

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:

**«Система управління академічними ресурсами  
студентів: інтеграція та взаємодія компонентів»**

Виконала студентка групи ІІЗ-21-4  
Горай Софія Віталіївна

Керівник роботи:  
ЄФРЕМОВ Юрій Миколайович

Рецензент:  
ТОЛСТОЙ Ігор Анатолійович

Житомир – 2025

**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

**«26» лютого 2025 р.**

**ЗАВДАННЯ**  
**на кваліфікаційну роботу**

Здобувач вищої освіти: **Горай Софія Віталіївна**

Керівник роботи: **Єфремов Юрій Миколайович**

Тема роботи: **«Система управління академічними ресурсами студентів: інтеграція та взаємодія компонентів»**,

затверджена наказом закладу вищої освіти №61/с від «14» лютого 2025р.

Вихідні дані для роботи: **об'єктом дослідження** є організаційні та технічні процеси надання доступу до академічних ресурсів у вищих навчальних закладах за допомогою інформаційно-цифрових систем, а **предметом дослідження** є архітектура клієнтської та серверної частин програмного комплексу для управління академічними ресурсами  
Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

| <b>Розділ</b> | <b>Консультант</b> | <b>Завдання видав</b> | <b>Завдання прийняв</b> |
|---------------|--------------------|-----------------------|-------------------------|
| 1             | Єфремов Ю. М.      | 27.02.2025р.          | 27.02.2025р.            |
| 2             | Єфремов Ю. М.      | 27.02.2025р.          | 27.02.2025р.            |
| 3             | Єфремов Ю. М.      | 27.02.2025р.          | 27.02.2025р.            |

## РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з програмного комплексу та пояснювальної записки. Пояснювальна записка містить 89 сторінок, включає 54 ілюстрації, 20 таблиць, 3 додатки та 31 використане літературне джерело.

Метою роботи є розробка алгоритмів функціонування та реалізація програмного комплексу системи управління академічними ресурсами студентів.

Для досягнення поставленої мети у роботі здійснено аналіз предметної області, визначено основні вимоги до інформаційної системи, спроектовано архітектуру програмного комплексу та об'єктно-орієнтовану модель даних.

В процесі розробки створено інтерфейс користувача, реалізовано основні функціональні модулі системи: управління користувачами, курсами, модулями, навчальними матеріалами, завданнями та оцінюванням. Особливу увагу приділено забезпеченню різних рівнів доступу до системи, побудові масштабованої структури бази даних.

В результаті виконання роботи було створено дієвий інструмент для адміністрування академічних ресурсів, який дозволяє ефективно організовувати навчальний процес, контролювати досягнення студентів, автоматизувати роботу викладачів та адміністраторів.

Ключові слова: СИСТЕМА УПРАВЛІННЯ, АКАДЕМІЧНІ РЕСУРСИ, СТУДЕНТИ, ІНФОРМАЦІЙНА СИСТЕМА, БАЗА ДАНИХ, PYTHON, FASTAPI, ORM, SQLALCHEMY, АВТОРИЗАЦІЯ, ОСВІТНІЙ ПРОЦЕС.

|           |      |               |        |      |                                                                                                                   |                                            |      |         |  |
|-----------|------|---------------|--------|------|-------------------------------------------------------------------------------------------------------------------|--------------------------------------------|------|---------|--|
|           |      |               |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ                                                                                          |                                            |      |         |  |
|           |      |               |        |      |                                                                                                                   |                                            |      |         |  |
| Змн.      | Арк. | № докум.      | Підпис | Дата |                                                                                                                   |                                            |      |         |  |
| Розроб.   |      | Горай С. В.   |        |      | Система управління академічними ресурсами студентів : інтеграція та взаємодія компонентів<br>Пояснювальна записка | Лім.                                       | Арк. | Аркушів |  |
| Керівник  |      | Єфремов Ю. М. |        |      |                                                                                                                   |                                            | 3    | 102     |  |
|           |      |               |        |      |                                                                                                                   | Житомирська політехніка,<br>група ІПЗ-21-4 |      |         |  |
| Рецензент |      | Толстой І. А. |        |      |                                                                                                                   |                                            |      |         |  |
| Зав. каф. |      | Вакалюк Т. А. |        |      |                                                                                                                   |                                            |      |         |  |

## ABSTRACT

The bachelor's thesis consists of a software suite and an explanatory note. The explanatory note contains 89 pages, includes 54 illustrations, 20 tables, 3 appendices, and 31 referenced literary sources.

The aim of the thesis is to develop operational algorithms and implement a software system for student academic resource management.

To achieve this goal, the thesis analyzes the subject domain, defines the main requirements for the information system, designs the architecture of the software suite and the object-oriented data model.

During the development process, a user interface was created, and the main functional modules of the system were implemented: user management, courses, modules, learning materials, assignments, and assessment. Particular attention was paid to providing different access levels within the system and building a scalable database structure.

As a result, an effective tool for managing academic resources was created, allowing efficient organization of the educational process, monitoring of student achievements, and automation of work for teachers and administrators.

**Keywords:** MANAGEMENT SYSTEM, ACADEMIC RESOURCES, STUDENTS, INFORMATION SYSTEM, DATABASE, PYTHON, FASTAPI, ORM, SQLALCHEMY, AUTHENTICATION, EDUCATIONAL PROCESS.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 4    |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

## ЗМІСТ

|                                                                                                                                      |    |
|--------------------------------------------------------------------------------------------------------------------------------------|----|
| РЕФЕРАТ .....                                                                                                                        | 3  |
| ABSTRACT .....                                                                                                                       | 4  |
| ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ .....                                                                                                      | 6  |
| ВСТУП.....                                                                                                                           | 7  |
| РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ<br>ТЕХНОЛОГІЙ ДЛЯ СИСТЕМИ УПРАВЛІННЯ АКАДЕМІЧНИМИ<br>РЕСУРСАМИ СТУДЕНТІВ ..... | 9  |
| 1.1    Постановка задачі .....                                                                                                       | 9  |
| 1.2    Аналіз аналогів програмного продукту .....                                                                                    | 11 |
| 1.3    Вибір архітектури системи управління академічними ресурсами.....                                                              | 17 |
| 1.4    Обґрунтування вибору інструментальних засобів. ....                                                                           | 19 |
| 1.5    Розподіл ролей та принципи командної взаємодії .....                                                                          | 23 |
| Висновки до першого розділу.....                                                                                                     | 29 |
| РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ УПРАВЛІННЯ АКАДЕМІЧНИМИ<br>РЕСУРСАМИ СТУДЕНТІВ .....                                                  | 30 |
| 2.1    Специфікація вимог до програмного продукту .....                                                                              | 30 |
| 2.2    Визначення варіантів використання та об'єктно-орієнтованої структури                                                          | 35 |
| 2.3    Розробка бази даних системи .....                                                                                             | 41 |
| 2.4    Проектування та реалізація алгоритмів роботи системи .....                                                                    | 53 |
| Висновки до другого розділу .....                                                                                                    | 62 |
| РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ УПРАВЛІННЯ<br>АКАДЕМІЧНИМИ РЕСУРСАМИ СТУДЕНТІВ .....                                | 63 |
| 3.1    Порядок встановлення та налаштування параметрів системи.....                                                                  | 63 |
| 3.2    Структура інтерфейсу системи управління академічними ресурсами .....                                                          | 64 |
| 3.3    Тестування роботи програмного комплексу .....                                                                                 | 83 |
| Висновки до третього розділу.....                                                                                                    | 85 |
| ВИСНОВКИ .....                                                                                                                       | 86 |
| СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....                                                                                                      | 88 |
| ДОДАТКИ .....                                                                                                                        | 90 |

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface

БД – база даних

ІС – Інформаційна система

СУБД – Система управління базами даних

CRUD – Create, Read, Update, Delete

HTTP – Hypertext Transfer Protocol

ORM – Object-Relational Mapping

SQL – Structured Query Language

UUID – Universally Unique Identifier

ID – Identifier

PK – Primary Key

FK – Foreign Key

UI – User Interface

LMS – Learning Management System

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 6    |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

## ВСТУП

У сучасних умовах стрімкого розвитку інформаційних технологій та впровадження цифрових рішень у всі сфери діяльності освіта зазнає значних змін, що вимагають нових підходів до організації навчального процесу. Вищі навчальні заклади стикаються з необхідністю автоматизації адміністративних, навчальних та фінансових процесів, щоб забезпечити ефективну взаємодію між студентами, викладачами та адміністрацією. Одним із ключових інструментів для досягнення цієї мети є створення інтегрованої інформаційної системи, яка об'єднає різні аспекти навчального процесу в єдиному цифровому середовищі.

Останніми роками перехід на дистанційну освіту став ще більш актуальним через глобальні виклики, такі як пандемія, військові дії та необхідність забезпечення безперервного навчального процесу. Багато студентів перейшли на дистанційну або змішану форму навчання, що висунуло нові вимоги до цифрових платформ. Зокрема, зросла потреба у створенні гнучких та ефективних онлайн-сервісів, які дозволяють не лише отримувати освітні матеріали, але й вести електронний документообіг, контролювати академічну успішність, та оперативно взаємодіяти з викладачами та адміністрацією.

Студентський портал як інформаційна система стає невід'ємною частиною сучасного навчального закладу. Він забезпечує зручний доступ до навчальних матеріалів, дозволяє контролювати академічну успішність, спрощує документообіг, а також забезпечує оперативну комунікацію між усіма учасниками освітнього процесу.

**Актуальність** роботи зумовлена необхідністю впровадження сучасного освітнього порталу, який допоможе усунути проблеми, пов'язані з ручним управлінням освітнім процесом та обмеженим доступом до навчальних матеріалів. Додатково, впровадження таких функцій, як система контролю під час тестувань, таймлайн запланований робіт та автоматизоване створення документів, сприятиме персоналізації навчального процесу та підвищенню його ефективності.

**Мета** випускної кваліфікаційної роботи створення інтегрованої інформаційної системи для управління академічними ресурсами студентів.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 7    |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

Система спрямована на підвищення ефективності взаємодії студентів, викладачів та адміністрації навчального закладу через автоматизацію освітніх процесів.

Завданням випускної кваліфікаційної роботи є розробка інтегрованої інформаційної системи для управління академічними ресурсами студентів. Для цього необхідно проаналізувати предметну область, визначити вимоги до системи, спроектувати й реалізувати архітектуру веб-додатку, впровадити основні функції електронного документообігу та навчального модуля.

**Об'єктом дослідження** є процеси організації навчальної діяльності студентів, адміністрування освітніх програм.

**Предметом дослідження** виступає застосування сучасних технологій веб-розробки, засобів інтеграції з зовнішніми сервісами та інструментів забезпечення безпеки інформаційних систем для побудови єдиного порталу академічних ресурсів. У процесі роботи використано такі методи дослідження: системний аналіз предметної області, моделювання бізнес-процесів, об'єктно-орієнтоване проєктування, розробка та тестування веб-додатків, а також впровадження заходів інформаційної безпеки.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 8    |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

# РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ СИСТЕМИ УПРАВЛІННЯ АКАДЕМІЧНИМИ РЕСУРСАМИ СТУДЕНТІВ

## 1.1 Постановка задачі

Постановка задачі полягає у розробці єдиного веб-порталу, який об'єднає навчальні, документальні та підтримкові сервіси в однорідному інтерфейсі. Своєчасна обробка даних, автоматизація рутинних процедур та розвинена аналітика мають сприяти зменшенню адміністративного навантаження, підвищенню ефективності навчального процесу та рівня задоволеності користувачів системи.

Сучасні заклади вищої освіти стикаються з дедалі зростаючою складністю організації навчального процесу, що передбачає взаємодію великої кількості зацікавлених сторін: студентів, викладачів та адміністративного персоналу. Існуючі інформаційні сервіси, як системи оцінювання, бібліотеки цифрових матеріалів, платформи для документообігу та комунікаційні засоби, працюють ізольовано один від одного. Унаслідок цього користувачі витрачають значну кількість часу на перемикання між різними інтерфейсами, пошук необхідних даних і виконання рутинних задач вручну. Недостатня інтеграція систем веде не лише до зниження ефективності навчального процесу, а й до підвищення ризику помилок у веденні обліку оцінок, затримок у формуванні довідок та інших адміністративних процедур.

Головною метою даної випускної кваліфікаційної роботи є створення інтегрованої інформаційної системи управління академічними ресурсами студентів, яка дозволить об'єднати функціональні можливості декількох окремих модулів у єдиному веб-порталі. Робота над проектом має забезпечити прозорість і автоматизацію ключових процесів освітнього закладу: від реєстрації й авторизації користувачів до проведення онлайн-тестування з моніторингом безпеки та аналітикою навчальної діяльності. У результаті використання розробленої системи користувачі

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 9    |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

отримають можливість зосередитися на головних освітніх завданнях, а адміністрація - оптимізувати управлінські процедури.

У ролі студента система повинна стати єдиним майданчиком для відстеження успішності та прогресу в навчанні, організації власних матеріалів і виконання лабораторних, практичних та тестових робіт. Студенти можуть легко завантажувати роботи й навчальні файли без необхідності дублювати їх у різних сховищах, а всі оцінки й результати тестувань зберігаються у зручному журналі. Передбачено календар дедлайнів, що допомагає ефективно планувати навчальне навантаження й не пропускати важливі завдання.

Завдяки впровадженню онлайн-тестів з моніторингом активності у вкладках браузера, система мінімізує можливості для порушень під час оцінювання та підтримує академічну доброчесність. Особистий кабінет дозволяє студенту бачити всі свої навчальні та особисті дані, подавати електронні заяви на отримання довідок або вирішення організаційних питань, отримувати зворотний зв'язок від викладачів щодо виконаних завдань, а також переглядати історію комунікації у межах курсу.

Викладач у межах нового порталу отримають зручні механізми для перевірки та оцінювання студентських робіт із вбудованим контролем версій і можливістю встановлення статусів “оцінено”, “відхилено” або “на доопрацювання” та написанням коментарів для версій надісланих робіт. Окремі компоненти конструкторів тестових завдань і лабораторних вправ дозволять формувати різноманітні форми контролю знань без додаткової розробки та налаштувань. Крім того, система забезпечить нагадування про строки перевірки та надасть зручні канали комунікації з авторами робіт для оперативного зворотного зв'язку.

Адміністративний персонал отримує інструменти для централізованого керування користувачами та налаштування прав доступу з розвиненою системою логування дій. Усі основні довідники навчального закладу – факультети, кафедри, спеціальності, освітні програми, академічні групи – доступні для редагування, додавання нових записів або видалення. Також адміністратор може управляти

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 10   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

структурою курсів, складом користувачів, їхніми ролями, а також погоджувати, затверджувати або відхиляти подані студентами заяви й запити на отримання документів.

Подача заяв і формування довідок здійснюються через електронну систему на основі актуальних даних про навчання студента та його статус у закладі. Це спрощує документообіг і дозволяє мінімізувати ручну обробку типових запитів, знижуючи ймовірність помилок та підвищуючи прозорість адміністративних процесів.

## 1.2 Аналіз аналогів програмного продукту

Для визначення ключових вимог і вибору оптимальних архітектурних рішень було проведено огляд існуючих інформаційних систем, що виконують частково або повністю функції, аналогічні майбутньому веб-порталу академічних ресурсів. Найбільш поширені серед них: Moodle, Blackboard, Canvas та Google Classroom.

Moodle – це безкоштовна, відкрита система управління навчанням (рис. 1.1), створена для організації дистанційного навчання у вищих навчальних закладах, школах та корпоративних структурах [1]. Основною особливістю Moodle є її модульність: завдяки системі плагінів користувачі можуть адаптувати платформу під свої потреби без зміни основного коду. Moodle забезпечує можливість створення курсів (рис. 1.2), проведення тестувань, обміну матеріалами, ведення журналу оцінок та організації комунікацій через форуми та чати. Також система підтримує стандарти SCORM, LTI, що спрощує інтеграцію зовнішніх навчальних ресурсів.

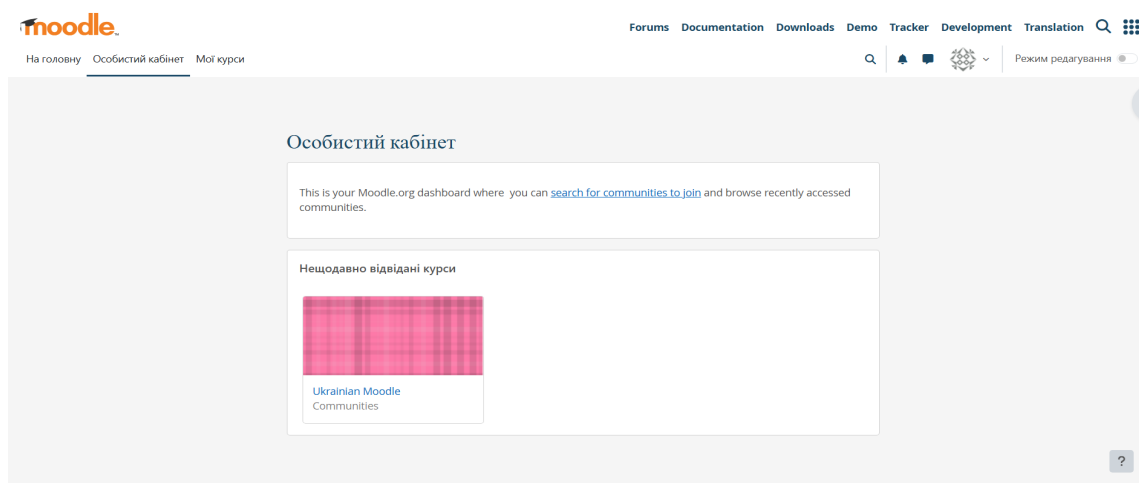


Рис. 1.1 - Особистий кабінет Moodle

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 11   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

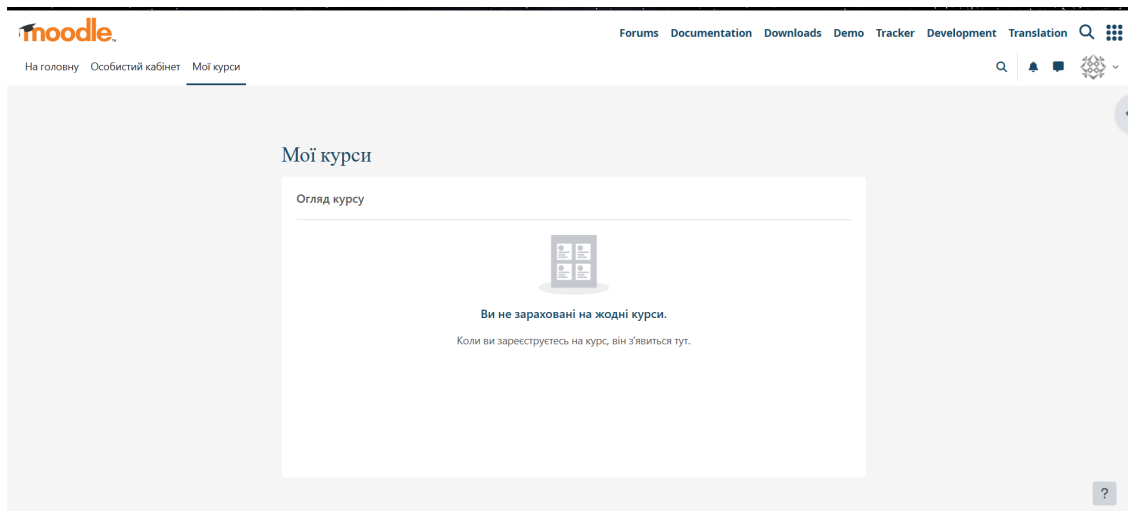


Рис. 1.2 - Сторінка з курсами Moodle

Інтерфейс Moodle можна налаштовувати під різні мови, включно з українською, що робить її зручною для локалізації (рис. 1.3). Платформа має мобільний додаток, дозволяє організувати персоналізовані панелі користувача та використовувати календарі для управління навчальним процесом. Її відкрита архітектура і широке міжнародне співтовариство розробників забезпечують постійний розвиток та підтримку.

