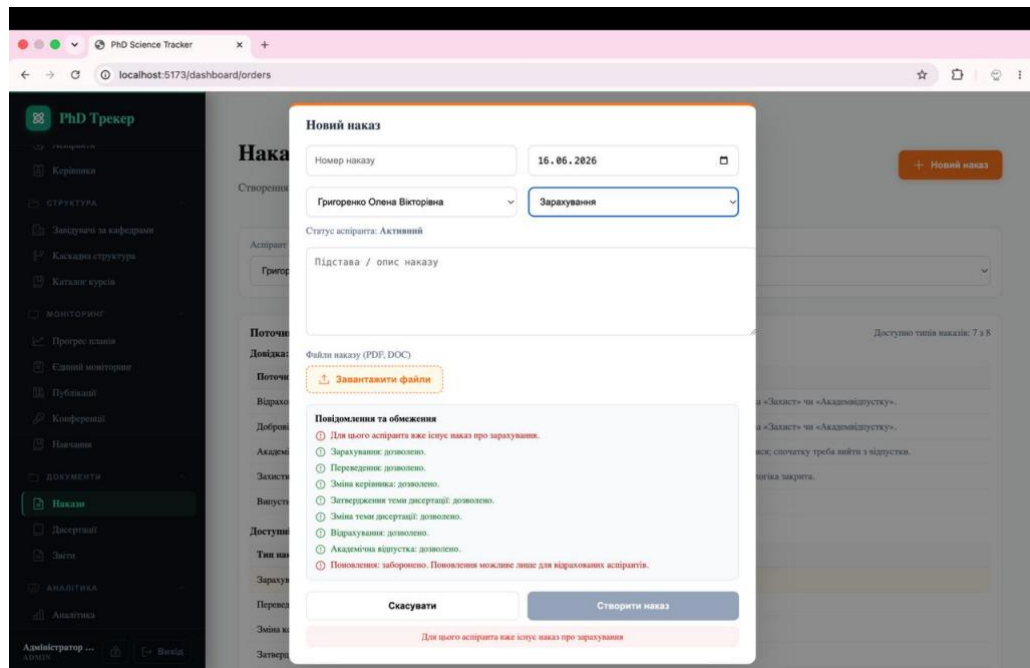


Рис. 3.12. Модальне вікно для створення наказів

Модуль управління дисертаціями (рис. 3.13) показує портфоліо дослідницьких робіт аспірантів у форматі карток з візуалізацією етапів підготовки, показників плагіату та статусів допуску до захисту, що дозволяє адміністрації комплексно контролювати весь процес написання наукової роботи.

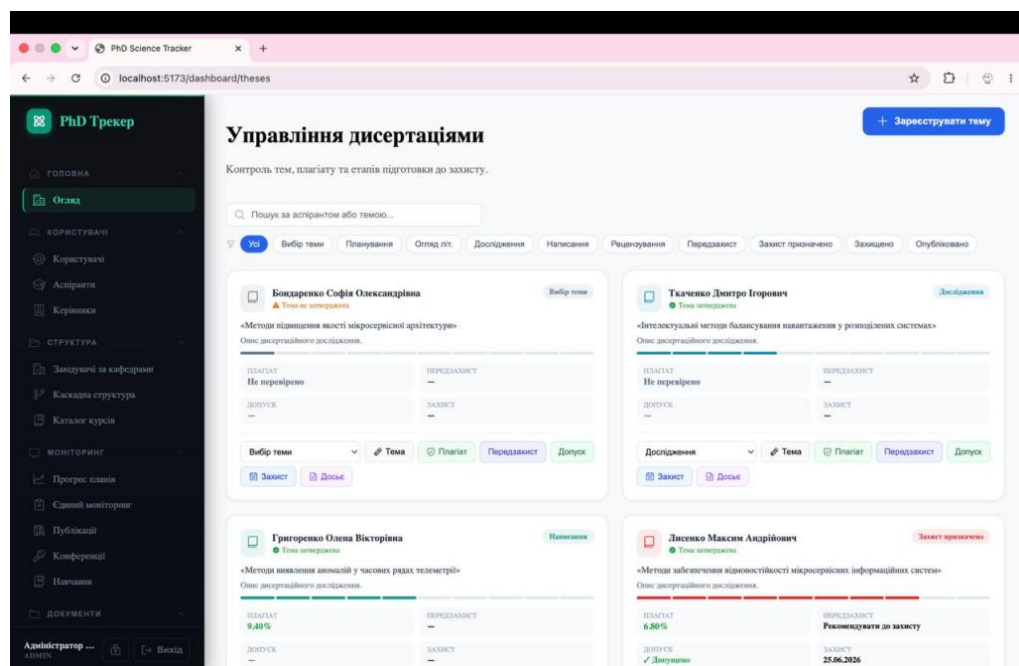
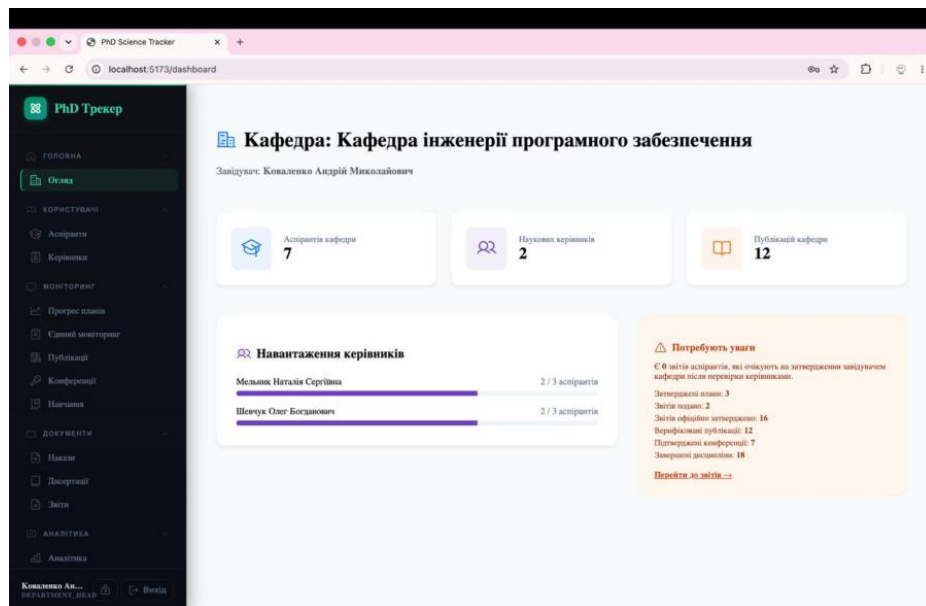


Рис. 3.13. Модуль управління дисертаціями

Дашборд кафедри (рис. 3.14) відображає загальну статистику конкретної кафедри, якою керує завідувач. На верхній панелі вмістяться ключові показники: кількість аспірантів, що навчаються на кафедрі, список наукових керівників та обсяг опублікованих матеріалів.



Семестрова атестація (рис. 3.15) реалізує реєстр семестрових звітів аспірантів його кафедри з фільтрацією за статусами затвердження.