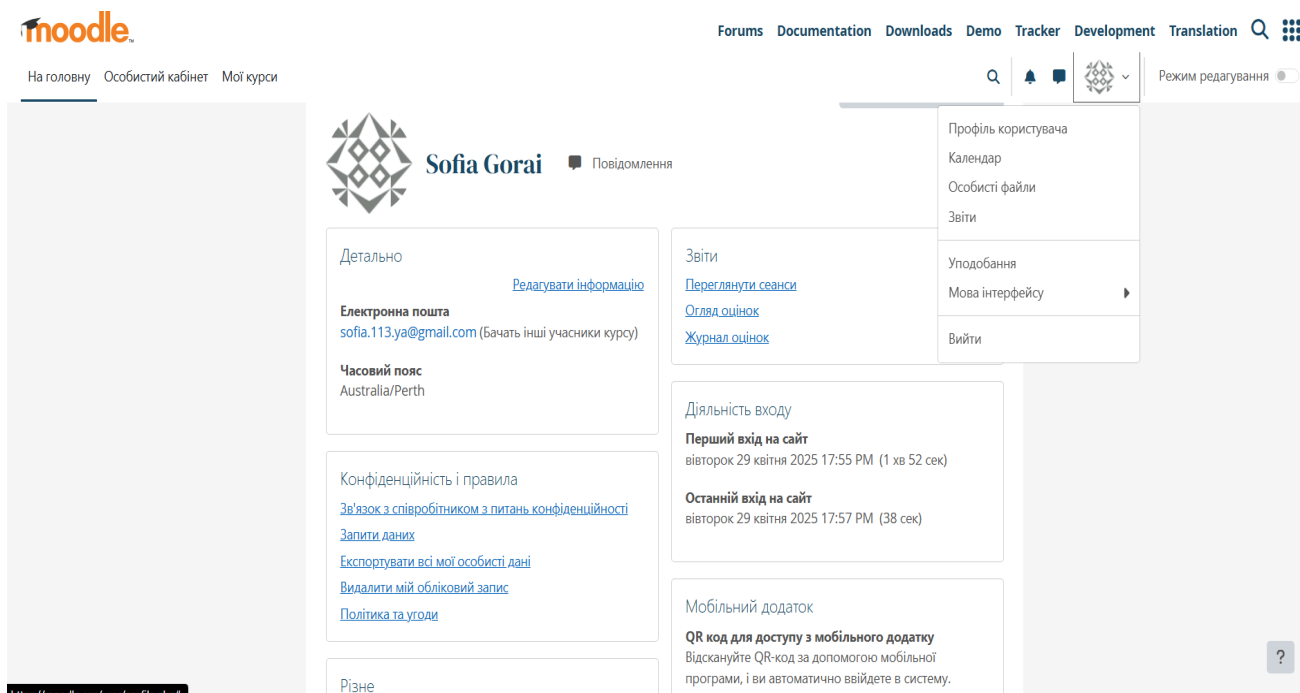


Рис. 1.3 - Особистий кабінет Moodle

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 12   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

Попри свої переваги, Moodle може бути складною для первинного налаштування та потребувати технічної підтримки для інтеграції всіх необхідних функцій. Це важливо враховувати при плануванні системи, орієнтованої на широку аудиторію з різним рівнем цифрової компетентності.

**Blackboard Learn** – це потужна система управління навчанням (рис. 1.4), яка широко використовується в академічному та корпоративному середовищах для організації, проведення та моніторингу навчальних процесів. Розроблена компанією Blackboard Inc. (нині частина Anthology), ця платформа підтримує як традиційне, так і дистанційне навчання, забезпечуючи гнучкість та масштабованість для різних типів освітніх установ [2].

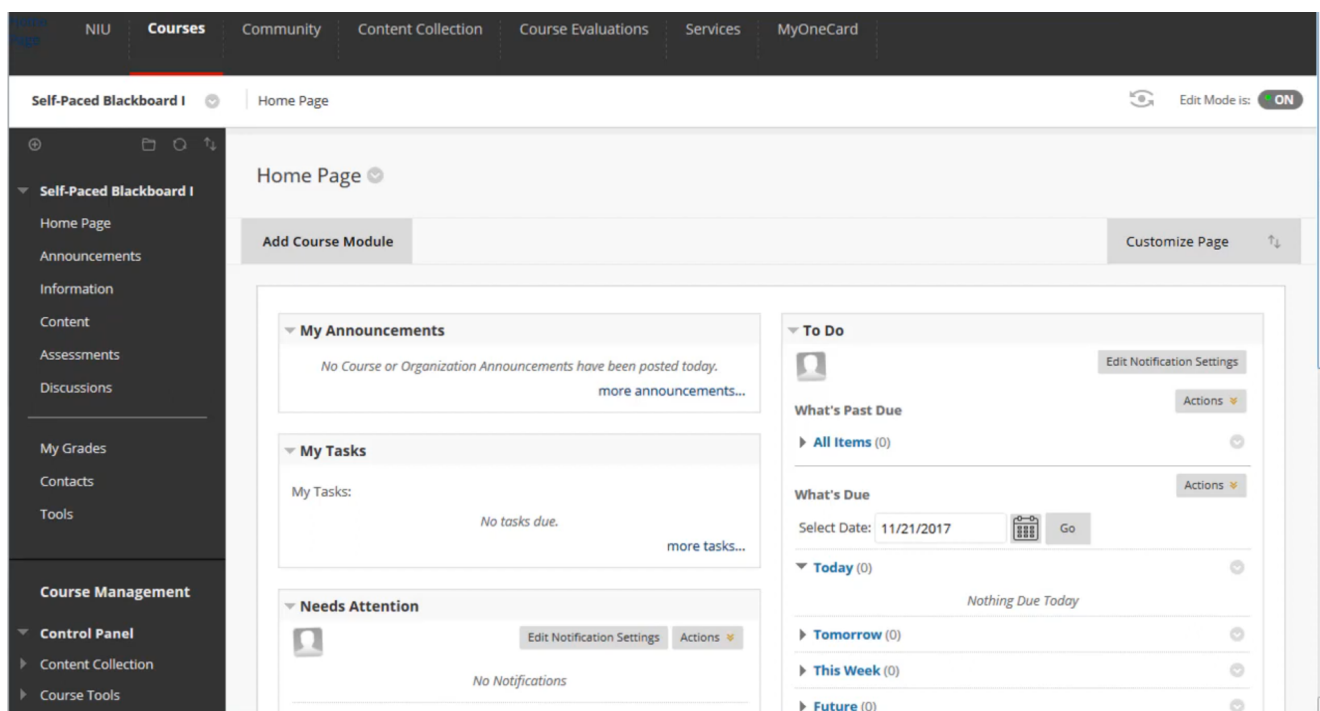


Рис. 1.4 - Приклад інтерфейсу Blackboard Learn

Мобільний додаток Blackboard App відкриває широкі можливості для студентів і викладачів, дозволяючи їм ефективно взаємодіяти з навчальними курсами, брати участь у віртуальних заняттях, проходити оцінювання та своєчасно отримувати сповіщення про важливі події. Однією з ключових функцій платформи є аналітика навчання, що реалізується через інструмент Blackboard Analytics. Завдяки цій можливості як адміністратори, так і студенти можуть відстежувати рівень

успішності, аналізувати активність у курсах та визначати напрямки, які потребують покращення.

Суттєвою перевагою Blackboard є інтеграція з понад 300 зовнішніми сервісами, серед яких Microsoft OneDrive, Dropbox та інші популярні інструменти, що забезпечує безперервний обмін даними та навчальними ресурсами. Також важливою складовою платформи є підтримка користувачів через функцію Blackboard Assist. Вона надає студентам швидкий доступ до ресурсів допомоги, таких як фінансова підтримка, послуги з ментального здоров'я та репетиторство, сприяючи цілісному підходу до навчального процесу.

Ще однією важливою особливістю є гнучкість у розгортанні системи. Blackboard Learn може бути впроваджений як у хмарному середовищі, так і на локальних серверах, що дозволяє навчальним закладам адаптувати платформу відповідно до своїх технічних можливостей та організаційних потреб. Таким чином, Blackboard пропонує всебічне рішення для підтримки навчального процесу в умовах цифрової трансформації освіти.

Попри широкі можливості, Blackboard Learn має деякі недоліки. Користувачі відзначають, що інтерфейс платформи може виглядати застарілим та не завжди інтуїтивно зрозумілим. Крім того, деякі освітні установи повідомляли про технічні проблеми та високі витрати на обслуговування системи.

**Sakai** є відкритою системою управління навчанням (рис. 1.5), створеною в межах спільноти вищих навчальних закладів з метою забезпечення гнучкої платформи для організації навчального процесу як очно, так і дистанційно [3]. Ця платформа була створена не комерційною компанією, а великою міжнародною спільнотою університетів та дослідницьких організацій, що відображає прагнення до спільного розвитку освітніх технологій і ділиться напрацюваннями у відкритому доступі. За допомогою Sakai освітні заклади отримують змогу організовувати курси, управляти навчальними матеріалами, створювати інтерактивні завдання, вести оцінювання й забезпечувати якісний зворотний зв'язок для студентів. Значна увага в цій системі приділяється можливостям інтеграції з іншими освітніми

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 14   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

платформами, зовнішніми сервісами та додатками, що робить Sakai універсальним інструментом для широкого спектра завдань у сфері освіти.

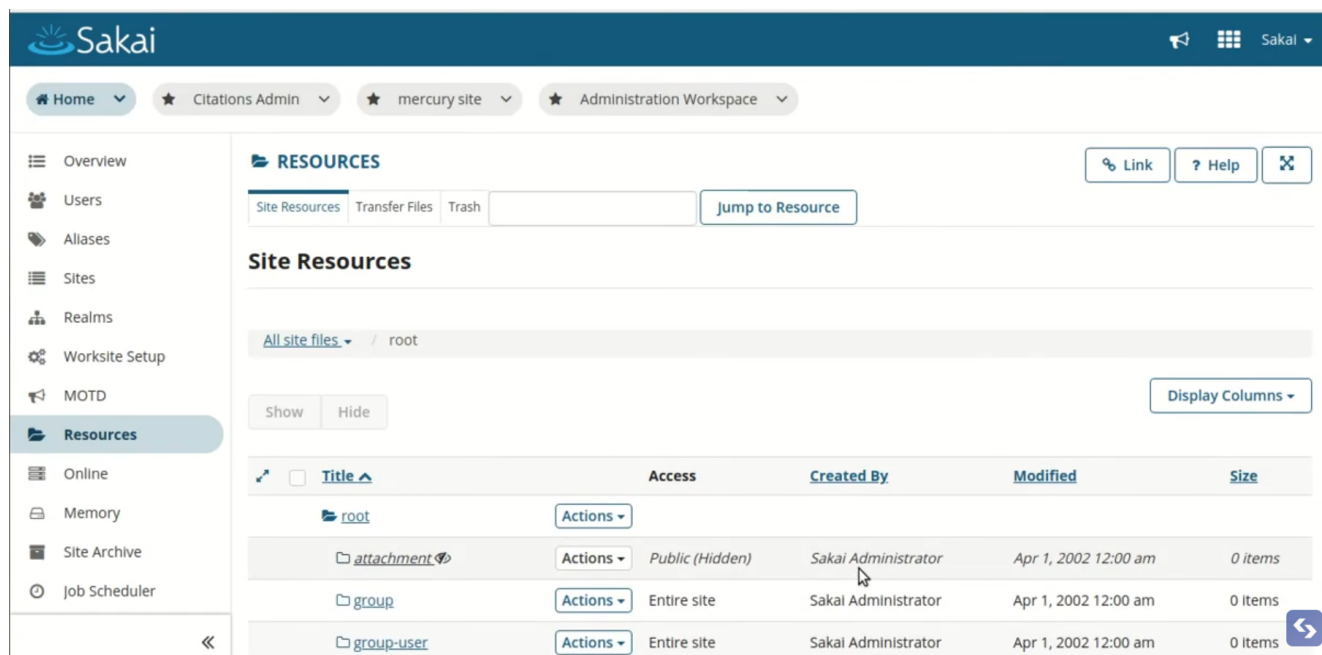


Рис. 1.5 - Приклад інтерфейсу Sakai

Платформа надає широкий спектр інструментів для створення та проведення курсів, включно з управлінням контентом, організацією тестів і завдань, а також журналами оцінок і календарями подій. Завдяки підтримці стандартів IMS LTI та SCORM, Sakai забезпечує глибоку інтеграцію з іншими освітніми сервісами та контент-провайдерами.

Оскільки Sakai є повністю відкритим проєктом, спільнота регулярно розробляє й поширює інструменти для розширення функціональності системи. Пакети додаткових модулів дозволяють інтегрувати платформу з корпоративними LMS, системами єдиного входу (CAS), а також з хмарними сховищами даних й аналітичними сервісами. REST API Sakai надає можливість створювати користувацькі застосунки та синхронізувати дані з іншими інформаційними системами університету. Sakai не завжди вирізняється інтуїтивністю інтерфейсу для нових користувачів, платформа має потужну документаційну підтримку, що суттєво спрощує процес впровадження та адаптації до інституційних потреб..

Порівняльна таблиця існуючих програмних продуктів

| Критерій                | Sakai                                   | Blackboard Learn                                    | Moodle                                                                                        |
|-------------------------|-----------------------------------------|-----------------------------------------------------|-----------------------------------------------------------------------------------------------|
| Мобільний доступ        | Через браузер, мобільна версія сайту    | Офіційний додаток (iOS, Android)                    | Офіційний додаток (iOS, Android) з офлайн-підтримкою                                          |
| Офлайн-режим            | Відсутній                               | Відсутній                                           | Часткова підтримка (перегляд матеріалів, підготовка відповідей і синхронізація через додаток) |
| Оцінки та прогрес       | Веб-доступ, базова аналітика            | Розширена аналітика, персоналізовані звіти          | Детальна система оцінювання, графіки прогресу, персоналізовані звіти                          |
| Комунікація             | Форуми, чати, групова робота            | Форуми, чати, відеозустрічі, блоги                  | Форуми, чати, особисті повідомлення                                                           |
| Подача робіт            | Через браузер, мобільна версія          | Через браузер і частково мобільний додаток          | Через браузер і мобільний додаток                                                             |
| Мови інтерфейсу         | Приблизно 19 мов, включно з українською | Підтримка багатьох мов                              | 120+ мов, включно з українською                                                               |
| Календар та нагадування | Є, базовий                              | Є, інтеграція з Google/Outlook, push-нагадування    | Є, підтримка подій, дедлайнів, експорт в iCal                                                 |
| Сповіщення              | Email, на платформі                     | Push-сповіщення, email, SMS, вбудовані повідомлення | Push-сповіщення, email, вбудовані повідомлення                                                |
| Інтерфейс користувача   | Простий, менш сучасний                  | Функціональний, але частково застарілий             | Гнучкий, адаптивний, підтримує сучасні теми                                                   |
| Підтримка доступності   | Часткова                                | Повна підтримка                                     | Повна підтримка                                                                               |

На основі проведеного аналізу аналогів було визначено перелік критично важливих функціональних можливостей, які повинна підтримувати майбутня інформаційна система. Насамперед це зручний та інтуїтивно зрозумілий інтерфейс для подачі, перевірки й оцінювання завдань, система сповіщень і нагадувань, що синхронізується з календарем, підтримка різних форматів навчального контенту, а також наявність візуалізації навчального прогресу та історії оцінювання. Не менш важливою є підтримка української мови, доступність для студентів з особливими потребами та можливість гнучкого розмежування прав доступу для різних ролей користувачів – студентів, викладачів і адміністраторів.

Також було визначено ключові аспекти, яких необхідно уникати під час розробки: перевантаженість інтерфейсу функціями, які не мають реальної цінності

для користувача; складна навігація та надмірна кількість кроків для виконання базових дій; відсутність адаптації під локальні освітні вимоги та потреби; низька швидкодія при великій кількості одночасних користувачів; а також обмеження в масштабованості, які ускладнюють впровадження системи в установах різного масштабу. Урахування цих факторів дозволить створити справді ефективний інструмент підтримки навчального процесу.

### **1.3 Вибір архітектури системи управління академічними ресурсами**

Під час проєктування інформаційної системи управління академічними ресурсами критично важливо обрати архітектурний підхід, який забезпечить надійність, масштабованість, простоту супроводу та гнучкість у подальшому розвитку. Для цього на початковому етапі доцільно розглянути найбільш поширені архітектурні шаблони, які використовуються у розробці сучасних веб-застосунків. Архітектурні шаблони відіграють роль каркасу, що визначає взаємодію компонентів, їхню відповідальність та спосіб розгортання. Недостатньо лише реалізувати набір функціональних блоків – необхідно також оцінити, наскільки обрана архітектура відповідатиме вимогам продуктивності, безпеки й можливості еволюції системи [4].

Серед найвідоміших архітектурних підходів варто виокремити монолітну архітектуру, клієнт-серверну модель, мікросервісну архітектуру, та клієнт-серверну архітектуру. У межах аналізу було розглянуто кілька загальноприйнятих архітектурних стилів, які відрізняються ступенем декомпозиції системи та підходами до інтеграції компонентів.

Монолітна архітектура передбачає єдине середовище виконання, де презентаційний шар, бізнес-логіка та рівень доступу до даних з'єднані в один артефакт. Такий підхід спрощує розгортання та тестування початкових версій, проте зростання системи призводить до суттєвого ускладнення підтримки, оскільки будь-яке розширення функціоналу вимагає повторного розгортання всієї платформи, а масштабування окремих фрагментів стає неможливим без фізичного поділу моноліту [27].

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 17   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

Мікросервісна архітектура пропонує альтернативу за рахунок розподілу системи на групу незалежних сервісів, кожен із яких відповідає за чітко окреслену бізнес-функцію. Кожний сервіс розгортається автономно та може бути масштабований відповідно до обсягу навантаження. Незалежність технологічних стеків дозволяє обирати оптимальні інструменти для кожного з компонентів, водночас вимагає впровадження інфраструктурних механізмів для балансування навантаження, реєстрації сервісів, централізованого моніторингу та трасування міжсервісних запитів. Організація розподілених транзакцій також ускладнюється через відсутність єдиного середовища виконання .

Схожою за ідеєю є сервіс-орієнтована архітектура (SOA), але вона орієнтована на більші сервіси [25]. Перевагою такого шаблону є гнучкість у використанні різних технологій і мов програмування для різних сервісів. Сервіси можна перевикористовувати, що зменшує дублювання логіки. Проте його складано інтегрувати та налаштувати комуніції між сервісами.

Клієнт-серверна архітектура (рис. 1.6), в якій чітко розділено обов'язки між клієнтською та серверною частинами, дозволяє поєднати простоту реалізації з необхідною масштабованістю. Клієнтська складова, реалізована як односторінковий веб-додаток, відповідає за презентаційний рівень і часткову валідацію даних, тоді як серверна частина, побудована на базі FastAPI, забезпечує обробку бізнес-логіки, централізоване зберігання інформації та контроль безпеки. Такий поділ дозволяє оновлювати сервер незалежно від клієнтського інтерфейсу, а також масштабувати кожний рівень окремо.

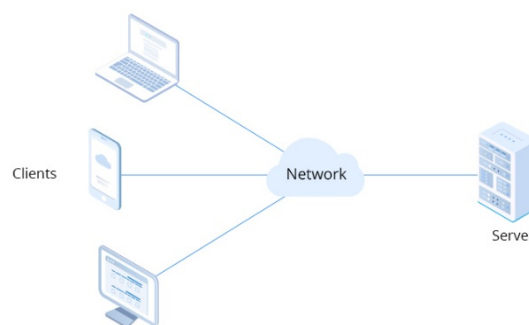


Рис. 1.6 - Клієнт-серверна архітектура

Проаналізувавши основні архітектурні шаблони, було обрано клієнт-серверну архітектуру як найбільш збалансоване рішення для побудови системи управління академічними ресурсами. Такий підхід дозволяє чітко розмежувати функціональні обов'язки між клієнтською та серверною частинами, що сприяє підвищенню структурованості коду, спрощенню його супроводу та забезпеченню високого рівня безпеки. У клієнт-серверній моделі усі критично важливі операції, пов'язані з обробкою даних, автентифікацією користувачів, контролем доступу та реалізацією бізнес-логіки, відбуваються на сервері. Це дозволяє централізовано контролювати інформаційні потоки, здійснювати журналювання дій користувачів і швидко реагувати на потенційні загрози.

Окрім цього, клієнт-серверна архітектура вирізняється високою адаптивністю до зростання навантаження. Завдяки можливості горизонтального масштабування серверної інфраструктури та розгортанню клієнтської частини в окремому середовищі, наприклад у хмарному сервісі або на незалежному хості, забезпечується стабільна робота системи навіть за умов великої кількості одночасних користувачів [26]. На відміну від монолітних архітектур, які складно масштабувати, або складних розподілених рішень на основі мікросервісів, клієнт-серверна модель забезпечує баланс між технічною реалізацією, керованістю, продуктивністю та можливістю подальшого розвитку без істотного ускладнення інфраструктури.

#### 1.4 Обґрунтування вибору інструментальних засобів.

Створення інформаційної системи управління академічними ресурсами потребує застосування сучасних інструментальних засобів, які дозволяють забезпечити гнучкість, продуктивність, надійність, масштабованість та безпеку всієї програмної інфраструктури. У зв'язку з цим постає необхідність обґрунтованого вибору технологій та середовищ, які не лише відповідають функціональним вимогам системи, але й дозволяють ефективно реалізувати її архітектурну модель, спрощують розгортання, супровід і майбутнє розширення.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 19   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

У якості серверного фреймворку було обрано FastAPI – високопродуктивне рішення, що функціонує на базі асинхронного стандарту ASGI (Asynchronous Server Gateway Interface) [5]. Цей фреймворк, завдяки своїй здатності обробляти запити паралельно та асинхронно, дозволяє значно підвищити швидкодію системи в умовах великої кількості одночасних користувачів. Його глибока інтеграція з Python type hints та використання Pydantic для валідації даних гарантує не лише чистоту та безпомилковість коду, але й дозволяє автоматично генерувати документацію за стандартом OpenAPI. Це особливо важливо в умовах потреби у прозорості інтерфейсів для зовнішніх сервісів.

У побудові програмної архітектури важливим елементом є правильний вибір патернів програмування, оскільки вони дозволяють зробити систему більш гнучкою, легкою для підтримки та масштабованою. Одним з таких патернів є Dependency Injection (DI), що використовується для зменшення зв'язності між класами і забезпечення інверсії залежностей. Застосування цього патерну дозволяє покращити тестованість, зменшити залежність компонентів один від одного та підвищити модульність коду [6].

FastAPI, обраний для розробки серверної частини, підтримує ін'єкцію залежностей через функціональні залежності. Це дозволяє ізолювати компоненти системи, підвищуючи тестованість і забезпечуючи гнучкість при роботі з сервісами. FastAPI реалізує DI через параметри функцій, що дає можливість при створенні API запитів чітко визначати, які залежності мають бути надані для обробки запиту. Вкладені залежності підтримують багаторівневу композицію сервісів, що полегшує масштабування й підтримку складних систем. Це робить архітектуру застосунку більш прозорою, а процес додавання нових можливостей – швидким і контрольованим.

Завдяки використанню DI в FastAPI можна ефективно управляти з'єднаннями з базою даних, кешуванням, обробкою запитів та іншими сервісами, що безпосередньо взаємодіють між собою. Крім того, це дозволяє легко тестувати кожен частину системи без необхідності запускати весь стек сервісів.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 20   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

У ролі клієнтської частини обрано React – сучасну бібліотеку JavaScript для створення динамічних користувацьких інтерфейсів [7]. React забезпечує розробку за компонентною архітектурою, що дозволяє розділяти інтерфейс на модулі, які легко тестуються, підтримуються та повторно використовуються. Це особливо важливо для систем, які будуть з часом доповнюватися новими функціями, інтерфейсами та сценаріями взаємодії. Завдяки використанню віртуального DOM (Document Object Model), React забезпечує високу ефективність оновлення сторінок, що позитивно впливає на взаємодію користувача з системою в реальному часі. Крім того, підтримка бібліотек управління станом (Redux) спрощує побудову складних форм та інтерактивних елементів.

У контексті зберігання даних вибір зупинено на PostgreSQL – потужній об'єктно-реляційній системі керування базами даних з відкритим кодом, яка підтримує складні SQL-запити, транзакції, тригери, представлення, розширення та широкі можливості роботи з JSON [8]. PostgreSQL визнана однією з найстабільніших і надійних СКБД, що підтверджується її багаторічним використанням у проєктах корпоративного рівня. Її підтримка паралельних запитів і масштабування дозволяє легко адаптуватися до збільшення навантаження. До того ж, PostgreSQL підтримує механізми реплікації, що стане у пригоді для побудови резервних інстанцій або забезпечення високої доступності.

Для адміністрування та візуальної роботи з базою даних використовується pgAdmin – офіційний графічний інтерфейс для PostgreSQL, що дозволяє керувати структурою БД, виконувати SQL-запити, контролювати користувачів і аналізувати журнали подій у зручному візуальному режимі. Це спрощує супровід бази даних як для розробників, так і для адміністративного персоналу.

Інтеграція всіх програмних компонентів системи та їх ефективне управління реалізується за допомогою Docker – платформи віртуалізації на рівні операційної системи, яка дозволяє створювати, розгортати та запускати додатки у відокремлених середовищах (контейнерах) [9]. Це дає можливість забезпечити ізоляцію середовищ виконання, усунути проблеми несумісності залежностей, а також стандартизувати процеси розгортання як у тестових, так і у бойових середовищах. Усі

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 21   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

компоненти – серверна частина, клієнтська частина, база даних, проксі-сервер, сервіси моніторингу – розгортаються як окремі контейнери [10].

Для управління багатоконтейнерним середовищем у процесі розробки використовується Docker Compose [11]. Цей інструмент дозволяє в одному YAML-файлі визначити усі сервіси, які складають систему, та запустити їх однією командою. Це значно спрощує локальну розробку, забезпечуючи швидкий старт для розробників та зниження ймовірності помилок при налаштуванні інфраструктури вручну [12].

На етапі промислової експлуатації доцільним є застосування Docker Swarm – інструменту оркестрації, який дозволяє створювати кластери Docker-хостів, що працюють як єдиний віртуальний сервер [13]. Docker Swarm забезпечує автоматичний розподіл контейнерів по вузлах, моніторинг їхнього стану, автоматичний перезапуск у випадку збою, можливість плавного оновлення сервісів (rolling updates) та балансування навантаження. Завдяки вбудованим механізмам безпеки, таким як TLS-шифрування трафіку між вузлами та керування секретами, він гарантує захист переданих даних та безпеку внутрішньої мережі. У випадку масштабування системи на декілька фізичних або віртуальних серверів, Docker Swarm є важливим компонентом забезпечення безперебійної роботи системи.

Ще одним критично важливим компонентом є проксі-сервер Traefik, який виконує роль зворотного проксі та маршрутизатора запитів [14]. Traefik автоматично виявляє запущені у Docker сервіси, підключає до них HTTPS із підтримкою Let's Encrypt, балансує запити між репліками одного сервісу та дозволяє зручно управляти маршрутизацією за допомогою міток. Його гнучкість та підтримка динамічної конфігурації в реальному часі робить його ідеальним рішенням для мікросервісної або контейнеризованої архітектури [15].

Обраний набір інструментальних засобів формує надійну, масштабовану, продуктивну і безпечну екосистему для реалізації складної багатокомпонентної інформаційної системи управління академічними ресурсами. Його структура дозволяє не лише ефективно реалізувати поточні функціональні вимоги, а й забезпечити

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 22   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

високий рівень адаптивності до майбутніх змін, модифікацій та інтеграцій з іншими цифровими освітніми сервісами.

### 1.5 Розподіл ролей та принципи командної взаємодії

У процесі розробки інформаційної системи особливу увагу було приділено організації ефективної командної роботи. Замість жорсткого поділу на окремі ролі, як фронтенд- чи бекенд-розробник, було обрано модель рівноправної участі у створенні програмного продукту. Такий підхід дав змогу забезпечити гнучкість у розподілі завдань і дозволив кожному учаснику долучатися до тих етапів роботи, де це було необхідно з огляду на загальні потреби проєкту.

Використання рівноправного підходу сприяло формуванню універсальності в навичках кожного учасника команди, адже під час розробки доводилося працювати як із логікою серверної частини, так і з клієнтським інтерфейсом, взаємодіяти з базою даних, реалізовувати маршрути API, обробляти запити, опрацьовувати помилки та забезпечувати узгодженість у взаємодії між модулями. Комунікація відбувалася на регулярній основі, що дозволяло вчасно узгоджувати деталі реалізації, уточнювати логіку взаємодії компонентів системи та вирішувати суперечливі або неоднозначні ситуації.

Особливістю командної взаємодії було також те, що кожен учасник міг пропонувати зміни до коду інших членів команди. Це передбачало не лише реалізацію нових функцій, але й рефакторинг, оптимізацію, виправлення багів, оновлення документації або адаптацію під нові вимоги. Подібна взаємодія дозволила підтримувати єдині підходи до організації коду, структури модулів, обробки помилок та реалізації бізнес-логіки, що, у свою чергу, сприяло підвищенню загальної якості програмного продукту.

Комплексна відповідальність за успіх розробки вимагала постійного узгодження дій, взаємної допомоги, перегляду змін, обговорення архітектурних рішень та розподілу задач. Кожен учасник мав змогу впливати на технічні рішення, пропонувати поліпшення або альтернативні варіанти реалізації. Це забезпечило не лише

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 23   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

технічну цілісність проєкту, а й ефективну співпрацю в команді, що є важливим елементом у сучасній розробці програмного забезпечення.

Моєю зоною відповідальності в командному проєкті був навчальний модуль і система електронних документів. Було реалізовано функціонал уроків, лабораторних робіт і тестування з підтримкою типів запитань, обмеженнями за часом і кількістю спроб, а також перевіркою відповідей і моніторингом активності. Крім того, було розроблено модуль подачі заявок на документи з використанням шаблонів Jinja2, генерацією PDF через WeasyPrint, HTML-прев'ю та системою погодження за ролями. Уся логіка побудована з урахуванням масштабованості та розділення відповідальностей. Постійна синхронізація команди через спільний репозиторій і Jira забезпечила узгоджену інтеграцію функціоналу та запобігла дублюванню зусиль.

У рамках комплексної розробки програмного забезпечення важливим елементом організаційної діяльності стала робота з системою керування завданнями Jira [24]. Уся взаємодія всередині команди, планування, відстеження виконання задач і контроль за прогресом проєкту здійснювались саме через цей інструмент, що дало змогу забезпечити структурованість і прозорість процесу розробки.

Проєкт було розбито на низку окремих задач (рис. 1.7), кожна з яких фіксувалась у Jira як окрема таска з унікальним ідентифікатором, наприклад, SU-1, SU-2 тощо. Кожна задача мала короткий опис суті роботи, пріоритет, очікуваний результат та була призначена відповідальному розробнику. Таким чином, на кожному етапі виконання проєкту можна було чітко бачити, хто за що відповідає, які задачі вже реалізовані, а які ще перебувають у процесі виконання або очікують на перевірку.

Кожен розробник при роботі над певною функціональністю або модулем системи створював відповідну задачу в Jira або брав на себе вже створену. Наприклад, реалізація первинної структури бази даних, створення моделей, CRUD-операцій, API-маршрутів, тестів чи сторінок для роботи з документами – усе це оформлювалося у вигляді окремих задач. Після завершення роботи над конкретною частиною

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 24   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

функціоналу розробник зазначав задачу як виконану та додавав коментарі у Jira щодо внесених змін або труднощів, що виникли під час реалізації.

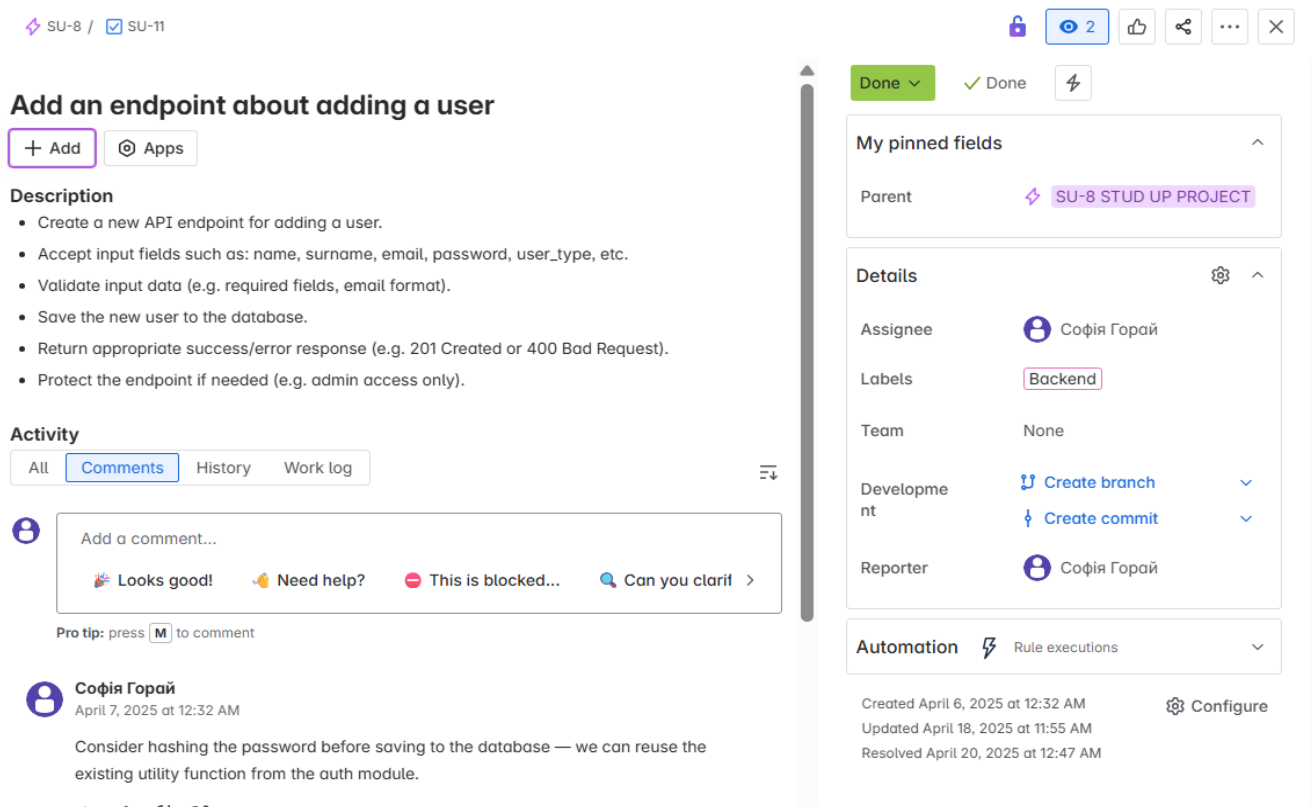


Рис. 1.7 – Робота в середовищі Jira

У процесі комплексної роботи використання статусів задач стало одним із ключових інструментів для організації ефективного workflow. Завдяки чіткій системі статусів, таких як «To Do» (заплановано до виконання), «In Progress» (у роботі), «Waiting» (очікує на зовнішній тригер, наприклад, перевірку чи допомогу) та «Done» (завершено) (рис.1.8), команда могла легко відслідковувати, на якому етапі перебуває кожне завдання. Це забезпечувало не лише прозорість у розподілі відповідальності, а й дозволяло уникати ситуацій, коли певні задачі «зависають» без уваги.

Особливо цінним був візуальний контроль за виконанням: на дошці Jira кожен розробник міг у реальному часі бачити поточний стан задач, оперативно переміщувати їх між статусами або додавати необхідні коментарі. Таке візуалізоване представлення workflow дозволяло швидко оцінювати загальний прогрес, розуміти, скільки залишилось до завершення того чи іншого етапу, і своєчасно реагувати на можливі затримки чи зміни пріоритетів.

Оновлення статусів задач відбувалося не лише формально, а й із врахуванням зворотного зв'язку. Регулярні обговорення допомагали не лише зафіксувати поточний стан справ, а й уточнити деталі, за потреби передати задачу іншому розробнику або змінити її пріоритет. Така гнучкість у керуванні процесом давала змогу миттєво адаптуватися до нових викликів або форс-мажорних обставин, що неминуче виникають у будь-якому проєкті. Як результат, система статусів задач стала невід'ємною частиною нашого командного workflow, яка не лише спрощувала контроль, а й робила весь процес розробки більш передбачуваним і прозорим для всіх учасників.

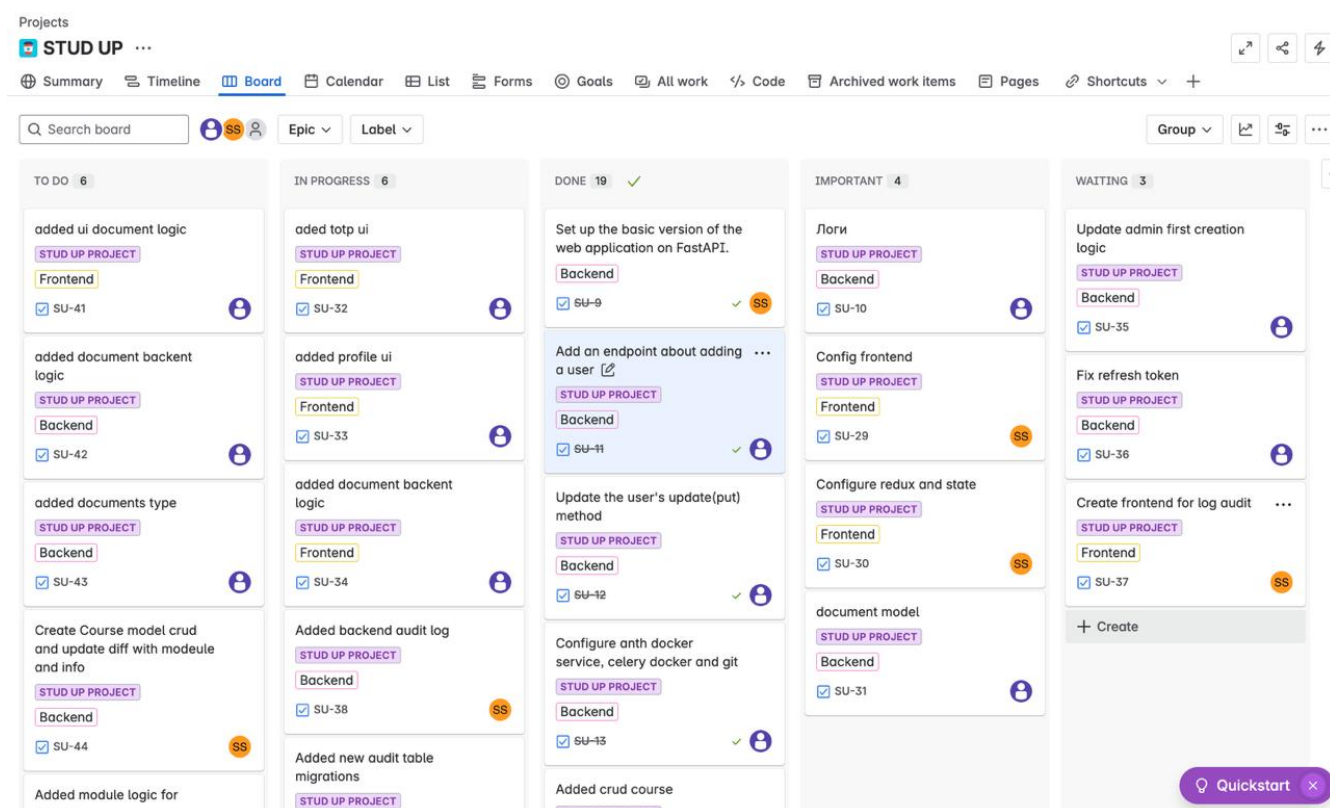


Рис. 1.8 – Робота в середовищі Jira

Загалом робота з Jira стала фундаментальною частиною організації командної взаємодії. Вона забезпечила високу прозорість, послідовність виконання завдань, простоту контролю за реалізацією кожного компонента системи, а також дозволила досягнути цілісності проєкту в умовах спільної, нерозмежованої участі всіх розробників. Використання Jira сприяло ефективному плануванню спринтів та забезпечило оперативне реагування на зміну пріоритетів під час розробки. Завдяки

централізованому управлінню задачами команда змогла підтримувати постійний прогрес і своєчасно виявляти та усувати проблеми в процесі розробки.