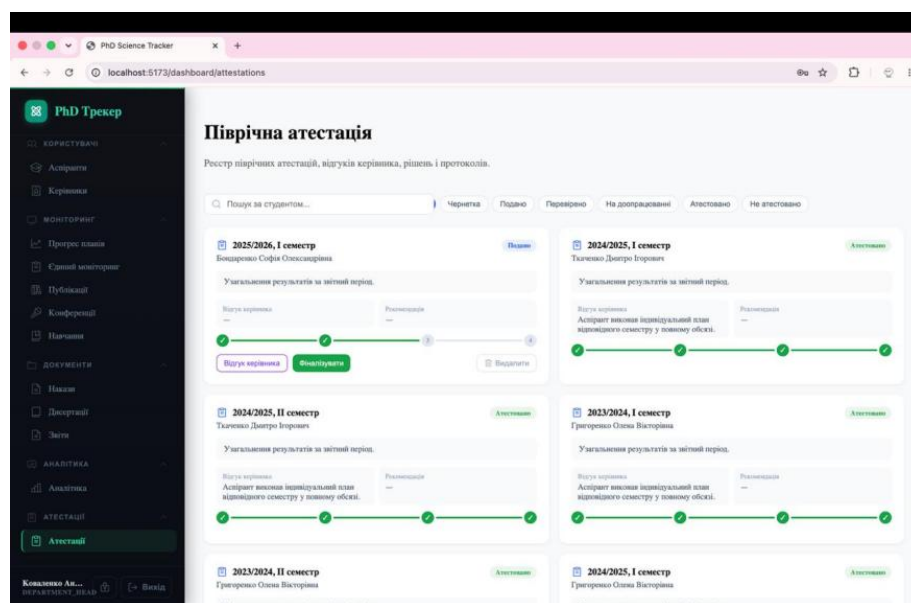


Рис. 3.15. Сторінка для відслідковування семестрових атестацій

Інтерфейс наукового керівника організований як персональна робоча панель для управління своїми аспірантами та контролю їхнього прогресу в межах навчально-наукової діяльності.

Кабінет наукового керівника (рис. 3.16) функціонує як інформаційний дашборд, що агрегує ключові дані про керівникову наукову групу.

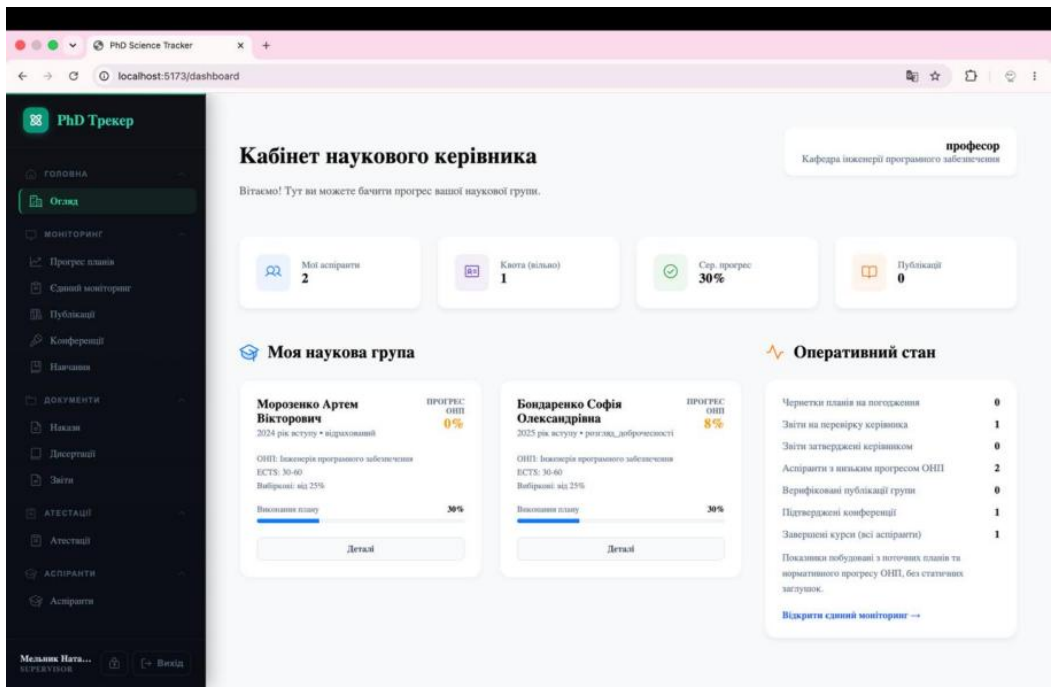
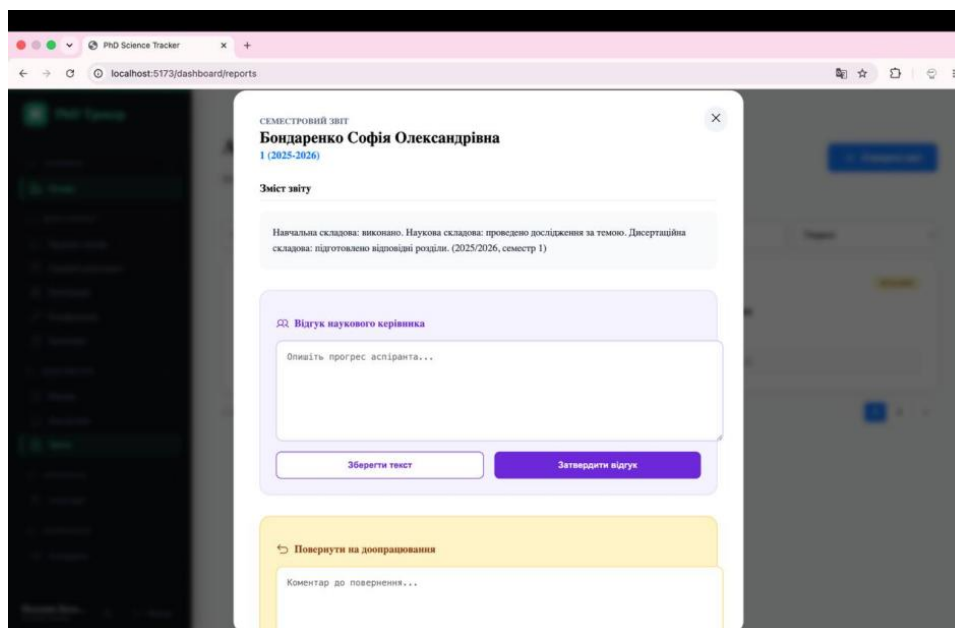


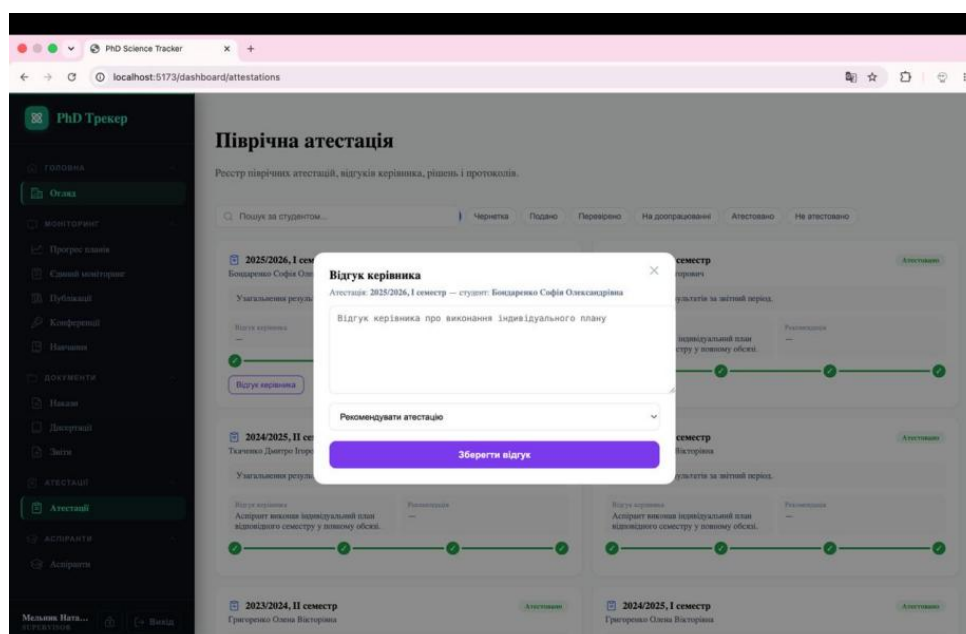
Рис. 3.16. Сторінка наукового керівника

На верхній панелі розташовані картки з базовими показниками: кількість аспірантів під керівництвом, кількість наукових публікацій, загальний відсоток прогресу та обсяг опублікованих матеріалів.