У межах комплексної роботи над проєктом важливою складовою процесу стала взаємодія через систему контролю версій Git. Спільна розробка вимагала ретельно організованого підходу до роботи з репозиторієм, дотримання узгоджених практик гілкування, комітування та оновлення локальних змін відповідно до актуального стану проєкту.

Основною гілкою розробки була dev, у яку інтегрувалися зміни після завершення відповідних завдань та проходження перевірки. Для роботи над окремими частинами системи кожен учасник створював власні гілки, зазвичай із назвою, що відповідала задачі в Jira – наприклад, SU-12-password-reset. Це дозволяло уникати конфліктів і паралельно реалізовувати кілька модулів або фіч без взаємного блокування.

Перед тим як виконати git push, регулярно оновлювались локальні репозиторії за допомогою git pull (рис. 1.9), щоб отримати останні зміни від інших учасників команди (рис. 1.10). У випадках, коли ці зміни конфліктували з локальними напрацюваннями, виникали merge-конфлікти. Їх вирішення відбувалось вручну – через уважне переглядення коду, видалення застарілих фрагментів, узгодження актуального функціоналу та тестування результату після об'єднання. У складніших випадках узгоджували зміни колегіально, щоб зберегти цілісність логіки та уникнути дублювання коду.

```
create mode 100644 backend/app/alembic/versions/2025_06_15_1629-f6a9a456f790_add_chat_models.py
create mode 100644 backend/app/app/api/api_v1/endpoints/different/chat.py
create mode 100644 backend/app/app/crud/different/crud_chat.py
create mode 100644 backend/app/app/models/chat/chat.py
create mode 100644 backend/app/app/schemas/chat/chat.py
create mode 100644 frontend/icons/chrome.svg
create mode 100644 frontend/icons/globe.svg
create mode 100644 frontend/icons/mozilla.png
create mode 100644 frontend/icons/safari.png
create mode 100644 frontend/src/api/ChatApi.ts
create mode 100644 frontend/src/api/SessionsApi.ts
create mode 100644 frontend/src/components/ChatListPage.tsx
create mode 100644 frontend/src/components/ChatView.tsx
create mode 100644 frontend/src/pages/ChatPage.tsx
create mode 100644 frontend/src/pages/SessionListPage.tsx
create mode 100644 frontend/src/socket/chatSocket.ts
```

Рис. 1.9 – Список нових файлів у репозиторії після виконання git pull

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 27   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

```

C:\Users\Admin\OneDrive\Робочий стіл\stud-up-dev>git pull
remote: Enumerating objects: 37, done.
remote: Counting objects: 100% (37/37), done.
remote: Compressing objects: 100% (23/23), done.
remote: Total 23 (delta 19), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (23/23), 2.41 KiB | 38.00 KiB/s, done.
From git.ztu.edu.ua:ipz/2021-2025/ipz-21-4/horai-sofia/stud-up
 02834ff..f9df3fd dev      -> origin/dev
Updating 02834ff..f9df3fd
Fast-forward
 .../api_v1/endpoints/education_module/course.py    | 4 +-
 backend/app/app/crud/education/info.py             | 30 +++++
 backend/app/app/crud/user/crud_user.py             | 97 ++++++
 3 files changed, 83 insertions(+), 48 deletions(-)

C:\Users\Admin\OneDrive\Робочий стіл\stud-up-dev>

```

Рис. 1.10 – Огляд змін у проєкті після виконання git pull

Після завершення роботи над певною функціональністю та її тестуванням відбувалося злиття гілки в main. Кожен коміт супроводжувався змістовним повідомленням із зазначенням ідентифікатора Jira-завдання, наприклад: SU-17: Реалізація логіки PDF-генерації. Такий підхід дозволяв легко відстежувати зміни у кодовій базі та забезпечував прозорість історії розробки. У випадках виникнення конфліктів під час злиття гілок оперативно вирішувались шляхом спільного аналізу та погодження змін. Регулярне оновлення локальних репозиторіїв через pull забезпечувало синхронізацію з актуальним станом проєкту.

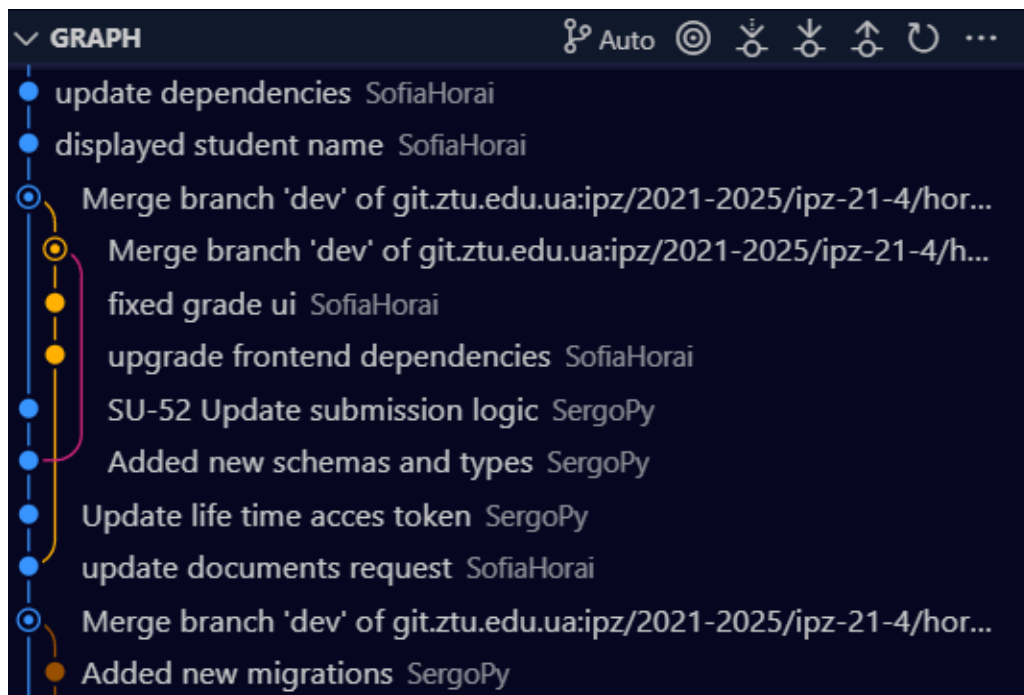


Рис. 1.11 – Структура гілок та комітів у системі контролю версій git

Особливу увагу було приділено логічному структуруванню комітів, що спрощувало відстеження змін і дозволяло швидко повернутися до попередньої версії в разі потреби.

Комплексна робота з Git забезпечила ефективну синхронізацію змін, паралельний розвиток окремих частин системи, а також безпечне та контрольоване впровадження нового функціоналу. Завдяки дотриманню єдиних правил і злагоджень співпраці, розробка проєкту відбувалася стабільно та з дотриманням високих стандартів якості коду.

### **Висновки до першого розділу**

У розділі було проаналізовано функціональні потреби системи управління академічними ресурсами у контексті освітнього процесу. Визначено основні модулі системи, серед яких: інструменти для виконання лабораторних, практичних та тестових робіт, модуль створення й проведення тестування, система моніторингу академічної доброчесності, планувальник завдань, а також система управління користувачами та наповненням модулів. Розглянуто механізми взаємодії між користувачами (студент, викладач, адміністратор) та системою. Особливу увагу приділено ролям користувачів і рівням доступу, що забезпечує гнучке керування правами в межах навчальних курсів. Для збереження інформації передбачено структуру бази даних з чітко визначеними зв'язками, яка підтримує масштабування та розширення.

Окремо охарактеризовано процеси реєстрації та автентифікації користувачів, організацію електронного документообігу та подачі заяв, а також інструменти для контролю виконання завдань і аналізу результатів навчання. Проаналізовано типові сценарії використання системи представниками кожної ролі, що дало змогу сформувати перелік функціональних вимог для подальшого проєктування та реалізації інформаційної системи.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 29   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

## РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ УПРАВЛІННЯ АКАДЕМІЧНИМИ РЕСУРСАМИ СТУДЕНТІВ

### 2.1 Специфікація вимог до програмного продукту

Проектування інформаційної системи управління академічними ресурсами студентів розпочинається із чіткого формулювання функціональних і нефункціональних вимог. Саме цей етап є основоположним усього життєвого циклу створення програмного забезпечення, оскільки від правильного й вчасного визначення вимог залежить не лише кінцева функціональність системи, а й ступінь задоволення реальних потреб її майбутніх користувачів, адаптивність до змін у процесі розвитку, можливість масштабування та інтеграції з іншими сервісами. Значна увага в цьому процесі приділяється не лише переліку бажаних можливостей, а й деталізації сценаріїв використання, визначенню ролей учасників, окресленню критичних моментів безпеки, надійності, продуктивності та підтримки. Вимоги відображають очікування різних груп користувачів, які задають орієнтири для всіх наступних етапів: архітектурного проектування, вибору технологій, моделювання даних, розробки інтерфейсів і, зрештою, для тестування готової системи.

**Бізнес-вимоги** до системи передбачають створення сучасного інтегрованого програмного рішення, орієнтованого на автоматизацію основних процесів у сфері вищої освіти. Основною метою є впровадження платформи, яка забезпечить централізоване управління навчальними, адміністративними та фінансовими аспектами взаємодії між студентами, викладачами та адміністрацією закладу.

Система повинна надати всім категоріям користувачів зручний та оперативний доступ до ключових функцій: студенти повинні мати змогу переглядати свої результати навчання, завантажувати освітні матеріали, надсилати виконані завдання, формувати заявки на документи та здійснювати оплату за навчання. Для викладачів необхідні інструменти для створення завдань, тестів, перевірки робіт студентів та виставлення оцінок. Адміністративний персонал має отримати доступ до механізмів керування навчальними курсами, користувачами, правами доступу, документацією та загальним контентом платформи.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 30   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

Важливим завданням системи є мінімізація ручної роботи, підвищення прозорості та ефективності комунікації між усіма учасниками освітнього процесу, а також створення єдиного інформаційного середовища для прийняття управлінських рішень на основі актуальних даних.

**Представлення** проєкту полягає в розробці веб-платформи, яка об'єднує різні аспекти навчання, фінансів та адміністративних процесів у єдину систему. Платформа включатиме кілька ключових модулів, таких як кабінет студента, який дасть змогу отримувати актуальну інформацію про успішність, матеріали для навчання та фінансові дані. Студенти також матимуть можливість взаємодіяти з викладачами через чат. Викладачі зможуть перевіряти роботи, створювати завдання та керувати оцінками в кабінеті викладача. Кабінет адміністратора дозволить здійснювати управління користувачами, контролювати доступи до системи та вносити зміни в контент. Окрім цього, система включатиме модуль підтримки студентів з можливістю коригування навчального графіка та кар'єрного консультування, а також систему нотифікацій для своєчасного повідомлення про важливі події. Всі ці компоненти будуть інтегровані в одну платформу, щоб забезпечити ефективну взаємодію між учасниками освітнього процесу, автоматизувати адміністративні задачі та підвищити загальну ефективність навчання.

#### **Вимоги до ролі студента:**

- Можливість перегляду своєї успішності.
- Доступ до навчальних матеріалів та завантажених ресурсів.
- Інтерфейс для виконання робіт.
- Автоматичне формування документів (наприклад, довідок, заяв).
- Спостереження під час тестів для забезпечення чесності.
- Функціонал для подання заяв на переведення, академічну відпустку чи перерву.

#### **Вимоги до ролі викладача:**

- Можливість перевірки студентських робіт та оцінювання результатів.
- Інтерфейс для створення тестів та завдань для студентів.
- Підтримка системи спостереження під час тестів для забезпечення чесності.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 31   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

- Легкість в управлінні навчальними матеріалами та розподіленими завданнями.
- Можливість оновлення інформації щодо студентів та їхнього навчального прогресу.

#### **Вимоги до ролі адміністратора:**

- Можливість керування користувачами, включаючи зміну ролей та видалення акаунтів.
- Контроль доступу та безпеки даних користувачів.
- Управління контентом системи (додавання, редагування та видалення матеріалів).
- Функції для моніторингу та аналізу роботи системи в цілому.
- Інструменти для підтримки студентів, включаючи гнучкий графік навчання та кар'єрне консультування.
- Можливість запуску та моніторингу автоматичних процесів, таких як формування документів

#### **Характеристика об'єкта комп'ютеризації:**

Об'єктом комп'ютеризації є навчальний заклад, що функціонує в межах сучасної освітньої інфраструктури та потребує цифрової трансформації ключових процесів управління академічними ресурсами. Основною метою автоматизації є створення єдиної інформаційної системи, яка забезпечить централізований, безпечний і зручний доступ до навчальних, адміністративних та комунікаційних сервісів для всіх категорій користувачів. Основними користувачами системи є студенти, викладачі та адміністрація навчального закладу.

Студент має персоналізований простір для навчання. Він може переглядати свої оцінки й аналізувати динаміку власної успішності, швидко знаходити навчальні матеріали з усіх дисциплін, ознайомлюватися із завданнями, які потрібно виконати, і своєчасно їх подавати. Система також забезпечує зручний канал комунікації зі своїми викладачами, дозволяє зберігати історію власного прогресу. Для студента це не лише інструмент контролю, а й потужний засіб самоорганізації та планування навчання.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 32   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

Викладач використовує систему для організації та проведення освітнього процесу. В його розпорядженні – інструменти для створення нових завдань, розміщення навчальних матеріалів, проведення тестувань і опитувань. Перевірка робіт студентів та виставлення оцінок відбуваються централізовано, з можливістю залишати розгорнуті коментарі й аналізувати результати.

Адміністратор закладу відповідає за підтримку стабільної та безпечної роботи всієї системи. На рівні адміністратора реалізовано повний набір функцій для управління обліковими записами користувачів, призначення й контролю ролей, розподілу доступів до різних модулів і розділів платформи. Адміністрація здійснює моніторинг подій, налаштування політик безпеки, резервне копіювання даних, а також має можливість формувати звіти щодо роботи системи й статистики використання.

#### **Функціональні вимоги:**

Система управління академічними ресурсами студентів має забезпечити функціональність для трьох основних груп користувачів: студентів, викладачів і адміністрації. Студенти повинні мати доступ до особистого кабінету, де відображається їхня успішність, навчальний прогрес, доступні навчальні матеріали, а також можливість виконання лабораторних і контрольних робіт, проходження тестів із системою контролю доброчесності, збереження матеріалів на Google Drive, спілкування з викладачами через чат і формування запитів на офіційні довідки чи переведення.

Викладачі використовують систему для перевірки робіт, оцінювання, створення тестів і завдань через конструктор. Адміністрація здійснює управління користувачами, безпекою, контентом і логуванням дій у системі.

**Нефункціональні вимоги** до системи охоплюють кілька важливих характеристик, які забезпечують комфортне, стабільне та масштабоване функціонування платформи в умовах реального використання [16].

З точки зору сприйняття, система повинна бути простою в освоєнні як для звичайних користувачів (студентів і викладачів), так і для адміністративного персоналу. Орієнтовний час, необхідний для навчання користувачів, не повинен

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 33   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

перевищувати 3–4 години для студентів і викладачів, і не більше 2 годин для працівників адміністрації. Інтерфейс має бути інтуїтивно зрозумілим, з логічною структурою, підказками та поясненнями, що дозволяє користувачам швидко орієнтуватися в системі.

Час відгуку при виконанні основних дій, таких як перегляд оцінок чи розкладу, має бути мінімальним – не більше 15 секунд. Для складніших запитів, наприклад, при формуванні звітів або завантаженні великих обсягів даних, допустимий час відгуку не повинен перевищувати 30 секунд [17].

Щодо надійності, система має бути постійно доступною для користувачів, з мінімальним часом на технічне обслуговування. Очікувана доступність не повинна опускатися нижче 94% часу. При цьому середній період безвідмовної роботи має складати не менше 15 робочих днів, що гарантує стабільне функціонування без збоїв. Кількість критичних помилок, що впливають на роботу, має бути мінімальною – не більше однієї на кожні 10 000 рядків коду.

Продуктивність системи також є критично важливою. Платформа має бути здатною одночасно обслуговувати щонайменше 8000 активних користувачів, які працюють із загальною базою даних, без зниження швидкості або ефективності обробки запитів. Це забезпечить належний рівень взаємодії в години пікового навантаження.

У плані можливості експлуатації важливою є масштабованість. Система повинна легко адаптуватися до зростання кількості користувачів і обсягу даних, при цьому не втрачаючи продуктивності та залишаючись стабільною.

Забезпечення можливості розширення – як у частині бази даних, так і в частині обчислювальних потужностей – дозволить підтримувати довгострокове функціонування проєкту без необхідності повної переробки технічної архітектури.

Усі ці вимоги створюють надійну основу для побудови системи, яка відповідатиме очікуванням користувачів, витримуватиме високі навантаження та демонструватиме стабільну роботу в рамках повсякденної експлуатації.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 34   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

## 2.2 Визначення варіантів використання та об'єктно-орієнтованої структури

Проектування системи управління академічними ресурсами студентів вимагає детального аналізу функціональних можливостей та організації взаємодії між користувачами і різними компонентами платформи. Одним з основних етапів цього процесу є визначення варіантів використання, які описують можливі сценарії взаємодії користувачів з системою.

Кожен варіант використання є важливим для створення чіткої моделі функціонування системи, що дозволяє забезпечити правильний потік даних, зручність використання та виконання необхідних операцій. Аналіз варіантів використання дає змогу отримати уявлення про те, як система буде функціонувати з точки зору користувача, які процеси необхідно автоматизувати, а також визначити критичні точки для інтеграції з іншими сервісами чи модулями. Завдяки цьому розробники можуть передбачити потенційні проблеми ще на етапі проектування та уникнути зайвого дублювання функцій або суперечливих вимог у майбутньому.

Об'єктно-орієнтоване проектування є основою для подальшого створення структури системи, де кожен об'єкт буде виконувати конкретні функції та взаємодіяти з іншими компонентами [18]. Для побудови такої структури важливо визначити, які класи і об'єкти повинні бути створені, а також як вони взаємодіють один з одним. Саме це забезпечує максимальну відповідність програмної моделі реальним процесам навчального закладу та сприяє ефективній підтримці і масштабуванню системи в майбутньому.

Цей підхід дозволяє реалізувати гнучкість системи, забезпечити її масштабованість та зручність у підтримці, адже кожен об'єкт має чітко визначену роль та відповідальність. Окрім того, структурованість об'єктно-орієнтованої моделі дає змогу швидко додавати новий функціонал або змінювати існуючий без необхідності переписувати значну частину коду, що значно скорочує витрати часу на супровід і розвиток платформи.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 35   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

Для більш наглядного представлення варіантів використання системи управління академічними ресурсами студентів була створена діаграма варіантів використання (рис. 2.1). Ця діаграма відображає взаємодію між користувачами та основними функціями системи, що дозволяє чітко візуалізувати сценарії її роботи та забезпечує прозорість для всіх учасників проєкту.