Перегляд семестрового звіту (рис. 3.17) відкривається модальним вікном, де керівник може ознайомитися з повним змістом звіту аспіранта та залишити фахове оцінювання. Інтерфейс розділено на дві частини: вміст звіту та панель «Відгук наукового керівника» з полем для внесення письмових рекомендацій. Керівник також має можливість відхилити звіт із коментарем на допрацювання через окремий блок «Повернути на допрацювання».



Семестрової атестація (рис. 3.18) представляє реєстр семестрових звітів всіх керівникових аспірантів з фільтрацією за статусами. На цій сторінці керівник відслідковує статус кожного звіту та може залишати коментарі до процесу затвердження.



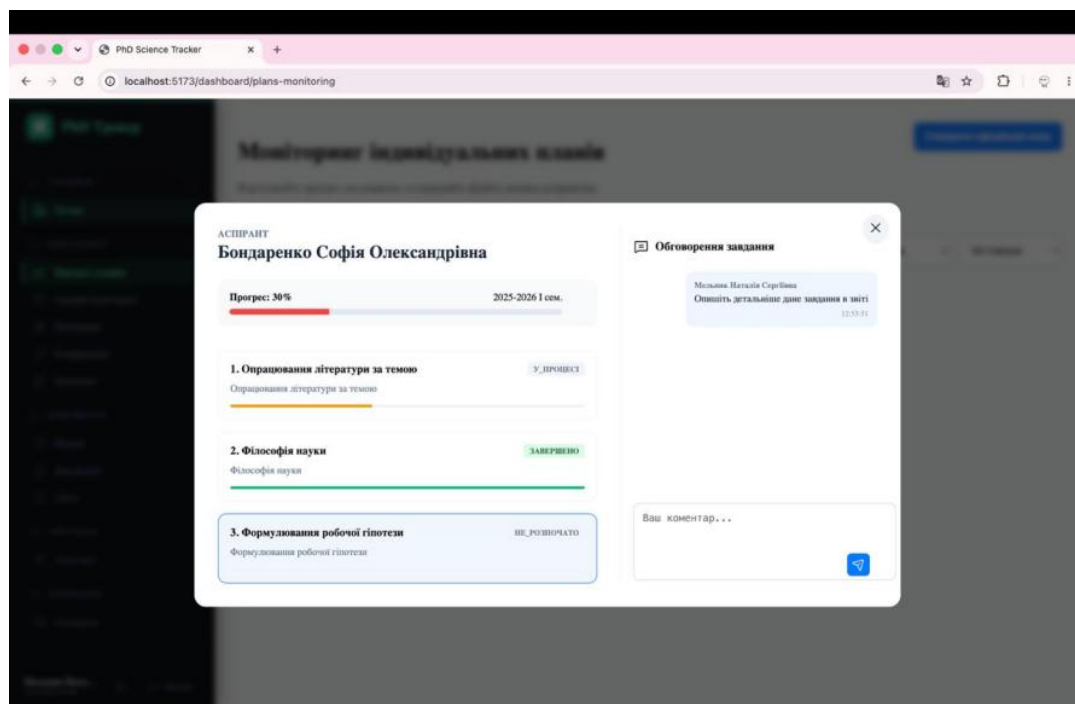
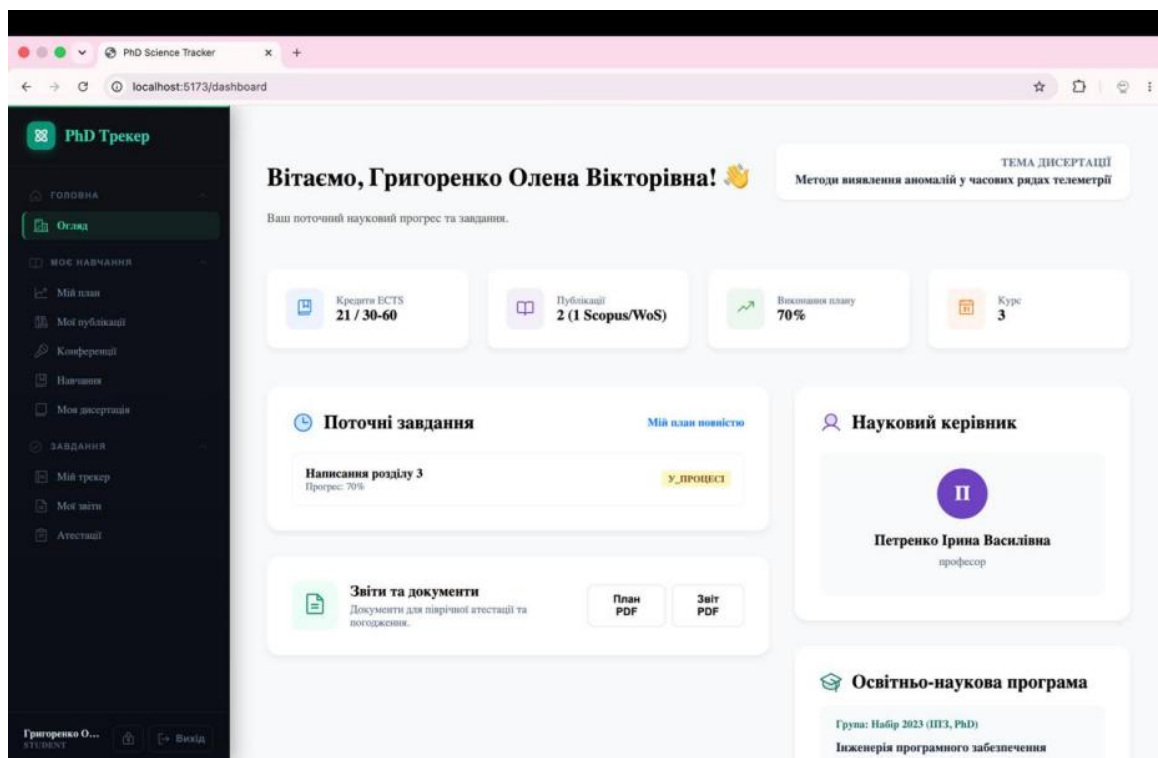


Рис. 3.19. Перегляд індивідуального плану

Інтерфейс аспіранта організований як персональна робоча панель для управління власною навчально-науковою діяльністю та контролю прогресу підготовки до захисту дисертації.

Головна сторінка системи (рис. 3.20) забезпечує аспіранта зведеною інформацією щодо поточного прогресу виконання освітньо-наукової програми. Інтерфейс містить віджети з даними про кількість здобутих кредитів ECTS, публікації, загальний відсоток виконання індивідуального плану, а також відображає пріоритетні поточні завдання та дані про наукового керівника.



Сторінка індивідуального плану (рис. 3.21) дозволяє користувачу переглядати офіційний графік підготовки, структурований за роками та семестрами. Здобувач може відслідковувати актуальні статуси виконання освітньої та наукової складових, а також завантажувати сформований документ у форматах PDF та DOCX для подальшого використання.

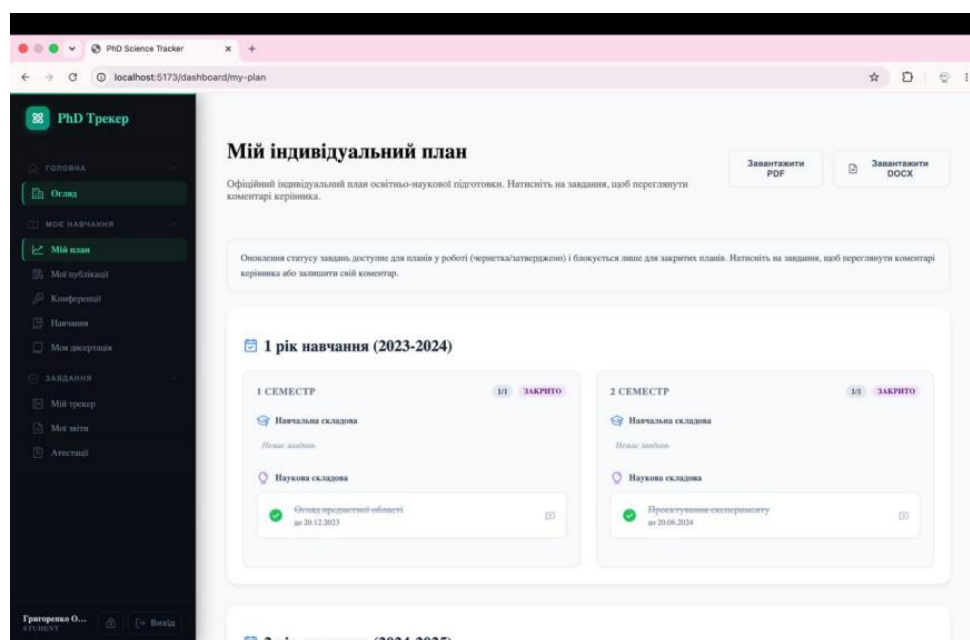


Рис. 3.21. Перегляд індивідуального плану

					ИПЗ.КР.Б-121-26-ПЗ	Арк.
						75

Розділ публікацій (рис. 3.22) надає інструменти для ведення наукового доробку здобувача. Інтерфейс дозволяє додавати нові статті чи монографії, вказувати індексацію в наукометричних базах (Scopus, Web of Science) та містить швидкі посилання на цифрові ідентифікатори об'єктів (DOI) для швидкої верифікації даних.

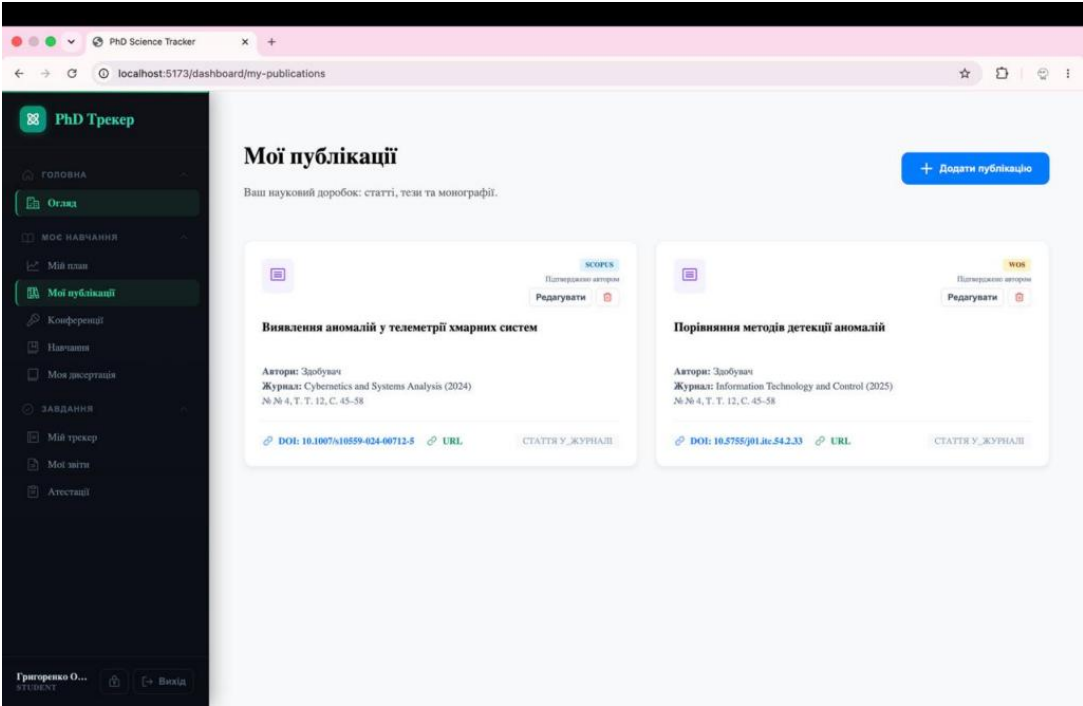


Рис. 3.22. Облік наукових публікацій

Сторінка обліку конференцій (рис. 3.23) призначена для документування участі аспіранта у наукових подіях та апробації результатів. Форма відображає рівень заходу, тему доповіді, дати та локацію проведення, а також роль здобувача під час виступу.

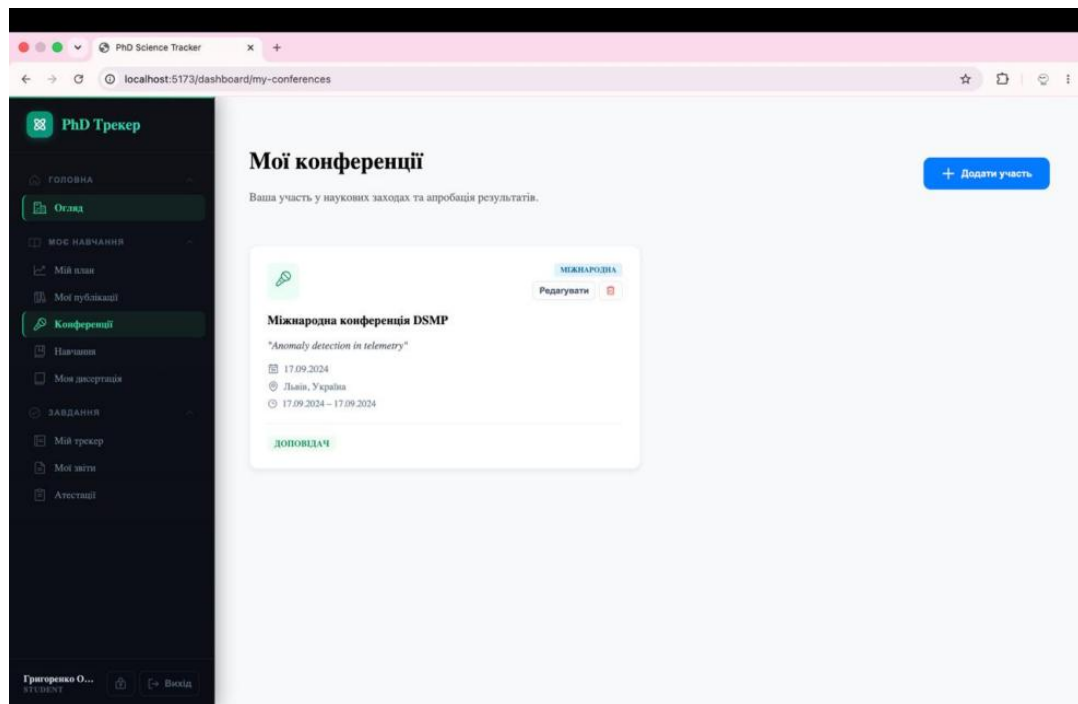


Рис. 3.23. Фіксація участі у наукових заходах

Вкладка навчальної складової (рис. 3.24) відображає прогрес опанування дисциплін у кредитах ECTS. Здобувач має змогу переглядати отримані оцінки за нормативні предмети, аналізувати співвідношення обов'язкових та вибірових компонентів, а також здійснювати запис на нові вибірові дисципліни з доступного каталогу.