Рисунок 2.1 – Діаграма варіантів використання системи

Важливість цього етапу полягає ще й у тому, що аналіз сценаріїв дає змогу максимально наблизити систему до реальних потреб користувачів, уникнути надлишкового функціоналу та закласти підґрунтя для зручної, інтуїтивної навігації.

Завершивши аналіз основних сценаріїв взаємодії користувачів із системою, варто зупинитися детальніше на тих, хто безпосередньо бере участь у цих процесах. Саме тому короткий опис акторів і окреслення їхніх основних сфер відповідальності стає логічним продовженням аналізу варіантів використання. Це дає змогу

побудувати цілісну картину майбутньої роботи системи та забезпечити гнучкість у її розширенні або адаптації під специфічні потреби навчального закладу.

Таблиця 2.1

Короткий опис акторів

| Актор                | Опис                                                                                                                                                                                                                                                           |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Студент</b>       | Користувач, який використовує систему для доступу до навчальних матеріалів, перевірки оцінок, виконання завдань та взаємодії з викладачами через. Студенти можуть переглядати свою успішність, проходити тести, завантажувати матеріали та запитувати довідки. |
| <b>Викладач</b>      | Користувач, який відповідає за створення навчального контенту, оцінювання робіт студентів, перевірку завдань і проведення тестів. Викладачі також можуть взаємодіяти з студентами через чат, надаючи зворотний зв'язок та допомогу.                            |
| <b>Адміністратор</b> | Користувач, який управляє акаунтами студентів та викладачів, налаштовує доступи та права користувачів, забезпечує безпеку системи та відповідає за моніторинг роботи платформи.<br><br>Адміністратор також має доступ до налаштувань та керує контентом.       |

У таблиці 2.2 представлені варіанти використання, які описують основні сценарії взаємодії акторів системи з її функціональними можливостями. Кожен варіант відображає конкретну задачу, що виконується певним актором, будь то студент, викладач або адміністратор. Зазначені варіанти використання допомагають чітко розуміти, які операції можуть бути виконані в межах системи та як користувачі будуть взаємодіяти з її компонентами. Зокрема, ці варіанти дозволяють визначити основні процеси, які повинні бути реалізовані в програмному забезпеченні для ефективного управління академічними ресурсами. Таблиця включає всі важливі функціональні можливості. Це дозволяє створити повне уявлення про обсяг задач,

які повинні бути виконані, та забезпечити зручність і ефективність у роботі з системою для всіх типів користувачів.

Таблиця 2.2

Опис варіантів використання

| Варіант використання                 | Актор               | Опис                                                                                                                                 |
|--------------------------------------|---------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| Перегляд успішності студента         | Студент             | Студент переглядає свої оцінки та поточну успішність з усіх дисциплін.                                                               |
| Календар запланованих робіт          | Студент             | Студент отримує календар з струкутованим поданням усіх навчальних задач та їх кінцевим терміном подання роботи, а також тип завдання |
| Відновлення паролю                   | Студент<br>Викладач | Студент/Викладач ініціює процес відновлення паролю через електронну пошту.                                                           |
| Завантаження навчальних матеріалів   | Студент             | Студент завантажує матеріали з бібліотеки або свої власні на платформу.                                                              |
| Виконання ЛР, ПР, КР                 | Студент             | Студент використовує інтерфейс для виконання лабораторних, практичних та курсових робіт.                                             |
| Система спостереження під час тестів | Студент             | Під час виконання тесту система, моніторить вкладки.                                                                                 |
| Підтримка зв'язку викладачем         | Студент<br>Викладач | Студент та викладач мають можливість залишати коментарі до виконаних робіт                                                           |
| Перевірка робіт                      | Викладач            | Викладач переглядає виконані студентами роботи, оцінює їх та визначає їх статус (зараховано, не зараховано, доопрацювання).          |
| Створення тестів та завдань          | Викладач            | Викладач створює тести та завдання для студентів за допомогою конструктора тестів.                                                   |
| Управління користувачами             | Адміністратор       | Адміністратор додає, редагує або видаляє користувачів з системи.                                                                     |
| Контроль доступу та безпеки          | Адміністратор       | Адміністратор налаштовує доступи до різних розділів системи та контролює безпеку.                                                    |
| Управління контентом                 | Адміністратор       | Адміністратор керує контентом системи. Має можливість створювати нові кафедри, групи, курси та факультети.                           |

|                                                         |               |                                                                                                                   |
|---------------------------------------------------------|---------------|-------------------------------------------------------------------------------------------------------------------|
| Автоматичне формування довідок                          | Адміністратор | Адміністратор перевіряє та погоджує автоматично згенеровані довідки про успішність студентів та інші документи.   |
| Запити на переведення між групами та академічну перерву | Адміністратор | Адміністратор обробляє запити студентів на переведення між групами, академічні відпустки або перерви.             |
| Гнучкий графік навчання                                 | Студент       | Студент має можливість обирати зручний графік для виконання навчальних завдань, лабораторних та практичних робіт. |

Після визначення основних ролей користувачів і їхніх сценаріїв взаємодії з платформою логічним кроком у проектуванні інформаційної системи стає побудова діаграми класів. Цей інструмент об'єктно-орієнтованого моделювання дозволяє наочно представити всі ключові сутності системи, їх атрибути, методи та зв'язки між собою. Саме діаграма класів слугує картою для подальшої реалізації, оскільки фіксує не лише те, які дані потрібно зберігати, а й яким чином вони об'єднуються в єдину логічну структуру.

Деталізуючи діаграму класів (рис. 2.2), вдається розбити систему на окремі компоненти, що відповідають за конкретні функції чи частини бізнес-логіки. Наприклад, у системі управління академічними ресурсами обов'язково відображаються такі основні класи, як «Користувач», «Студент», «Викладач», «Адміністратор», «Курс», «Модуль», «Заняття», «Оцінка», «Завдання», а також різноманітні зв'язки між ними – від наслідування до асоціацій і композицій.

Важливо підкреслити, що продумана діаграма класів дає змогу мінімізувати дублювання коду, забезпечити гнучкість при розвитку платформи, а також полегшує тестування та підтримку системи.

Вона виступає посередником між аналізом вимог і безпосередньою реалізацією програмної логіки, дозволяючи кожному члену команди розробників бачити цілісну картину архітектури та швидко орієнтуватися в структурі проекту.

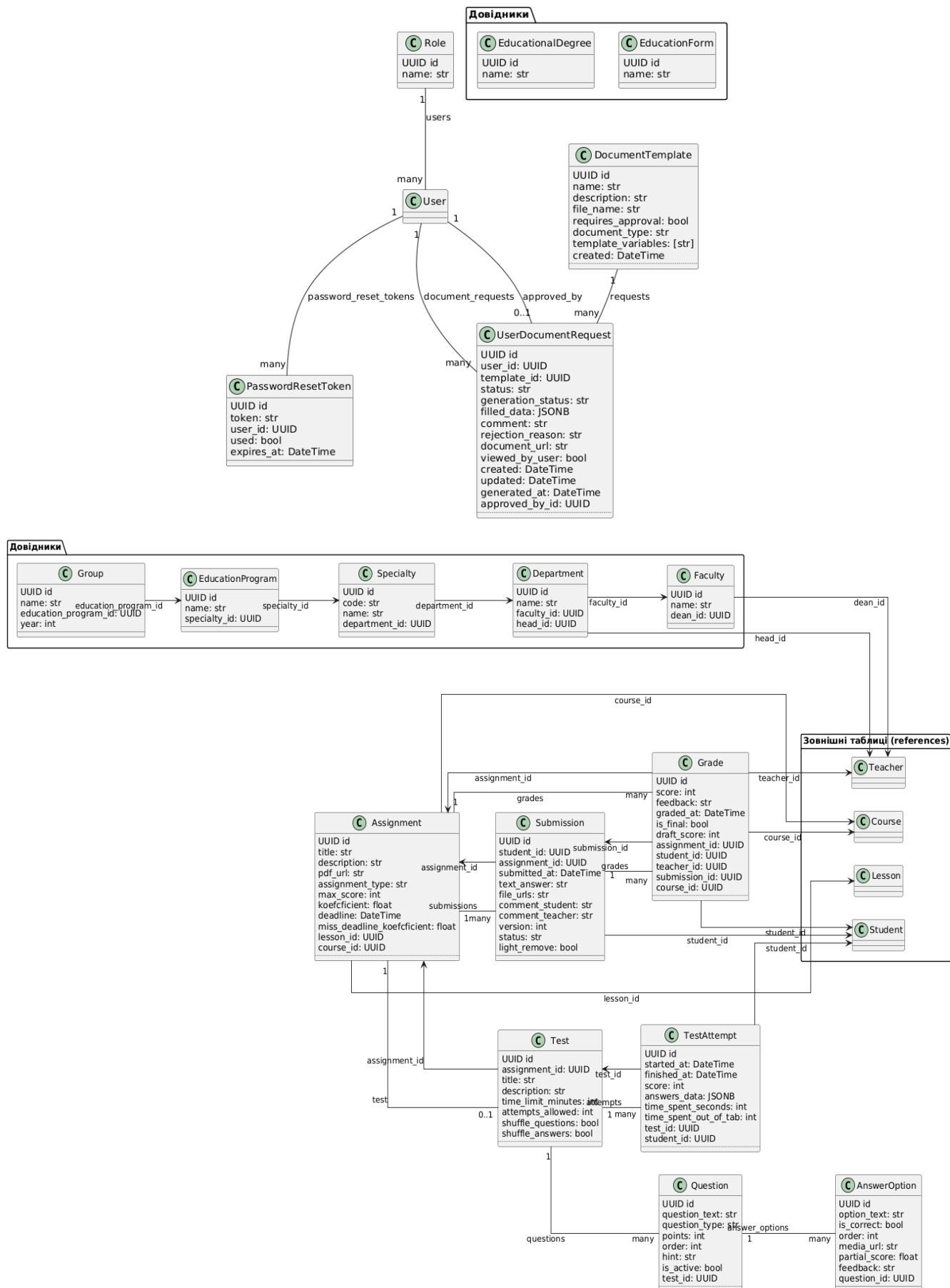


Рис. 2.2 – Діаграма класів

## 2.3 Розробка бази даних системи

Проектування бази даних є одним із ключових етапів у створенні системи управління академічними ресурсами студентів, оскільки саме вона забезпечує збереження, цілісність та доступність даних, необхідних для функціонування системи.

У контексті розробки даної інформаційної системи, база даних має підтримувати широкий спектр функцій – від обліку академічної успішності студентів до автоматизації документообігу, комунікації між користувачами та контролю адміністративних процесів.

Основою структури є реляційна модель даних, яка реалізована за допомогою системи управління базами даних PostgreSQL [19]. Вибір саме цього СУБД обумовлений її високою надійністю, масштабованістю, широким набором інструментів для забезпечення транзакційної цілісності та підтримкою складних зв'язків між сутностями.

Таблиця roles містить перелік ролей, які можуть мати користувачі системи. Вона служить довідником для розмежування прав доступу та функціональності між різними типами користувачів – наприклад, студентами, викладачами, адміністраторами тощо.

Кожен запис містить унікальний ідентифікатор та назву ролі. Таблиця має зв'язок один-до-багатьох із таблицею users, завдяки чому одному типу ролі може відповідати багато користувачів.

Таблиця 2.3

Опис таблиці Roles

| Поле | Тип даних | Опис                                                     |
|------|-----------|----------------------------------------------------------|
| id   | UUID      | Унікальний ідентифікатор ролі                            |
| name | String    | Назва ролі (наприклад: студент, викладач, адміністратор) |

Таблиця password\_reset\_tokens відповідає за зберігання токенів для скидання пароля користувачів. Кожен запис у таблиці містить унікальний ідентифікатор, текстовий токен, що є унікальним, і посилання на користувача, якому цей токен

належить. Також передбачені поля для збереження інформації про те, чи був токен використаний, та дату і час його завершення дії. Завдяки зв'язку з таблицею користувачів через зовнішній ключ `user_id`, система може автоматично видаляти токени при видаленні користувача, що забезпечується каскадним видаленням.

Таблиця 2.4

Опис таблиці `password_reset_tokens`

| Поле                    | Тип даних | Опис                                               |
|-------------------------|-----------|----------------------------------------------------|
| <code>id</code>         | UUID      | Унікальний ідентифікатор токена                    |
| <code>token</code>      | Text      | Унікальний токен для скидання пароля               |
| <code>user_id</code>    | UUID      | Посилання на користувача, для якого створено токен |
| <code>used</code>       | Boolean   | Чи був токен використаний                          |
| <code>expires_at</code> | DateTime  | Дата та час завершення дії токена                  |

Таблиця `faculties` зберігає інформацію про факультети, які є ключовими одиницями академічної структури університету. Кожен факультет має унікальну назву та ідентифікатор, а також може бути закріплений за певним викладачем, який виконує роль декана. Це реалізується через зовнішній ключ `dean_id`, що посилається на таблицю `teachers`, де зберігається інформація про всіх викладачів. У разі видалення декана значення поля автоматично обнуляється, завдяки каскадній поведінці `SET NULL`. Один факультет може містити декілька кафедр, що реалізовано через зв'язок один-до-багатьох з таблицею `departments`.

Таблиця 2.5

Опис таблиці `faculties`

| Поле                 | Тип даних | Опис                                                          |
|----------------------|-----------|---------------------------------------------------------------|
| <code>id</code>      | UUID      | Унікальний ідентифікатор факультету                           |
| <code>name</code>    | String    | Назва факультету (унікальна)                                  |
| <code>dean_id</code> | UUID      | Посилання на користувача-викладача, який є деканом факультету |

Таблиця document\_templates призначена для зберігання шаблонів документів, які можуть бути використані для автоматизованого формування довідок та інших документів у системі. Кожен шаблон має унікальний ідентифікатор, назву, опис та ім'я файлу, яке вказує на місце розташування файлу шаблону на сервері. Також визначено, чи потребує шаблон попереднього погодження, тип документа (наприклад, сертифікат) та список змінних, які мають бути підставлені у шаблон. Зберігається дата створення шаблону, що дозволяє відслідковувати хронологію змін.

Таблиця 2.6

Опис таблиці document\_templates

| Поле               | Тип даних     | Опис                                                                |
|--------------------|---------------|---------------------------------------------------------------------|
| id                 | UUID          | Унікальний ідентифікатор шаблону документа                          |
| name               | String        | Назва шаблону                                                       |
| description        | Text          | Опис шаблону (необов'язковий)                                       |
| file_name          | String        | Шлях до шаблону документа на сервері                                |
| requires_approval  | Boolean       | Ознака, чи потребує документ погодження                             |
| document_type      | String        | Тип документа (наприклад, "certificate")                            |
| template_variables | ARRAY(String) | Змінні шаблону, які потрібно підставити під час генерації документа |
| created            | DateTime      | Дата створення шаблону                                              |

Таблиця document\_requests використовується для обробки запитів користувачів на створення документів. Вона містить посилання на користувача, який подав запит, а також на відповідний шаблон документа. Стан обробки запиту зберігається у вигляді статусу з обмеженого переліку (очікує розгляду, в обробці, сформовано, погоджено, відхилено, помилка). Додатково зберігається статус генерації документа, коментарі, причина відхилення, підставлені дані у вигляді JSON, а також URL згенерованого файлу, якщо він вже створений. Поле viewed\_by\_user дозволяє

фіксувати, чи переглянув користувач результат. Таблиця також містить часові мітки створення, оновлення та фактичної генерації документа. Якщо документ був погоджений іншим користувачем, ця інформація також зберігається у відповідному полі з посиланням на користувача, який погодив запит.

Таблиця 2.7

Опис таблиці document\_requests

| Поле              | Тип даних | Опис                                                                  |
|-------------------|-----------|-----------------------------------------------------------------------|
| id                | UUID      | Унікальний ідентифікатор запиту                                       |
| user_id           | UUID      | Посилання на користувача, який подав запит                            |
| template_id       | UUID      | Посилання на шаблон документа                                         |
| status            | Enum      | Статус запиту (pending, processing, generated, approved, rejected...) |
| generation_status | String    | Стан генерації документа (наприклад, "pending")                       |
| filled_data       | JSONB     | JSON-дані, підставлені в шаблон                                       |
| comment           | Text      | Коментар користувача або адміністратора                               |
| rejection_reason  | Text      | Причина відмови (якщо запит відхилено)                                |
| document_url      | String    | Посилання на згенерований документ                                    |
| viewed_by_user    | Boolean   | Чи переглянув користувач документ                                     |
| created           | DateTime  | Дата створення запиту                                                 |
| updated           | DateTime  | Дата останнього оновлення                                             |
| generated_at      | DateTime  | Дата генерації документа                                              |
| approved_by_id    | UUID      | Посилання на користувача, який погодив запит                          |

Таблиця departments відповідає за зберігання даних про кафедри, які структурно підпорядковуються факультетам. Кожна кафедра має унікальну назву, ідентифікатор, а також зовнішній ключ faculty\_id, що пов'язує її з відповідним факультетом. Крім цього, кафедра може мати завідувача, інформація про якого зберігається у полі head\_id, що посилається на користувача з таблиці teachers. Така

структура дозволяє гнучко формувати академічну ієрархію та динамічно змінювати керівників кафедр і факультетів.

Таблиця 2.8

Опис таблиці departments

| Поле       | Тип даних | Опис                                               |
|------------|-----------|----------------------------------------------------|
| id         | UUID      | Унікальний ідентифікатор кафедри                   |
| name       | String    | Назва кафедри (унікальна)                          |
| faculty_id | UUID      | Посилання на факультет, до якого належить кафедра  |
| head_id    | UUID      | Посилання на викладача, який є завідувачем кафедри |

Таблиця educational\_degrees зберігає список можливих освітніх ступенів, які може здобувати студент (наприклад, бакалавр, магістр, доктор філософії тощо). Кожен запис містить унікальний ідентифікатор та назву ступеня. Ця таблиця виконує роль довідника для інших сутностей у системі.

Таблиця 2.9

Опис таблиці educational\_degrees

| Поле | Тип даних | Опис                             |
|------|-----------|----------------------------------|
| id   | UUID      | Унікальний ідентифікатор ступеня |
| name | String    | Назва освітнього ступеня         |

Таблиця education\_forms містить перелік форм навчання, які доступні в закладі (наприклад, денна, заочна, вечірня тощо). Вона слугує довідником для зберігання уніфікованих типів освітніх форматів у системі.

Таблиця 2.10

Опис таблиці education\_forms

| Поле | Тип даних | Опис                           |
|------|-----------|--------------------------------|
| id   | UUID      | Унікальний ідентифікатор форми |
| name | String    | Назва форми навчання           |

Таблиця specialties призначена для зберігання інформації про навчальні спеціальності, які реалізуються в межах кафедр. Кожна спеціальність має унікальний код, назву та зовнішній ключ на кафедрі, яка відповідає за реалізацію цієї спеціальності. Один запис у таблиці specialties може бути пов'язаний з кількома освітніми програмами.

Таблиця 2.11

Опис таблиці specialties

| Поле          | Тип даних | Опис                                             |
|---------------|-----------|--------------------------------------------------|
| id            | UUID      | Унікальний ідентифікатор спеціальності           |
| code          | String    | Код спеціальності (може бути порожнім)           |
| name          | String    | Назва спеціальності                              |
| department_id | UUID      | Посилання на кафедру, яка реалізує спеціальність |

Таблиця education\_programs зберігає інформацію про конкретні освітні програми, які реалізуються в межах певної спеціальності. Кожна програма має унікальну назву та зовнішній ключ specialty\_id, який вказує на відповідну спеціальність. До освітньої програми може належати декілька академічних груп і навчальних курсів.