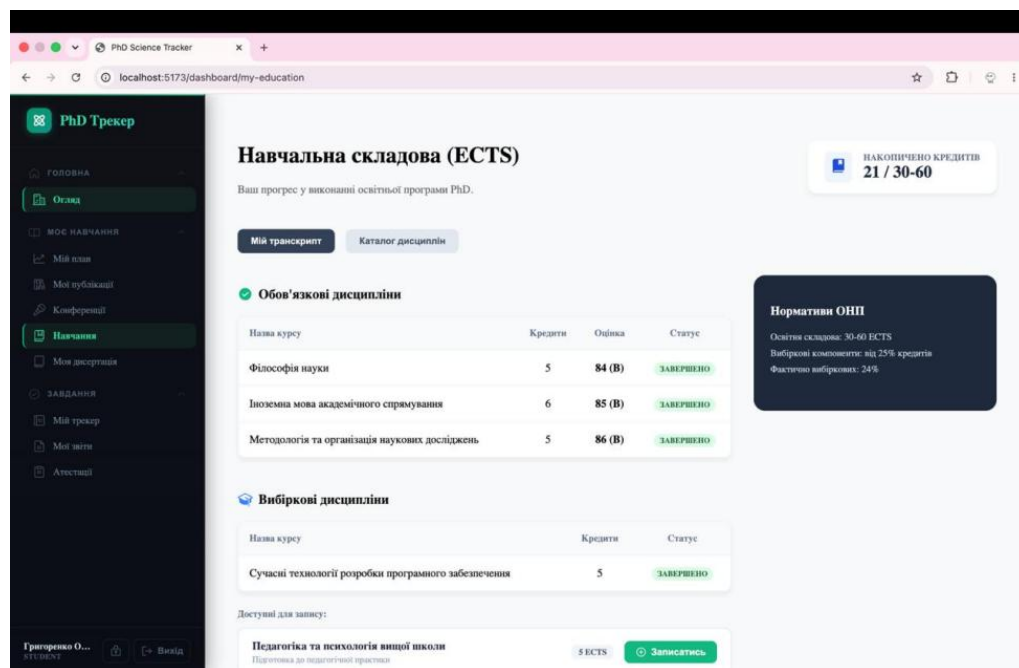


Рис. 3.24. Моніторинг навчальної складової

Сторінка дисертації (рис. 3.25) акумулює ключові дані про дослідження, включаючи затверджену тему та відслідковування пройдених етапів підготовки. Інтерфейс дозволяє завантажувати повний текст роботи та автореферат для формування досьє, а також відображає статус перевірки на академічну доброчесність.

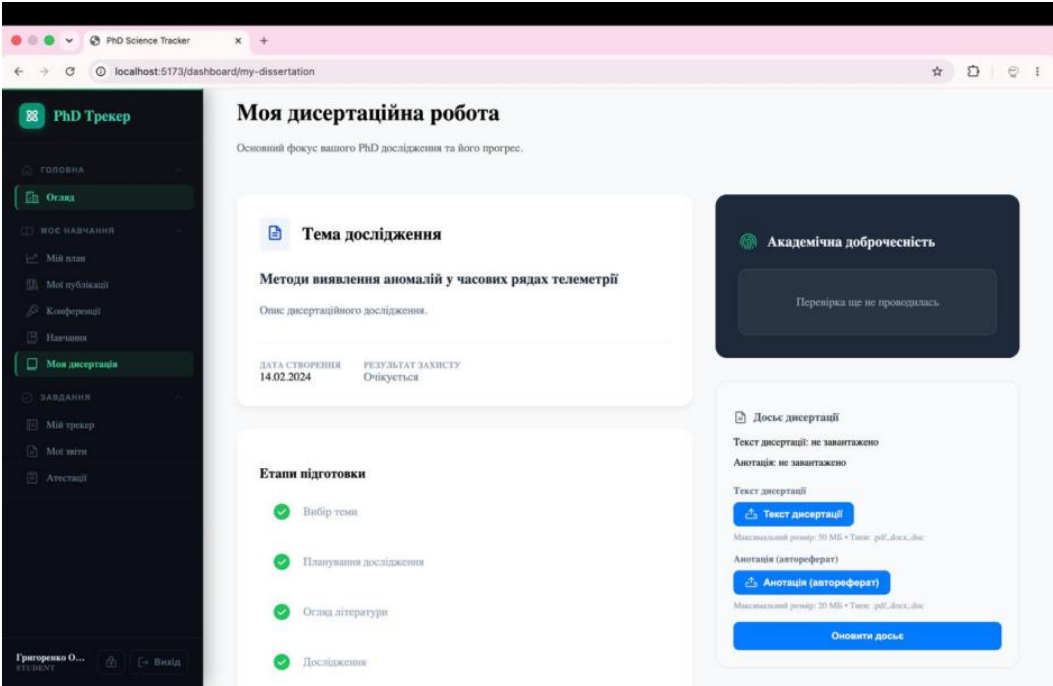
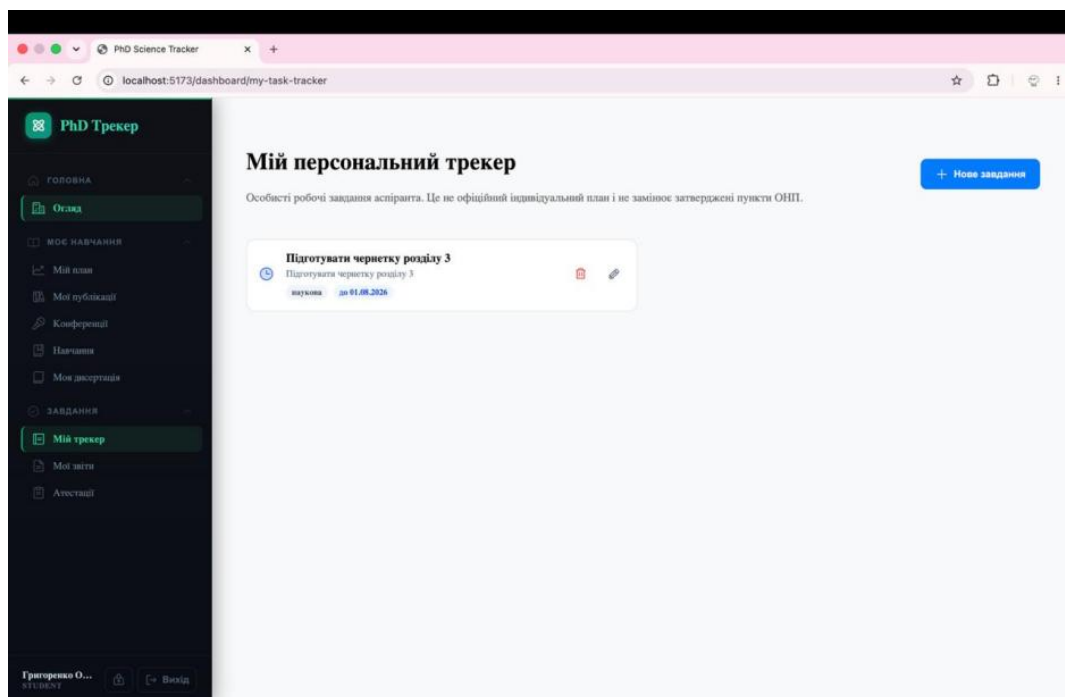


Рис. 3.25. Управління дисертаційною роботою

Персональний трекер (рис. 3.26) призначений для управління особистим робочим часом аспіранта та мікроменеджменту. Користувач може створювати нові поточні задачі, встановлювати дедлайни та відстежувати статуси виконання проміжних робіт, які не деталізуються в офіційному плані.



Модальне вікно звітності (рис. 3.27) забезпечує процес подачі звітів та комунікації з науковим керівником. У разі повернення звіту зі статусом «На доопрацюванні», здобувач бачить зауваження керівника, має можливість редагувати текстовий опис виконання плану та повторно подати документ на розгляд.