Таблиця 2.12

Опис таблиці education\_programs

| Поле         | Тип даних | Опис                                                  |
|--------------|-----------|-------------------------------------------------------|
| id           | UUID      | Унікальний ідентифікатор освітньої програми           |
| name         | String    | Назва освітньої програми                              |
| specialty_id | UUID      | Посилання на спеціальність, до якої належить програма |

Таблиця groups зберігає академічні групи, які формуються в рамках певної освітньої програми. Кожна група має унікальну назву, посилання на відповідну освітню програму, а також поле year, яке зберігає рік набору або навчання.

Таблиця 2.13

Опис таблиці groups

| Поле                 | Тип даних | Опис                                                       |
|----------------------|-----------|------------------------------------------------------------|
| id                   | UUID      | Унікальний ідентифікатор групи                             |
| name                 | String    | Назва академічної групи (унікальна)                        |
| education_program_id | UUID      | Посилання на освітню програму, до якої належить група      |
| year                 | Integer   | Рік навчання або рік формування групи (необов'язкове поле) |

Таблиця assignments містить усі навчальні завдання, що формуються в межах занять або курсів. Кожне завдання має назву, опис, тип (наприклад, тест, проект, домашнє завдання), максимальний бал і ваговий коефіцієнт. Також враховується дедлайн і коефіцієнт запізнення, який впливає на підсумкову оцінку у випадку несвоєчасної здачі. Додаткові поля дозволяють вказати тип подачі (текст, файл, або комбінація), кількість дозволених перездач, критерії оцінювання, порядок відображення у занятті, обов'язковість виконання, а також прив'язку до конкретного заняття (lesson\_id) або курсу (course\_id). Завдання можуть бути пов'язані з тестами та мати відношення до оцінок і подань студентів. Таблиця також передбачає логіку “м'якого видалення” для збереження історії.

Таблиця 2.14

Опис таблиці education\_programs

| Поле        | Тип даних | Опис                                            |
|-------------|-----------|-------------------------------------------------|
| id          | UUID      | Унікальний ідентифікатор завдання               |
| title       | String    | Назва завдання                                  |
| description | Text      | Опис завдання                                   |
| pdf_url     | String    | Посилання на PDF-документ або файл із завданням |

|                           |          |                                                           |
|---------------------------|----------|-----------------------------------------------------------|
| assignment_type           | String   | Тип завдання (тест, есе, домашня робота тощо)             |
| max_score                 | Integer  | Максимальна кількість балів за завдання                   |
| koefcfcient               | Float    | Коефіцієнт ваги завдання у підсумковій оцінці             |
| deadline                  | DateTime | Дата та час дедлайну                                      |
| miss_deadline_koefcfcient | Float    | Коефіцієнт, який застосовується у разі запізнення         |
| is_required               | Boolean  | Чи є завдання обов'язковим до виконання                   |
| order                     | Integer  | Порядок відображення завдання у занятті                   |
| submission_type           | String   | Формат подання завдання (текст, файл, обидва тощо)        |
| light_remove              | Boolean  | Ознака м'якого видалення завдання                         |
| resubmission_limit        | Integer  | Максимальна кількість дозволених перездач                 |
| grading_criteria          | Text     | Критерії оцінювання                                       |
| is_shown                  | Boolean  | Чи відображається завдання у системі для студентів        |
| lesson_id                 | UUID     | Посилання на заняття (може бути порожнім)                 |
| course_id                 | UUID     | Посилання на курс (якщо завдання стосується всього курсу) |

Таблиця submissions фіксує відповіді студентів на завдання. Кожне подання містить ідентифікатор студента та завдання, дату подання, текстову відповідь, прикріплені файли (у вигляді JSON), коментарі від студента та викладача, статус

(подано, оцінено, повернуто тощо), номер версії відповіді та ознаку м'якого видалення. Це дає змогу відстежувати історію подач і підтримувати багатоверсійність відповідей.

Таблиця 2.15

Опис таблиці

| Поле            | Тип даних | Опис                                                         |
|-----------------|-----------|--------------------------------------------------------------|
| id              | UUID      | Унікальний ідентифікатор відповіді                           |
| student_id      | UUID      | Посилання на студента, який подав завдання                   |
| assignment_id   | UUID      | Посилання на завдання                                        |
| submitted_at    | DateTime  | Дата та час подання відповіді                                |
| text_answer     | Text      | Текстова відповідь                                           |
| file_urls       | Text      | JSON зі списком файлів (URL)                                 |
| comment_student | Text      | Коментар студента                                            |
| comment_teacher | Text      | Коментар викладача                                           |
| version         | Integer   | Версія поданої відповіді                                     |
| status          | String    | Статус відповіді (submitted, graded, returned, pending тощо) |
| light_remove    | Boolean   | Ознака м'якого видалення                                     |

Таблиця tests відіграє ключову роль у структурі оцінювання, оскільки дозволяє формувати індивідуальні та масові перевірки знань студентів у межах курсу. Вона тісно пов'язана із таблицями питань, варіантів відповідей і результатів, що забезпечує повну інтеграцію процесу тестування в освітній процес. Така гнучка структура дозволяє реалізовувати як прості тести для самоперевірки, так і складні контрольні роботи з обмеженнями та автоматичною перевіркою. Тести можуть бути адаптовані під різні сценарії, з урахуванням обмежень часу, кількості спроб і способів відображення результатів. Налаштування дозволяють викладачу контролювати поведінку користувача під час тесту, наприклад, обмежити перемикання вкладок або приховати правильні відповіді після завершення. Завдяки цьому

система забезпечує високий рівень академічної доброчесності та зручність в адмініструванні тестування.

Таблиця 2.16

Опис таблиці tests

| Поле                 | Тип даних | Опис                                                           |
|----------------------|-----------|----------------------------------------------------------------|
| id                   | UUID      | Унікальний ідентифікатор тесту                                 |
| assignment_id        | UUID      | Посилання на завдання, до якого належить тест                  |
| title                | String    | Назва тесту                                                    |
| description          | Text      | Опис тесту                                                     |
| time_limit_minutes   | Integer   | Обмеження часу в хвилинах (null – без обмежень)                |
| attempts_allowed     | Integer   | Кількість дозволених спроб (null – необмежено)                 |
| shuffle_questions    | Boolean   | Чи перемішувати питання                                        |
| shuffle_answers      | Boolean   | Чи перемішувати відповіді                                      |
| show_correct_answers | Boolean   | Чи показувати правильні відповіді                              |
| show_score_after     | String    | Коли показувати результат ("never", "always", "after_attempt") |
| show_answers_after   | String    | Коли показувати відповіді ("never", "always", "after_attempt") |
| browser_restrictions | Boolean   | Заборона перемикавання вкладок                                 |
| result_view_type     | String    | Тип перегляду результату ("detailed", "brief", "none")         |
| schedule_type        | String    | Графік проходження ("flexible", "fixed_time")                  |
| available_from       | DateTime  | Дата початку доступності тесту                                 |
| available_to         | DateTime  | Дата завершення доступності тесту                              |

Таблиця test\_attempts зберігає дані про кожну спробу студента пройти тест. Фіксуються час початку і завершення, кількість набраних балів, відповіді студента у форматі JSON, тривалість проходження, активність за межами вкладки, статус спроби та відмітка про підозрілу поведінку. Передбачена можливість ручної перевірки та залишення нотаток для рецензування.

Таблиця 2.17

Опис таблиці test\_attempts

| Поле                  | Тип даних | Опис                                                    |
|-----------------------|-----------|---------------------------------------------------------|
| id                    | UUID      | Унікальний ідентифікатор спроби                         |
| started_at            | DateTime  | Дата й час початку тесту                                |
| finished_at           | DateTime  | Дата завершення (може бути null, якщо не завершено)     |
| score                 | Integer   | Кількість набраних балів                                |
| answers_data          | JSONB     | Відповіді студента у форматі JSON                       |
| time_spent_seconds    | Integer   | Час, витрачений на проходження тесту                    |
| time_spent_out_of_tab | Integer   | Час, проведений поза вкладкою тесту                     |
| status                | String    | Статус спроби ("in_progress", "completed", "abandoned") |
| suspicious_activity   | Boolean   | Ознака підозрілої поведінки                             |
| manual_review         | Boolean   | Чи потребує ручної перевірки                            |
| review_notes          | Text      | Примітки для рецензента                                 |
| questions_order       | Text      | Послідовність питань у спробі                           |
| test_id               | UUID      | Посилання на тест                                       |
| student_id            | UUID      | Посилання на студента                                   |

Таблиця test\_questions містить усі питання, що входять до тестів. Кожне питання має текст, тип (один правильний варіант, декілька, текстове, на відповідність

тощо), кількість балів, порядок у тесті, пояснення після відповіді, медіафайли, підказки, складність та статус активності.

Таблиця 2.18

Опис таблиці test\_questions

| Поле          | Тип даних | Опис                                                          |
|---------------|-----------|---------------------------------------------------------------|
| id            | UUID      | Унікальний ідентифікатор питання                              |
| question_text | Text      | Текст питання                                                 |
| question_type | String    | Тип питання ("single_choice", "multiple_choice", "text" тощо) |
| points        | Integer   | Кількість балів за правильну відповідь                        |
| order         | Integer   | Порядковий номер у тесті                                      |
| explanation   | Text      | Пояснення, що з'являється після відповіді                     |
| media_url     | String    | Посилання на медіа (зображення, відео, аудіо)                 |
| media_type    | String    | Тип медіа ("image", "video", "audio")                         |
| difficulty    | String    | Рівень складності ("easy", "medium", "hard")                  |
| hint          | Text      | Підказка до питання                                           |
| is_active     | Boolean   | Чи є питання активним у тесті                                 |
| test_id       | UUID      | Посилання на тест, до якого належить питання                  |

Таблиця answer\_options містить варіанти відповідей для кожного питання. Кожен варіант має текст, позначку правильності, порядок, можливий медіафайл, часткову оцінку у випадку частково правильних відповідей та зворотний зв'язок для студента. Це дозволяє реалізувати як одно-, так і багатоваріантні тести з гнучкою системою оцінювання. Часткове нарахування балів стимулює студентів до уважного аналізу завдань і мінімізує втрату балів при незначних помилках. Додаткові пояснення до відповідей сприяють глибшому засвоєнню матеріалу та формуванню зворотного зв'язку. Завдяки структурованому зберіганню варіантів

відповідей система забезпечує автоматичну перевірку результатів та підвищує ефективність процесу оцінювання.

Таблиця 2.19

Опис таблиці answer\_options

| Поле          | Тип даних | Опис                                                      |
|---------------|-----------|-----------------------------------------------------------|
| id            | UUID      | Унікальний ідентифікатор варіанту відповіді               |
| option_text   | Text      | Текст варіанту відповіді                                  |
| is_correct    | Boolean   | Ознака правильності відповіді                             |
| order         | Integer   | Порядок варіанту в межах питання                          |
| media_url     | String    | Посилання на медіафайл                                    |
| partial_score | Float     | Часткова оцінка у випадку частково правильних варіантів   |
| feedback      | Text      | Зворотній зв'язок у випадку вибору цього варіанту         |
| question_id   | UUID      | Посилання на питання, до якого належить варіант відповіді |

## 2.4 Проєктування та реалізація алгоритмів роботи системи

Проєктування та реалізація алгоритмів системи управління академічними ресурсами студентів відбувалися з урахуванням принципів гнучкої архітектури та розділення функціональних блоків. В основі серверної частини лежить вебфреймворк FastAPI (рис. 2.3), який поєднує у собі високу продуктивність та зручний синтаксис. Завдяки вбудованій підтримці асинхронного виконання, система забезпечує швидку обробку запитів та стабільну роботу при одночасній взаємодії з великою кількістю користувачів. Цей підхід дозволяє забезпечити масштабованість і адаптивність системи до зростаючих вимог користувачів та навантаження. Крім того, використання FastAPI спрощує документування API та інтеграцію з фронтомом завдяки автоматичній генерації OpenAPI-специфікацій.



Рис. 2.3 – Схема роботи FastAPI

Весь застосунок структуровано за логікою окремих компонентів, кожен з яких відповідає за конкретний аспект роботи системи. На початку були визначені основні сутності, які відображають академічні об'єкти: курси, модулі, заняття, студенти, викладачі, завдання, тести, спроби проходження та оцінки. Для кожної з них створено модель, яка відповідає структурі бази даних. Ці моделі реалізовані через ORM-бібліотеку SQLAlchemy, що дозволяє працювати з базою даних, використовуючи об'єктно-орієнтований підхід [20]. У моделях описано зв'язки між сутностями, обмеження цілісності та специфічну поведінку об'єктів.

Щоб обробляти зовнішні запити та забезпечити коректну передачу даних між клієнтом і сервером, створено схеми з використанням Pydantic [21]. Pydantic-схеми (рис. 2.4) - це класи, які використовуються в системі для визначення структури, типів та правил валідації даних, що передаються між клієнтом і сервером. Вони базуються на бібліотеці Pydantic, яка є стандартним інструментом у FastAPI для роботи з типізованими моделями [22].

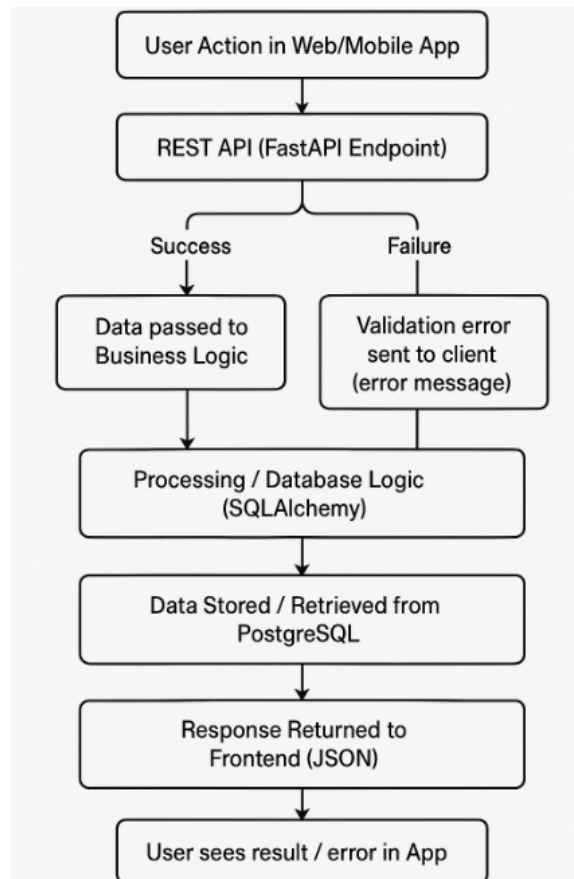


Рис. 2.4 – Схема роботи Pydantic

Основна ідея полягає в тому, щоб забезпечити сувору перевірку вхідних та вихідних даних, автоматично обробляти перетворення типів і генерувати документацію до API без зайвих зусиль. Кожна Pydantic-схема описує набір полів із зазначенням їх типів (наприклад, рядок, ціле число, дата, список, UUID тощо), обов'язковість поля, а також може містити додаткові правила перевірки. Схеми розділяються за контекстом використання: деякі застосовуються лише для запитів (наприклад, створення нового об'єкта), інші - для відповідей (відображення об'єктів клієнту), а також можуть бути комбіновані у базові або оновлювані форми. Такий підхід дозволяє уникнути дублювання коду та спростити підтримку системи. Крім того, Pydantic-схеми автоматично інтегруються у Swagger-документацію FastAPI (рис. 2.5), що дає змогу розробникам і користувачам бачити усі типи вхідних та вихідних даних прямо в інтерактивному інтерфейсі. Це значно полегшує тестування та використання API. Завдяки цьому, процес розробки стає прозорішим, а взаємодія між фронтендом і бекендом – більш передбачуваною та контрольованою.

The screenshot displays the Swagger UI interface for an API. It lists four endpoints:

- POST** `/api/v1/course/{course_id}/enroll/{student_id}` Enroll Student In Course
- POST** `/api/v1/course/{course_id}/assign_teacher/{teacher_id}` Assign Teacher To Course
- DELETE** `/api/v1/course/{course_id}/unenroll_teacher/{teacher_id}` Unenroll Teacher From Course
- DELETE** `/api/v1/course/{course_id}/unenroll_student/{student_id}` Unenroll Student From Course

The selected endpoint, **DELETE** `/api/v1/course/{course_id}/unenroll_student/{student_id}`, is expanded to show its parameters:

| Name                                                     | Description                          |
|----------------------------------------------------------|--------------------------------------|
| <b>course_id</b> * required<br>string(\$uuid)<br>(path)  | dda6c420-d23d-430e-b8f2-1a3b95889da7 |
| <b>student_id</b> * required<br>string(\$uuid)<br>(path) | b5dd815f-17c0-444c-8ffc-df51f753d54e |

Below the parameters, there is an **Execute** button and a **Clear** button. The **Responses** section is currently empty. The **curl** section shows the following command:

```
curl -X 'DELETE' \
  'http://localhost/api/v1/course/dda6c420-d23d-430e-b8f2-1a3b95889da7/unenroll_student/b5dd815f-17c0-444c-8ffc-df51f753d54e' \
  -H 'accept: application/json'
```

The **Request URL** section shows the full URL: `http://localhost/api/v1/course/dda6c420-d23d-430e-b8f2-1a3b95889da7/unenroll_student/b5dd815f-17c0-444c-8ffc-df51f753d54e`. The **Server response** section is currently empty.

Рис. 2.5 - Swagger-документація

Основна бізнес-логіка розміщена у модулі CRUD. Тут реалізовані всі необхідні дії з базою даних: створення, читання, оновлення та видалення записів. Для кожної моделі є окремий клас, який містить як стандартні операції, так і специфічні запити, наприклад, отримання студентів певної навчальної програми, фільтрація за статусом спроби або автоматичне оцінювання тестів. Такий підхід дозволяє ізолювати логіку роботи з даними, спрощуючи тестування та подальше розширення системи.

Усі функції, доступні користувачам через HTTP-запити, реалізовані у вигляді API-ендпоінтів [23]. Вони згруповані за логікою у маршрутизаторах: окремо для користувачів, курсів, уроків, тестів, оцінок. Кожен маршрутизатор відповідає за свій блок функціоналу, а також включає механізми авторизації, перевірки прав доступу та обробки виключень. Це дозволяє чітко контролювати, хто і які дії може виконувати у системі.

Окрім основних функціональних частин, у системі передбачені допоміжні модулі. Наприклад, у core зберігаються конфігурації, змінні середовища, налаштування підключення до бази даних та параметри безпеки. У utils містяться

універсальні функції, які використовуються в різних частинах проєкту, а middleware забезпечують обробку запитів до їх надходження до маршрутизаторів, наприклад, логування або перевірку токенів.

Уся реалізація побудована з орієнтацією на подальше масштабування та розширення. Вже зараз система дозволяє студентам отримувати завдання, проходити тести, переглядати оцінки, а викладачам – створювати курси, перевіряти роботи та виставляти бали. Завдяки чітко сформованій структурі, у майбутньому можливе швидке додавання нового функціоналу, наприклад, генерації аналітики, інтеграції зі сторонніми сервісами або реалізації адаптивного навчання.

Окрему увагу при проєктуванні системи було приділено безпеці та захисту персональних даних користувачів. Одним з важливих аспектів у цьому напрямку є механізм автентифікації та відновлення доступу до облікового запису. Зокрема, реалізовано функціонал безпечного скидання пароля (рис 2.6), який дозволяє студентам і викладачам швидко відновити доступ у разі його втрати без ризику для конфіденційної інформації.