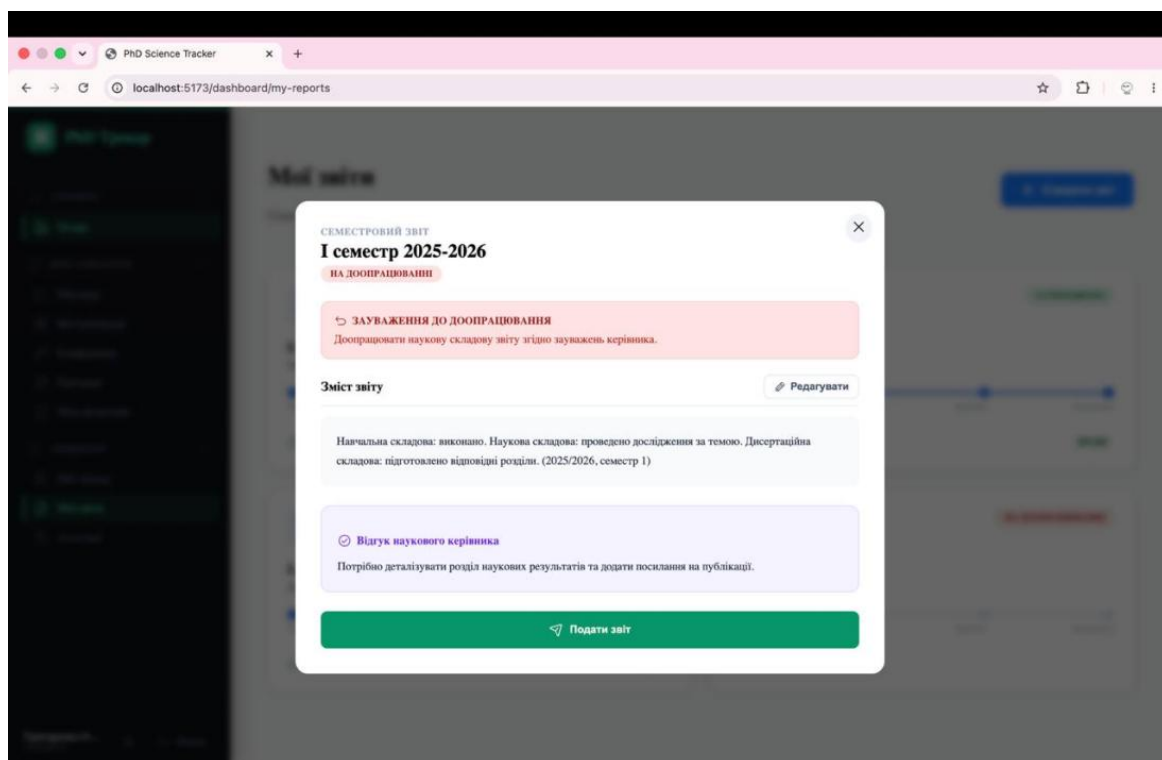


Рис. 3.27. Робота з семестровим звітом.

3.3. Тестування роботи програмного комплексу

Тестування програмного комплексу PhD Science Tracker проводилося з метою перевірки відповідності реалізованої системи поставленим функціональним вимогам, оцінки коректності її роботи у різних режимах, а також для виявлення та усунення дефектів до моменту впровадження. Особливість системи полягає у наявності чотирьох ролей з різними правами доступу й складної бізнес-логіки каскадного затвердження документів, тому основну увагу під час тестування було приділено перевірці розмежування доступу, коректності переходів між статусами та цілісності даних.

Серверну частину було запущено у режимі розробки командою для відповідного середовища, клієнтську частину було запущено через сервер розробки Vite, а як сховище даних використовувалася окрема база PostgreSQL зі структурою, сформованою міграціями TypeORM.

Для забезпечення повноти перевірки застосовувався комбінований підхід, що поєднує кілька видів тестування. Функціональне тестування виконувалося вручну через інтерфейс системи й охоплювало кожен зі сценаріїв роботи користувача відповідно до його ролі. Тестування програмного інтерфейсу серверної частини проводилося окремо від клієнта, що дозволило перевірити серверну логіку у відриві від інтерфейсу і впевнитися, що правила доступу й валідація реалізовані саме на стороні сервера. Інтеграційне тестування перевіряло взаємодію між клієнтом, сервером і базою даних як єдиним цілим у наскрізних сценаріях.

Першим етапом стала перевірка підсистеми автентифікації та авторизації як основи безпеки всієї системи. Перевірялося, що вхід можливий лише з правильною парою логіна і пароля, а при введенні хибних облікових даних система повертає відповідне повідомлення й не надає доступу. Окрему увагу приділено механізму токенів: підтверджено, що після успішного входу клієнт отримує токен доступу і токен оновлення, що захищені маршрути недоступні без дійсного токена і що після завершення терміну дії токена доступу він коректно оновлюється за допомогою токена оновлення без

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						80

повторного введення пароля. Також було перевірено, що користувач кожної ролі бачить лише ті розділи й дані, які передбачені його рівнем доступу, а спроба звернутися до чужого ресурсу напряму через адресу запиту блокується на стороні сервера, а не лише прихована в інтерфейсі.

Наступним етапом стала перевірка функцій адміністратора як ролі з найширшими повноваженнями. Тестувалися створення, редагування та фільтрування облікових записів у модулі управління користувачами, призначення завідувачів у модулі кафедр, а також робота каскадної структури від кафедри через спеціальності й освітні програми до груп з можливістю редагування на кожному рівні. Перевірялося коректне відображення карток наукових керівників з індикаторами навантаження й правильність розподілу аспірантів між керівниками. Окремо тестувався модуль наказів, у якому система автоматично перевіряє поточний статус аспіранта й блокує формування суперечливих розпоряджень, наприклад спробу відрахувати вже відрахованого здобувача. Перевірка підтвердила, що такі дії перехоплюються до збереження й супроводжуються зрозумілим повідомленням про причину блокування.

Окремий і найважливіший блок тестування стосувався процесу каскадного затвердження документів, що є ключовою бізнес-логікою системи. Перевірялося проходження індивідуального плану аспіранта повним маршрутом, а саме формування плану, затвердження науковим керівником і подальше погодження завідувачем кафедри, з контролем того, що на кожному кроці змінюється статус документа і що наступний учасник процесу отримує доступ до документа лише після дії попереднього. Аналогічно тестувався життєвий цикл семестрового звіту через усі передбачені статуси. Окремо перевірявся сценарій повернення звіту на допрацювання, коли керівник залишає коментар, аспірант бачить зауваження, вносить правки в текстовий опис виконання плану й повторно подає документ на розгляд. Результати показали, що переходи між статусами відбуваються коректно й що система не дозволяє пропустити обов'язкові етапи погодження або змінити статус

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						81

документа в обхід встановленого маршруту.

Перевірялися також функції обліку науково-освітньої діяльності аспіранта. Тестувалося додавання публікацій із зазначенням індексації у наукометричних базах Scopus і Web of Science та посилань на цифрові ідентифікатори об'єктів, фіксація участі у конференціях із зазначенням рівня заходу й ролі доповідача, запис на вибіркові дисципліни з доступного каталогу та облік отриманих кредитів ECTS. Окремо було перевірено формування й завантаження індивідуального плану у форматах PDF і DOCX, коректність відображення показників прогресу й доброчесності у модулі управління дисертаціями, а також роботу персонального трекера завдань аспіранта.

Поряд із перевіркою основних сценаріїв проводилося негативне тестування й перевірка валідації вхідних даних. Перевірялася поведінка форм при незаповнених обов'язкових полях, введенні значень некоректного формату та перевищенні допустимих меж, зокрема при введенні відсотка плагіату поза межами діапазону від нуля до ста. Підтверджено, що серверна частина не покладається лише на перевірки у браузері й самостійно відхиляє некоректні дані, повертаючи зрозумілі повідомлення про помилку без аварійного завершення роботи.

Узагальнені результати ключових тестових сценаріїв наведено в таблиці 3.1.

Завершальним етапом стала перевірка сумісності та зручності використання інтерфейсу. Клієнтську частину було протестовано у кількох сучасних браузерах для підтвердження однакового відображення й працездатності елементів, а також перевірено поведінку інтерфейсу при різних розмірах вікна. Перевірялося, що бічне навігаційне меню залишається доступним на всіх сторінках і що перехід між модулями не призводить до втрати контексту поточної роботи.