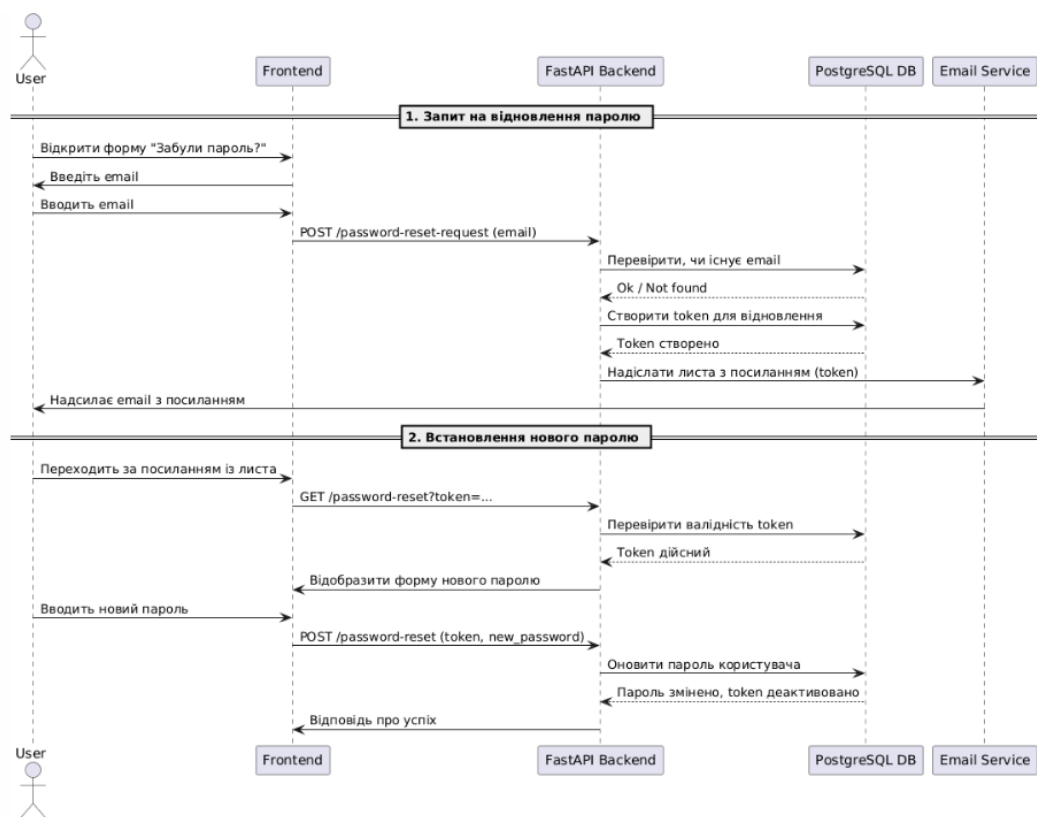


Рис. 2.6 - Діаграма послідовності відновлення паролю

Механізм скидання пароля в системі реалізований у два етапи – ініціація запити на скидання та безпосереднє встановлення нового пароля. Його головна мета – дозволити користувачу безпечно відновити доступ до облікового запису у випадку, якщо він втратив або забув свій пароль.

На першому етапі користувач надсилає запит із вказаною електронною поштою. Система перевіряє, чи існує обліковий запис з таким email, і якщо так – створює тимчасовий токен для відновлення пароля. Цей токен містить зашифровану інформацію про користувача, термін дії та тип операції. Далі формується спеціальне посилання для скидання пароля, яке включає цей токен. Це посилання надсилається користувачу на електронну пошту. Якщо такого користувача не існує, система все одно повертає нейтральне повідомлення, щоб уникнути витоку інформації – таким чином жоден зломисник не зможе перевірити, чи зареєстровано певну адресу у системі.

На другому етапі, коли користувач переходить за посиланням і вводить новий пароль, система перевіряє переданий токен. Якщо токен дійсний, ще не використаний і не прострочений – пароль оновлюється. Також перевіряється, чи новий пароль і його підтвердження збігаються. Після успішного скидання пароля токен позначається як використаний, а всі попередні сесії користувача анулюються – це гарантує, що старі пристрої або браузері більше не матимуть доступу до облікового запису.

Таким чином, весь процес скидання пароля захищений, передбачає перевірку надійності токена, дотримання терміну дії, однократне використання та додаткову безпеку через завершення активних сесій. Це дозволяє гарантувати, що доступ буде надано виключно користувачу, який контролює електронну пошту, пов’язану з акаунтом.

У системі управління академічними ресурсами студентів важливе місце займає модуль подання документів (рис. 2.7), який дає змогу студентам надсилати заявки на отримання довідок, характеристик, залікових книжок та інших офіційних документів. Його реалізація базується на принципах зручності користування,

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 58   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

автоматизації обробки та прозорості кожного етапу – від формування заявки до фінального завантаження готового документа.

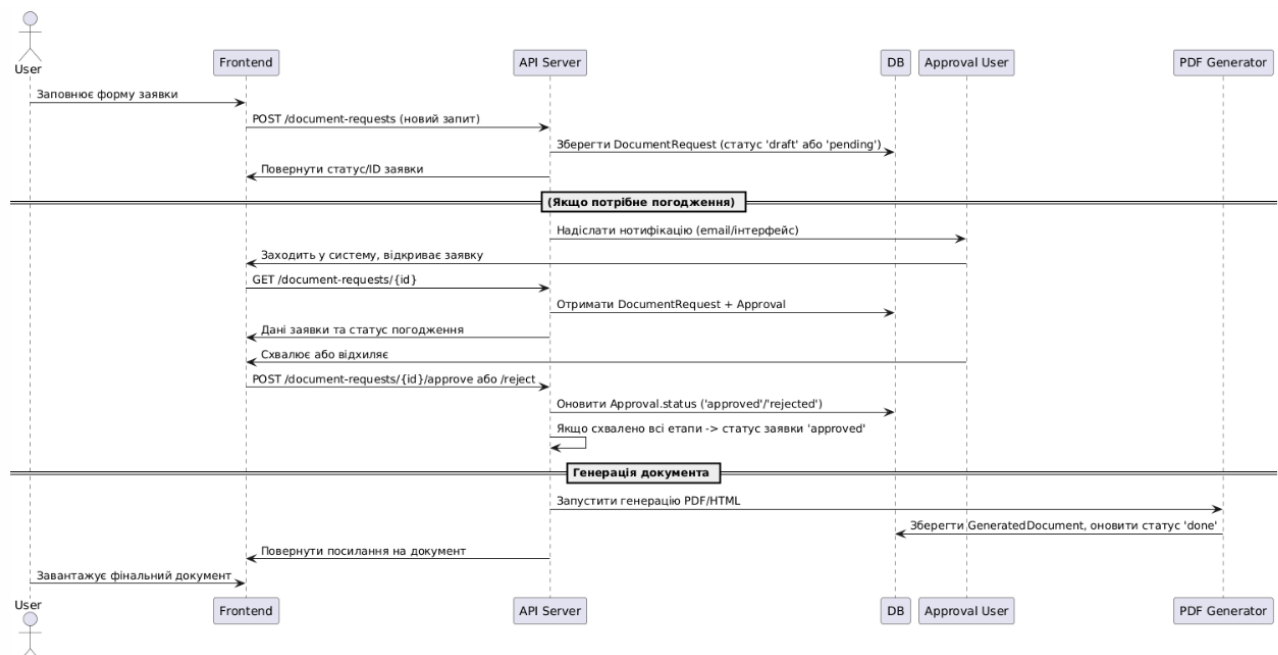


Рис. 2.7 - Діаграма послідовності документообігу

Логіка роботи модуля побудована так, щоб максимальне навантаження було перенесене на бекенд, а інтерфейс залишався простим і зрозумілим для користувача.

Ключова роль у цьому модулі відведена CRUD-класу, який інкапсулює всю бізнес-логіку: від перевірки, чи вже існує запит на той самий шаблон (щоб уникнути дублювання), до рендерингу документа на основі динамічних даних. При створенні нового запиту метод перевіряє, чи є вже активна заявка для поточного користувача на цей шаблон із незавершеним статусом. Якщо така існує – система не дозволяє створити нову, натомість повертає інформативне повідомлення про очікування розгляду.

У разі відсутності активної заявки система створює новий запис у таблиці запитів. Важливо, що під час створення одразу завантажується модель шаблону, до якої прив'язаний запит. Сам шаблон зчитується з файлової системи – у форматі HTML з маркованими змінними, які потім заповнюються переданими у запиті даними за допомогою механізму вкладених словників за допомогою шаблонізатору Jinja2.

Після цього обробка поділяється на два варіанти: якщо шаблон не вимагає погодження – HTML одразу конвертується у PDF через бібліотеку WeasyPrint і повертається як StreamingResponse. Якщо ж погодження обов’язкове, PDF-файл ще не створюється – заявка отримує статус pending, і лише після ручного затвердження адміністратором запускається окремий метод генерації PDF, де HTML уже є попередньо збереженим або переданим повторно. Після генерації файл зберігається на сервері у заздалегідь підготовленому каталозі зі структурованою назвою, яка прив’язується до заявки. Цей шлях записується в поле document\_url, що дозволяє пізніше завантажити документ за допомогою окремого ендпоінта.

Механізм також підтримує попередній перегляд – маршрутизатор /preview-html дозволяє адміністраторам переглядати результат ще до формування PDF. Крім того, реалізовано гнучке оновлення заявки через PATCH, можливість затвердити або відхилити запит, змінити статус, додати причину відмови, а також отримати список усіх заявок користувача або список всіх заявок у статусі очікування.

У межах реалізації модуля тестування для освітньої системи було створено повноцінний набір функцій на стороні бекенду, що забезпечує як створення тестів викладачами, так і їх проходження та оцінювання студентами. Цей функціонал був побудований відповідно до принципів чистої архітектури, з чітким розділенням відповідальностей між CRUD-логікою, схемами, перевітками прав доступу та контролерами маршрутизаторів FastAPI.

В основі лежить модель тесту, яка пов’язується з відповідним завданням (assignment). Вона підтримує структуру з питаннями та варіантами відповідей, де тип відповіді може бути як з однією правильною відповіддю, так і з кількома. Ми реалізували можливість створення тесту разом із питаннями за один запит – це значно полегшує процес для викладача. Дані типізуються через Pydantic-схеми, що забезпечує надійну перевірку вхідних запитів.

З боку студента ми реалізували процес початку тестування (start\_test\_attempt), при якому створюється окрема спроба проходження тесту. Ми ввели додаткову перевірку, яка не дозволяє студенту розпочати нову спробу, якщо вже існує активна (тобто не завершена). Крім цього, ми врахували обмеження

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 60   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

кількості дозволених спроб, тому якщо всі завершені, але ліміт вичерпано – нова спроба не створюється і повертається відповідне повідомлення.

Коли студент завершує проходження тесту, він надсилає свої відповіді через ендпоінт submit. У цей момент у бекенді виконується логіка перевірки відповідей: оцінюється правильність кожної відповіді, обраховується підсумковий бал, зберігається час виконання тесту, а також фіксується інформація про те, скільки часу користувач перебував поза вкладкою (як превентивний захід від шахрайства).

Коли студент починає тест, система сліdkує за тим, чи є поточна вкладка активною. Якщо користувач перемикається на іншу вкладку або згортає вікно браузера, в цей момент фіксується час, коли вкладка втратила фокус. Після повернення користувача у вкладку обчислюється тривалість, протягом якої він перебував поза активним вікном. Цей період додається до спеціального лічильника, що відображає загальний час неактивності.

Такий підхід дозволяє не лише підрахувати загальний час проходження тесту, а й окремо зберігати інформацію про неактивний час. Після завершення тесту ці дані передаються на сервер разом із відповідями студента. У результаті, система може більш об'єктивно оцінити процес проходження тесту, в тому числі враховуючи можливі спроби обійти правила.

Цей механізм також доповнюється збереженням даних у локальному сховищі браузера, що дозволяє відновити тест у разі випадкового закриття чи перезавантаження сторінки. Завдяки цьому студент може продовжити роботу без втрати прогресу, а система – зберегти цілісність усіх записів про хід тестування. Усі спроби тестування зберігаються у базі даних, і доступ до них обмежено роллю користувача. Студент бачить лише свої результати, а викладач або адміністратор – усі спроби для відповідного завдання. Для зручності ми реалізували окремі маршрути: один для отримання детального результату конкретної спроби, інший – для перегляду усіх спроб по завданню.

Також реалізовано можливість оновлення тесту з усіма його питаннями, що дозволяє викладачеві редагувати вже створений тест без необхідності вручну змінювати пов'язані записи.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 61   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

Загалом, модуль тестування забезпечує весь цикл роботи з тестами – від створення, проходження і до перевірки та збереження результатів, враховуючи доступи, ліміти спроб, гнучку структуру питань і зручну взаємодію з іншими модулями системи, зокрема з курсами та завданнями.

### **Висновки до другого розділу**

У результаті опрацювання всіх етапів другого розділу було повністю сформовано архітектурну та логічну основу інформаційної системи управління академічними ресурсами студентів. На етапі специфікації вимог проведено ґрунтовний аналіз функціональних і нефункціональних потреб майбутніх користувачів системи – студентів, викладачів і адміністраторів. Це дозволило чітко окреслити ключові задачі, які має вирішувати система, а також визначити критерії її ефективності, масштабованості та безпеки.

Було побудовано модель взаємодії користувачів із системою, описано ролі та відповідні сценарії роботи. Об'єктно-орієнтований підхід дав змогу структурувати систему за допомогою UML-діаграм і визначити логіку взаємозв'язків між основними компонентами.

Було реалізовано моделі з урахуванням всіх типів сутностей: користувачів, курсів, завдань, тестів, оцінок, документів та заявок. Встановлено зв'язки між таблицями з використанням механізмів зовнішніх ключів, cascade-видалення, обмежень цілісності й підтримки типів користувачів через поліморфізм.

Розроблено API-ендпоінти з підтримкою ролей, доступу до ресурсів, перевірки прав та валідації. Упроваджено сучасні практики розробки з розділенням відповідальностей, використанням FastAPI, Pydantic, асинхронної роботи з БД через SQLAlchemy та механізмами локального збереження даних на фронтенді. Окрему увагу приділено алгоритмам проходження тестів, обробці заявок на документи, управлінню курсами та автоматизованій генерації PDF-файлів.

У підсумку, вся архітектура системи побудована на сучасних підходах і забезпечує гнучкість, масштабованість та зручність як для розробників, так і для кінцевих користувачів. Результати другого розділу заклали надійну основу для реалізації повноцінного функціонального програмного продукту.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 62   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

## РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ УПРАВЛІННЯ АКАДЕМІЧНИМИ РЕСУРСАМИ СТУДЕНТІВ

### 3.1 Порядок встановлення та налаштування параметрів системи

Система управління академічними ресурсами студентів постачається як готовий інтегрований програмний продукт, який може бути розгорнутий у межах локальної інфраструктури навчального закладу або у хмарному середовищі. Для забезпечення швидкого та безпечного запуску використовується підхід контейнеризації з використанням Docker. Усі ключові компоненти системи – серверна логіка на FastAPI, клієнтська частина на React, база даних PostgreSQL, проксі-сервер Traefik та супутні сервіси – розгортаються як окремі контейнери, конфігурація яких описується в одному файлі за допомогою Docker Compose. Це дозволяє зменшити час інсталяції, знизити ризик помилок при налаштуванні, а також забезпечити однакове середовище для всіх користувачів.

Після успішного встановлення системи першим кроком є вхід адміністратора до панелі управління. Адміністратор має розширені повноваження, що дають змогу виконати первинне налаштування: створити облікові записи користувачів, призначити їм ролі (студент, викладач, адміністратор), а також прив'язати їх до відповідних факультетів, кафедр або навчальних програм. Усі користувачі отримують листи на електронну пошту з посиланням для активації облікового запису, після чого можуть самостійно встановити пароль і розпочати роботу з платформою. Завдяки цьому забезпечується безпечний доступ до системи та виключається несанкціоноване використання акаунтів.

Після активації облікових записів викладачі отримують доступ до функціоналу створення курсів. Кожен курс містить навчальні модулі, які включають лекції, мультимедійні матеріали, тести, завдання та інші ресурси. Викладач має можливість задавати структуру курсу, додавати теоретичні блоки, завантажувати презентації, відео, PDF-файли, створювати інтерактивні елементи тощо. Для оцінювання знань студентів реалізовано механізм завдань та тестів із підтримкою автоматичної або ручної перевірки. Передбачено також функціональність

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 63   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

встановлення дедлайнів, вагових коефіцієнтів для різних типів завдань та обмеження кількості спроб.

Студенти, своєю чергою, після додавання до курсів, отримують повний доступ до навчального контенту, тестування, а також можливість подавати відповіді на завдання. Інтерфейс особистого кабінету передбачає інтуїтивно зрозумілу навігацію, зручний доступ до розкладу, дедлайнів, повідомлень та результатів. Вся взаємодія з платформою ведеться через захищене REST API, що забезпечує стабільність передачі даних між клієнтом і сервером.

Отже, процес розгортання і налаштування системи охоплює не лише технічну інсталяцію, але й організаційні аспекти – керування ролями, активація користувачів, створення курсів, розподіл доступів та забезпечення контролю над навчальним процесом. Уся логіка побудована таким чином, щоб забезпечити гнучкість, безпеку та зручність у роботі для всіх учасників освітнього процесу.

### 3.2 Структура інтерфейсу системи управління академічними ресурсами

Система управління академічними ресурсами є закритою системою і тому доступ до неї мають тільки створені раніше адміністратором користувачі. У разі втрати пароля доступ до облікового запису може бути відновлено за допомогою функції скидання пароля.

Рис. 3.1 – Вхід в систему з можливістю відновлення паролю

Для цього на форму відновлення вводиться електронна адреса, вказана під час реєстрації (рис. 3.2).

Рис. 3.2 – Вхід в систему з можливістю відновлення паролю

На цю адресу надсилається лист із посиланням для встановлення нового пароля(рис. 3.4). Після переходу за посиланням та створення нового пароля доступ до системи відновлюється(рис. 3.5).

Рис. 3.3 – Повідомлення про успішний запит

Рис. 3.4 – Лист на пошті для скидання паролю

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 65   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

## Встановіть новий пароль

Пароль успішно змінено!

Новий пароль

Підтвердіть пароль

Змінити пароль

Рис. 3.5 – Заміна паролю

Особистий кабінет користувача містить усю ключову інформацію, яка сто- сується як персональних, так і навчальних даних (рис. 3.6 – 3.7). Всі дані структу- ровано за трьома основними напрямками: особиста інформація, навчальна інфор- мація та налаштування. У першому блоці відображаються основні персональні дані, такі як ПІБ, електронна пошта та інші контактні відомості. Навчальна інфор- мація охоплює факультет, групу, освітню програму й інші академічні параметри, що стосуються навчального процесу. Окремий розділ передбачений для виведення інформації про профіль.