Таблиця 3.1

Ключові тестові сценарії

Тестовий сценарій	Очікуваний результат	Статус
Вхід з правильними обліковими даними	Надання доступу, видача токенів	Пройдено
Вхід з хибним паролем	Відмова в доступі, повідомлення про помилку	Пройдено
Створення та редагування користувача адміністратором	Збереження змін у базі даних	Пройдено
Каскадне затвердження індивідуального плану	Послідовна зміна статусів за маршрутом	Пройдено
Повний цикл семестрового звіту зі статусами	Коректні переходи між усіма статусами	Пройдено
Повернення звіту на допрацювання	Відображення коментаря, повторне подання	Пройдено
Формування суперечливого наказу	Блокування дії з поясненням	Пройдено
Введення некоректних даних у форму	Відхилення сервером, повідомлення про помилку	Пройдено
Завантаження плану у форматах PDF і DOCX	Коректне формування документів	Пройдено

Висновки до третього розділу

У третьому розділі описано практичну реалізацію вебсистеми керування процесами аспірантури PhD Science Tracker, порядок її встановлення, структуру інтерфейсу та результати тестування.

У підрозділі 3.1 викладено порядок встановлення й налаштування системи, що складається із серверної частини на базі NestJS і клієнтської частини на базі React зі збіркою інструментом Vite. Описано вимоги до програмного середовища, послідовність встановлення залежностей, створення й налаштування бази даних PostgreSQL, конфігурування через файл змінних оточення, виконання міграцій TypeORM і заповнення бази початковими даними, а також питання безпеки.

У підрозділі 3.2 розглянуто структуру інтерфейсу з розподілом можливостей між чотирма ролями, а саме адміністратором, завідувачем кафедри, науковим керівником та аспірантом. Показано, що кожна роль отримує спеціалізований робочий простір з доступом лише до відповідних повноважень, а єдине навігаційне меню забезпечує зручний перехід між

модулями.

У підрозділі 3.3 наведено результати тестування, що проводилося із застосуванням функціонального, інтеграційного та негативного тестування й тестування програмного інтерфейсу. Перевірку зосереджено на найбільш критичних аспектах, а саме розмежуванні доступу, коректності каскадного затвердження документів і цілісності даних. Тестування підтвердило стабільну роботу системи за всіма передбаченими сценаріями та її стійкість до некоректних дій користувача.

Таким чином, розроблена система є завершеним продуктом, який реалізує поставлені функціональні вимоги й готовий до впровадження в навчально-науковий процес для автоматизації управління підготовкою аспірантів.

ВИСНОВКИ

Вирішено актуальне завдання автоматизації процесів обліку та моніторингу наукової діяльності аспірантів кафедри шляхом проєктування та програмної реалізації веб-орієнтованої інформаційної системи PhD Science Tracker. Поставлену мету досягнуто, а всі визначені у вступі завдання виконано в повному обсязі.

У результаті аналізу предметної галузі встановлено, що традиційні засоби ведення аспірантської звітності, а саме паперові журнали, електронні таблиці та листування в месенджерах, не відповідають сучасним вимогам до підготовки аспірантів й призводять до фрагментарності даних, дублювання інформації та втрати контролю над її актуальністю. Визначено чотири категорії користувачів системи з різними рівнями доступу, а саме аспіранта, наукового керівника, завідувача кафедри та адміністратора, і сформовано вичерпний перелік функціональних та нефункціональних вимог. Проведений порівняльний аналіз наявних рішень, серед яких системи управління навчанням, універсальні трекери завдань і хмарні офісні пакети, переконливо засвідчив, що жодне з них не задовольняє комплексу вимог до супроводу підготовки здобувачів за українськими нормативними стандартами, що обґрунтувало доцільність розробки спеціалізованого програмного продукту.

Для реалізації системи обрано й обґрунтовано архітектурні та технологічні рішення. Прийнято модульну монолітну архітектуру з виділеним клієнтським односторінковим застосунком, що взаємодіє з серверним ядром через REST API, а серверну частину побудовано за принципами Чистої Архітектури з розмежуванням коду на доменний, застосунковий та інфраструктурний шари. Технологічний стек, що включає мову TypeScript, фреймворк NestJS, бібліотеку React, систему управління базами даних PostgreSQL та інструмент об'єктно-реляційного відображення TypeORM, дібрано як оптимальну сукупність зрілих і добре задокументованих технологій, що природно підтримують обрану архітектуру та забезпечують високі показники безпеки й надійності.

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						85

На етапі проєктування побудовано повну функціональну модель системи у вигляді діаграм варіантів використання для кожної з чотирьох ролей та зведеної таблиці сценаріїв. Спроектовано структуру реляційної бази даних із двадцяти восьми таблиць з опертям на третю нормальну форму та частковою денормалізацією через тип JSONB для напівструктурованих даних, визначено зв'язки між таблицями й каскадні правила видалення. Розроблено п'ять ключових алгоритмів роботи системи, а саме автентифікацію на основі двох токенів, скінченні автомати статусів звіту та академічного статусу аспіранта, транзакційну видачу кадрового наказу й агрегацію аналітичних даних для дашбордів, які разом утворюють основу бізнес-логіки платформи.

Здійснено програмну реалізацію системи з розподілом серверної частини на вісімнадцять предметно-орієнтованих модулів і клієнтської частини на чотири рольові групи маршрутів. Реалізовано багаторівневий захист доступу, що включає перевірку токенів автентифікації, контроль ролей та захист від доступу до чужих ресурсів, а також незмінний журнал аудиту всіх операцій зі зміни даних. Описано порядок встановлення й налаштування системи та структуру інтерфейсу для кожної з ролей, що підтверджує завершеність і працездатність продукту.

Проведене тестування програмного комплексу із застосуванням функціонального, інтеграційного та негативного тестування й тестування програмного інтерфейсу підтвердило стабільну роботу системи за всіма передбаченими сценаріями, коректне розмежування доступу, надійну роботу механізму каскадного затвердження документів і стійкість до некоректних дій користувача. Виявлені у процесі роботи недоліки усунено, що засвідчило повторне тестування.

Практичне значення роботи полягає у тому, що розроблена система може бути впроваджена на кафедрах закладів вищої освіти для автоматизації обліку наукової діяльності аспірантів, замінюючи ручне ведення таблиць і паперових звітів, що сприяє оптимізації документообігу, мінімізації помилок при атестації та підвищенню якості управління підготовкою наукових кадрів.

Перспективними напрямками подальшого розвитку системи є інтеграція із зовнішніми наукометричними базами для автоматичного підтягування метаданих публікацій, розширення аналітичних можливостей засобами візуалізації даних, розробка мобільного застосунку на основі наявного API, а також додавання модуля автоматичної генерації офіційних документів та звітності для університетських інформаційних систем.

Таким чином, у кваліфікаційній роботі досягнуто поставленої мети та повністю виконано визначені завдання, а розроблена інформаційна система PhD Science Tracker є продуктом, що реалізує поставлені функціональні вимоги й готовий до впровадження в навчально-науковий процес..

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Закон України «Про вищу освіту» від 01.07.2014 р. № 1556-VII. Дата оновлення: 24.04.2024. URL: <https://zakon.rada.gov.ua/laws/show/1556-18> (дата звернення: 15.06.2026).
2. Про опублікування результатів дисертацій на здобуття наукових ступенів доктора і кандидата наук : наказ Міністерства освіти і науки України від 23.09.2019 р. № 1220. URL: <https://zakon.rada.gov.ua/laws/show/z1086-19> (дата звернення: 15.06.2026).
3. Moodle: Open-source learning platform. Moodle. URL: <https://moodle.org> (дата звернення: 15.06.2026).
4. Notion – Your connected workspace for wiki, docs & projects. Notion. URL: <https://www.notion.so> (дата звернення: 15.06.2026).
5. Морзе Н. В., Гладун М. А. Використання хмарних технологій Google Workspace в освітньому процесі університету. Відкрите освітнє е-середовище сучасного університету. 2022. № 12. С. 45–58.
6. Newman S. Building Microservices: Designing Fine-Grained Systems. 2nd ed. O'Reilly Media, 2021. 612 p.
7. Sommerville I. Software Engineering. 10th ed. Pearson, 2015. 816 p.
8. Fowler M. MonolithFirst. MartinFowler.com. 2015. URL: <https://martinfowler.com/bliki/MonolithFirst.html> (дата звернення: 15.06.2026).
9. Fowler M., Lewis J. Microservices: a definition of this new architectural term. MartinFowler.com. 2014. URL: <https://martinfowler.com/articles/microservices.html> (дата звернення: 15.06.2026).
10. Kurose J. F., Ross K. W. Computer Networking: A Top-Down Approach. 8th ed. Pearson, 2021. 800 p.
11. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : Ph.D. dissertation. University of California, Irvine, 2000. 162 p. URL: <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm> (дата звернення: 15.06.2026).

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						88

12. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall, 2017. 432 p.
13. Richards M., Ford N. Fundamentals of Software Architecture: An Engineering Approach. O'Reilly Media, 2020. 416 p.
14. Vernon V. Implementing Domain-Driven Design. Addison-Wesley Professional, 2013. 656 p.
15. NestJS: A progressive Node.js framework. NestJS. URL: <https://nestjs.com> (дата звернення: 15.06.2026).
16. React – The library for web and native user interfaces. Meta Platforms. URL: <https://react.dev> (дата звернення: 15.06.2026).
17. TanStack Query: Powerful asynchronous state management. TanStack. URL: <https://tanstack.com/query/latest> (дата звернення: 15.06.2026)..
18. PostgreSQL: The World's Most Advanced Open Source Relational Database. PostgreSQL Global Development Group. URL: <https://www.postgresql.org> (дата звернення: 15.06.2026).
19. TypeORM – ORM for TypeScript and JavaScript. URL: <https://typeorm.io> (дата звернення: 15.06.2026).
20. Unified Modeling Language (UML) Specification. Version 2.5.1. Object Management Group (OMG). 2017. URL: <https://www.omg.org/spec/UML> (дата звернення: 15.06.2026).
21. Tanenbaum A. S., Wetherall D. J. Computer Networks. 5th ed. Pearson, 2010. 960 p.
22. Cormen T. H., Leiserson C. E., Rivest R. L., Stein C. Introduction to Algorithms. 4th ed. MIT Press, 2022. 1312 p.
23. Jones M., Bradley J., Sakimura N. JSON Web Token (JWT). RFC 7519. Internet Engineering Task Force (IETF), 2015. URL: <https://datatracker.ietf.org/doc/html/rfc7519> (дата звернення: 15.06.2026).
24. Provos N., Mazières D. A Future-Adaptable Password Scheme. Proceedings of the 1999 USENIX Annual Technical Conference (ATC). Monterey, CA, USA, June 6–11, 1999. P. 81–92. URL:

https://www.usenix.org/legacy/events/usenix99/provos/provos_html (дата
звернення: 15.06.2026).

25. Hopcroft J. E., Motwani R., Ullman J. D. Introduction to Automata Theory, Languages, and Computation. 3rd ed. Pearson, 2006. 560 p.

26. TanStack Query v5 Documentation. TanStack. URL: <https://tanstack.com/query/latest> (дата звернення: 15.06.2026).

27. NestJS 10 Documentation. NestJS. URL: <https://docs.nestjs.com> (дата звернення: 15.06.2026).

28. TypeScript: Strict Type-Checking Options. Microsoft. URL: https://www.typescriptlang.org/tsconfig/#Strict_Type_Checking_Options (дата звернення: 15.06.2026).

29. React v18.0 Release. Meta Platforms. 2022. URL: <https://react.dev/blog/2022/03/29/react-v18> (дата звернення: 15.06.2026).

30. Madden N. API Security in Action. Manning Publications, 2020. 520 p.

ДОДАТКИ

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						91

Лістинг програми app.module.ts

```

import { Module } from '@nestjs/common';
import { ConfigModule } from '@nestjs/config';
import { EventEmitterModule } from '@nestjs/event-emitter';
import { ScheduleModule } from '@nestjs/schedule';
import { ThrottlerGuard, ThrottlerModule } from '@nestjs/throttler';
import { APP_GUARD, APP_INTERCEPTOR } from '@nestjs/core';
import { JwtAuthGuard, OwnershipCheckInterceptor, RolesGuard, } from
'./infrastructure/auth';
import { AuditInterceptor } from './infrastructure/audit';
import { NotificationsModule } from './application/notifications';
import { UserModule } from './user.module';
import { AuditLogModule } from './audit-log.module';
import { SupervisorModule } from './supervisor.module';
import { StudentModule } from './student.module';
import { IndividualPlanModule } from './individual-plan.module';
import { ReportModule } from './report.module';
import { PublicationModule } from './publication.module';
import { ConferenceModule } from './conference.module';
import { DissertationModule } from './dissertation.module';
import { DepartmentModule } from './department.module';
import { EducationModule } from './education.module';
import { DocumentModule } from './document.module';
import { OrdersModule } from './orders.module';
import { FileModule } from './file.module';
import { PersonalTaskModule } from './personal-task.module';
import { AttestationModule } from './attestation.module';
import { CohortModule } from './cohort.module';

```

```

import { ProgramModule } from './program.module';
import { AnnouncementModule } from './announcement.module';
import { SystemSettingsModule } from './system-settings.module';
import { DatabaseConfig } from './config/database.config';
import { OWNERSHIP_STRATEGIES } from './domain/shared/ports/ownership-
strategy.interface';
import { ReportOwnershipStrategy } from './infrastructure/report/report-
ownership.strategy';
import { StudentOwnershipStrategy } from './infrastructure/student-
adapters/student-ownership.strategy';
import { DissertationOwnershipStrategy } from './infrastructure/dissertation-
adapters/dissertation-ownership.strategy';
import {
    IndividualPlanOwnershipStrategy
} from './infrastructure/individual-plan-adapters/individual-plan-ownership.strategy';
import { PublicationOwnershipStrategy } from './infrastructure/publication-
adapters/publication-ownership.strategy';

@Module({
    imports: [
        ConfigModule.forRoot({
            isGlobal: true,
            envFilePath: '.env',
        }),
        ThrottlerModule.forRoot([
            {
                name: 'default',
                ttl: 60000, // 1 minute
                limit: 100, // 100 requests per minute globally
            },

```

```

    ],
    DatabaseConfig,
    EventEmitterModule.forRoot(),
    ScheduleModule.forRoot(),
    UserModule,
    AuditLogModule,
    SupervisorModule,
    StudentModule,
    IndividualPlanModule,
    ReportModule,
    PublicationModule,
    ConferenceModule,
    DissertationModule,
    DepartmentModule,
    EducationModule,
    DocumentModule,
    OrdersModule,
    FileModule,
    PersonalTaskModule,
    AttestationModule,
    CohortModule,
    ProgramModule
  ],
  providers: [
    {
      provide: APP_GUARD,
      useClass: ThrottlerGuard,
    },
    {
      provide: APP_GUARD,

```

```

    useClass: JwtAuthGuard,
  },
  {
    provide: APP_GUARD,
    useClass: RolesGuard,
  },
  {
    provide: APP_INTERCEPTOR,
    useClass: AuditInterceptor,
  },
  {
    provide: APP_INTERCEPTOR,
    useClass: OwnershipCheckInterceptor,
  },
  {
    provide: OWNERSHIP_STRATEGIES,
    useFactory: (
      report: ReportOwnershipStrategy,
      student: StudentOwnershipStrategy,
      dissertation: DissertationOwnershipStrategy,
      individualPlan: IndividualPlanOwnershipStrategy,
      publication: PublicationOwnershipStrategy,
    ) => [report, student, dissertation, individualPlan, publication],
    inject: [
      ReportOwnershipStrategy,
      StudentOwnershipStrategy,
      DissertationOwnershipStrategy,
      IndividualPlanOwnershipStrategy,
      PublicationOwnershipStrategy,
    ],
  },

```

```
    },  
  ],  
  })  
  export class AppModule {}
```