- Головна сторінка
- Бібліотека курсів
- Мои курси
- Профіль**
- Подати заяву
- Оцінки
- Календар
- Чат
- Активні сесії

Е - університет
ipz214\_gsv@student.ztu.edu.ua Вийти

### Горай Софія Віталіївна

EMAIL

ipz214\_gsv@student.ztu.edu.ua

PHONE

+380981069673

BIRTH DATE

03.06.2025

ADDRESS

Чуднівська 103

Редагувати

#### Освітня інформація

|                                                                                                                   |                                                                                                                         |                                                                                                |                                                                          |
|-------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------|
| <p>ФАКУЛЬТЕТ</p> <p>Факультет інформаційних технологій</p> <p>КАФЕДРА</p> <p>Кафедра програмного забезпечення</p> | <p>СТУДЕНТСЬКИЙ КВИТОК</p> <p>8762996826044392</p> <p>СПЕЦІАЛЬНІСТЬ</p> <p>Інженерія програмного забезпечення (ІПЗ)</p> | <p>ОСВІТНЯ ПРОГРАМА</p> <p>Інженерія програмного забезпечення</p> <p>ГРУПА</p> <p>ІПЗ-21-5</p> | <p>СТУПІНЬ ОСВІТИ</p> <p>Бакалавр</p> <p>ФОРМА НАВЧАННЯ</p> <p>Денна</p> |
|-------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------|

#### Інформація про обліковий запис

|                                                                                                              |                                                                                                                                                                                                                            |
|--------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>ДАТА СТВОРЕННЯ</p> <p>5 червня 2025 р. о 15:59</p> <p>ДАТА ОНОВЛЕННЯ</p> <p>18 червня 2025 р. о 21:24</p> | <p>СТАТУС АКАУНТА</p> <p><span style="color: green;">Активний</span></p> <p>ДВОФАКТОРНА АУТЕНТИФІКАЦІЯ</p> <p><input type="checkbox"/> Вимкнено</p> <p style="font-size: 0.7em;">Рекомендується для підвищення безпеки</p> |
|--------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Рис. 3.6 – Особистий кабінет студента

**Е - університет**

teacher4@gmail.com Вийти

**Профіль**

**Гуцуляк Світлана Василівна** Редагувати

EMAIL: teacher4@gmail.com BIRTH DATE: 25.11.1988  
PHONE: +380674444444 ADDRESS: Одеса

**Освітня інформація**

НАВЧАЛЬНИЙ: ФІЗИЧ. ІД ВИКЛАДАЧА: 131371330 ПОСАДА: Дікан  
КАТЕДРА: РІЧНЕ ЗВАННЯ: Асистент НАУКОВИЙ СТУПІНЬ: магістр  
ІН. ПУО

**Інформація про обліковий запис**

ДАТА СТВОРЕННЯ: 10 червня 2025 р. о 15:33 СТАТУС АКАДИТА: Активний  
ДАТА ОНОВЛЕННЯ: 20 червня 2025 р. о 07:00 ДИНАМІЧНА АУТЕНТИКАЦІЯ: Вимкнено  
Рекомендується для підключення безпеки

Рис. 3.7 – Особистий кабінет викладача

Зміна особистих даних відбувається у відповідному розділі особистого кабінету (рис. 3.8). Користувач може редагувати лише персональну інформацію, таку як електронна пошта або контактні дані. Відомості про навчання (наприклад, факультет, група, освітня програма) є незмінними для звичайних користувачів і можуть бути оновлені лише адміністраторами системи.

**Горай** Зберегти Скасувати

**Софія**

**Віталіївна**

EMAIL: ipz214\_gsv@student.ztu.edu.ua BIRTH DATE: 03.06.2025  
PHONE: +380981069673 ADDRESS: Чуднівська 103

Рис. 3.8 – Редагування особистих даних

Розділ “Запит на документ” призначений для подачі електронних заяв або отримання шаблонів офіційних документів. На сторінці доступні різні типи заяв та довідок (рис. 3.9).

**Е - університет**

ipz214\_gsv@student.ztu.edu.ua Вийти

**Запит на документ**

**Заява на перескладання іспиту**  
Шаблон заяви студента на повторне складання іспиту у зв'язку з пропуском або незадовільною оцінкою.  
заява Потрібне затвердження

**Сертифікат учасника конференції**  
Сертифікат, що підтверджує участь у студентській науковій конференції 2025 року.  
сертифікат Потрібне затвердження

**Довідка про навчання**  
Шаблон для довідки, що підтверджує факт навчання студента.  
сертифікат

**Направлення на практику**  
Шаблон документа, який підтверджує направлення студента на проходження виробничої або навчальної практики.  
заява Потрібне затвердження

**Заява на академічну відпустку**  
Шаблон заяви студента про надання академічної відпустки на підставі поважних причин (за станом здоров'я, сімейними обставинами тощо).  
заява Потрібне затвердження

Рис. 3.9 – Доступні варіанти заяв

Кожен документ супроводжується коротким описом його призначення та точним статусом - наприклад, чи потребує запит погодження. Оформлення заявки відбувається через вибір потрібного документа й заповненням відповідної форми (рис 3.10).

Якщо необхідні для формування довідки дані вже зберігаються у профілі користувача (наприклад, ПІБ, дата народження, навчальна група, факультет або освітня програма), ці відомості автоматично підтягуються у відповідні поля заяви або довідки під час створення запиту. Користувач бачить ці дані у формі, але не має можливості їх змінювати чи редагувати – це гарантує коректність та актуальність інформації, яка використовується для офіційних документів. Якщо документ не потребує погодження, він генерується автоматично та доступний для завантаження.

**Запит на документ**

**Довідка про навчання**

Назад до шаблонів

Шаблон для довідки, що підтверджує факт навчання студента.

Номер студентського (з профілю)

8762996826044392

Це поле заповнюється автоматично з профілю

Прізвище (з профілю)

Горай

Це поле заповнюється автоматично з профілю

Ім'я (з профілю)

Софія

Це поле заповнюється автоматично з профілю

По батькові (з профілю)

Віталіївна

Це поле заповнюється автоматично з профілю

Дата народження (з профілю)

2025-06-03

Рис. 3.10 – Заповнення форми даними

У випадку, якщо для отримання документа передбачено попереднє погодження, після подачі заяви вона автоматично надсилається на розгляд адміністрації (рис. 3.12). Користувач отримує сповіщення про статус розгляду, а можливість отримати готовий документ з'являється лише після його затвердження адміністратором (рис. 3.15).

Якщо студент вже подавав запит на отримання певної заяви, ще раз він цього зробити не може, поки не буде відповіді від адміністратора (рис. 3.11).

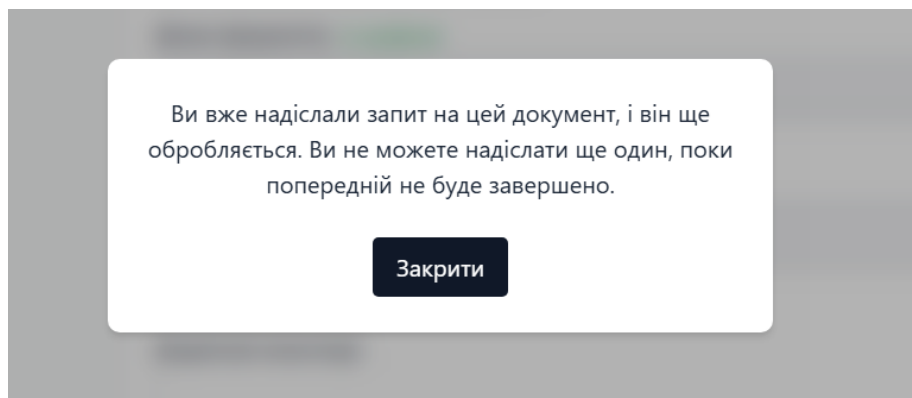


Рис. 3.11 – Обробка помилок

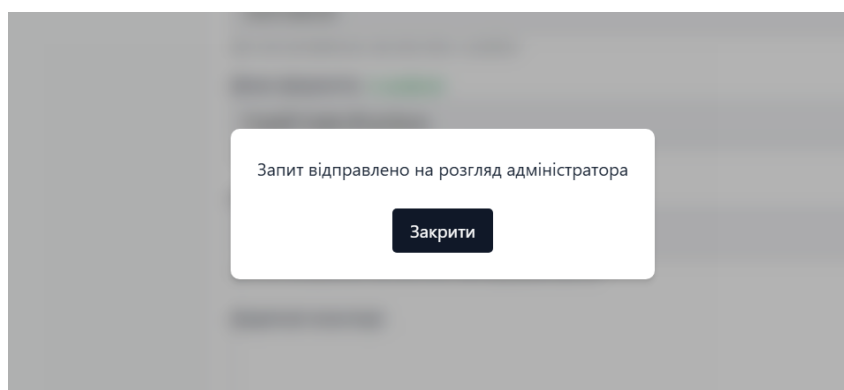


Рис. 3.12 – Сповіщення про успішне надсилання форми

У розділі для адміністратора відображається перелік усіх запитів на отримання документів, поданих користувачами. Для кожної заявки вказується унікальний ідентифікатор документа, поточний статус (наприклад, “Очікує”), а також дата створення запиту (рис. 3.13).

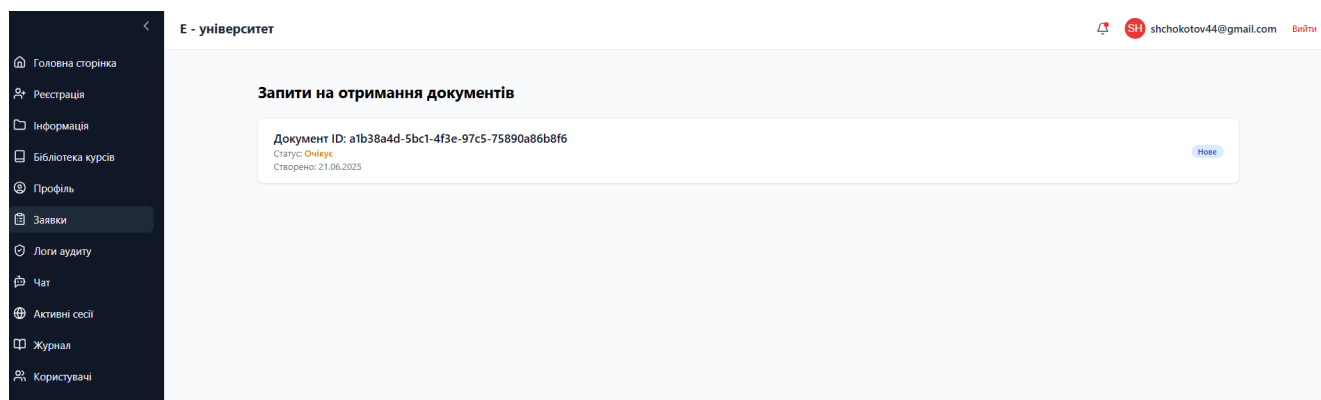


Рис. 3.13 – Запит на підтвердження у адміністратора

|      |      |          |        |      |                          |            |
|------|------|----------|--------|------|--------------------------|------------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк.<br>69 |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |            |

Коли адмін переглядає запит то йому надається відповідний статус (рис. 3.14)

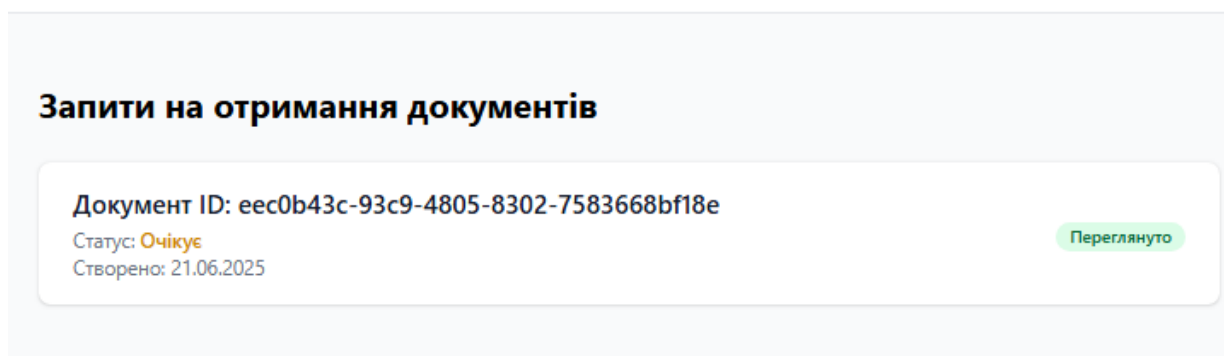


Рис. 3.14 – Переглянутий запит

Адміністратор має можливість переглядати подані заявки, за необхідності редагувати їх зміст, а також приймати рішення про затвердження або відхилення заяви (рис. 3.15).

Назад до списку

Документ: a1b38a4d-5bc1-4f3e-97c5-75890a86b8f6 Затвердити Відхилити

Метадані

| Статус          | Дата створення       | Коментар         |
|-----------------|----------------------|------------------|
| Очікує розгляду | 21.06.2025, 12:02:45 | Немає коментарів |

**Е-УНІВЕРСИТЕТ**  
**ДОВІДКА**

№ 2511730608456841

На підставі наказу про зарахування від 2025-06-17

ПІБ: Красновбоский2 Михайло2 Олександрович2  
Дата народження: 2003-05-13  
Номер студентського квитка: 2511730608456841  
Адреса проживання: Харків  
Контактний телефон: +380671234354  
Електронна пошта: student2-2@gmail.com  
Факультет: ФІКТ  
Кафедра: test\_department  
Спеціальність: Право  
Освітня програма: ІПЗ  
Рівень вищої освіти: test  
Форма навчання: test  
Група: ІПЗ-21-4

Довідка видана для підтвердження факту навчання у вищому навчальному закладі та може бути використана для пред'явлення в офіційних установах, органах державної влади, для отримання пільг, що передбачені чинним законодавством України для студентів вищих навчальних закладів.

Довідка дійсна протягом 30 (тридцяти) календарних днів з дня її видачі.

Довідка видана без права пред'явлення вищою до університету.

Гуцулак І. В.  
(Декан факультету)

Дата видачі: 2025-06-21

ДЕРЖАВНИЙ  
УНІВЕРСИТЕТ

№ 1234-0012

Рис. 3.15 – Форма для перегляду з можливістю редагування

У особистому кабінеті студент може завантажити свою готову довідку (рис. 3.16)

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 70   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

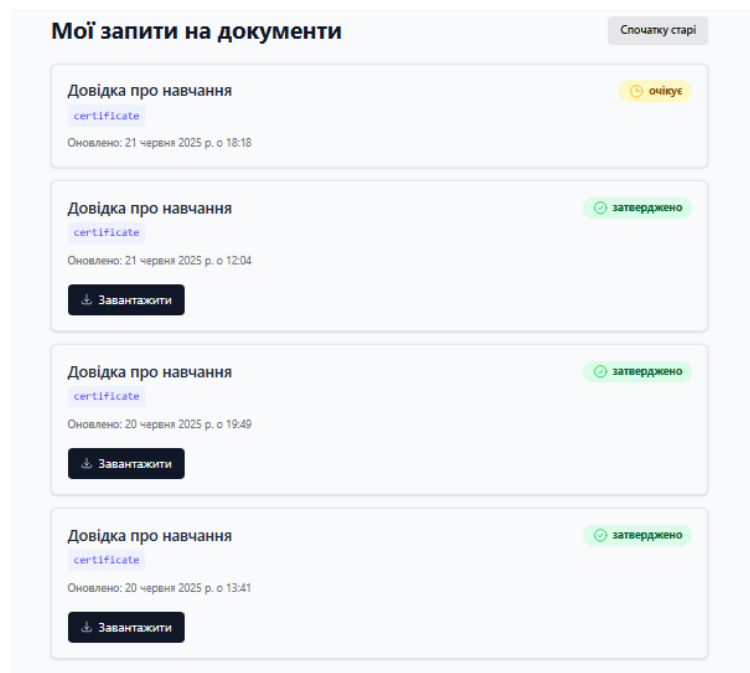


Рис. 3.16 – Сформовані заяви у кабінеті студента

На сторінці курсу відображається структура навчальних модулів із докладним описом змісту. Для кожного модуля подано короткий опис тематики, що вивчається, а також перелік уроків, які входять до складу цього модуля. Усередині кожного уроку можуть міститися різні типи завдань. Для кожного завдання зазначаються назва, тип, терміни виконання та деталі (рис. 3.17).

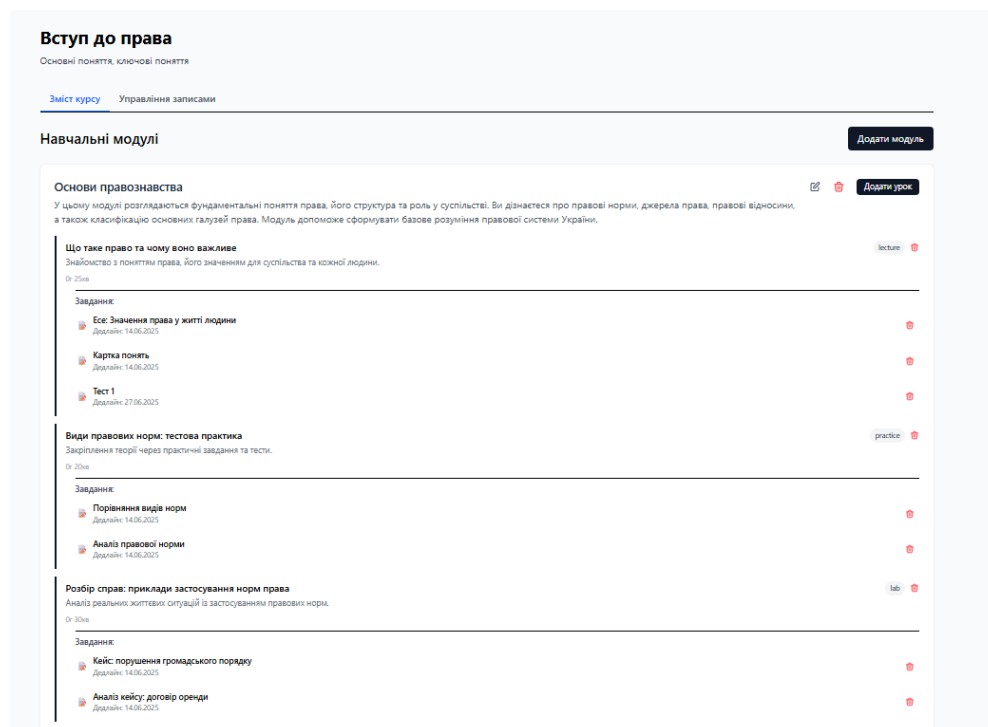


Рис. 3.17 – Навчальна сторінка курсу для викладача

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 71   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

Інтерфейс для адміністратора та викладача дозволяє додавати нові модулі й уроки за допомогою відповідних кнопок у правій частині екрана. Також передбачена можливість редагування опису модуля, видалення уроків чи окремих завдань.

**Основи правознавства**

У цьому модулі розглядаються фундаментальні поняття права, його структура та роль у суспільстві. Ви дізнаєтеся про правові норми, джерела права, правові відносини, а також класифікацію основних галузей права. Модуль допоможе сформувати базове розуміння правової системи України.

**Додати урок**

Назва\*  Тривалість (хв)

Опис

Контент

Відео

Презентація

Рис. 3.18 – Додавання нового уроку

Кожен урок у складі навчального модуля може містити різні типи навчальних матеріалів (рис. 3.19).

**Е - університет**

← Назад до курсу

**Урок1**

Ми ознайомимося з основами мови Python: синтаксисом, типами даних

**Відео контент**

**Навчальні матеріали**

Змінна в Python — це іменована «контейнер» або «ящик», у якому зберігається певне значення або дані. За допомогою змінної можна зберігати інформацію, щоб потім використовувати її у програмі. У Python змінні не потрібно заздалегідь оголошувати з типом — мова сама визначає тип даних автоматично, залежно від того, що ви їй присвоїли. Це називається динамічною типізацією. Змінні використовують для збереження чисел, тексту, логічних значень та інших типів даних, щоб оперувати ними в програмах.

Рис. 3.19 – Інтерфейс уроку

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 72   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

Зокрема, до уроку можуть бути додані відеоматеріал, основний текстовий навчальний матеріал та презентація. Викладач та адміністратор має можливість редагувати зміст уроку: додавати або оновлювати відео, змінювати текстовий матеріал, завантажувати нові презентації, а також коригувати структуру й назву уроку відповідно до навчального плану. До уроку можна створити різні типи завдань: практична робота, лабораторна робота або тест. Важливо зазначити, що доступ на редагування та видалення є лише у адміністратора, вчитель може редагувати тільки свої модулі.

На сторінці завдання відображається повна інформація, необхідна для виконання студентом лабораторної або практичної роботи (рис. 3.20). Опис завдання містить умови, приклади вхідних та вихідних даних, перелік вимог до програми або файлу-відповіді, а також критерії оцінювання. Для зручності подано деталізовані параметри: тип подачі (наприклад, файл), кінцевий термін здачі, максимальний бал, коефіцієнт, ліміт повторних спроб і поточний статус завдання.

Назад

**Лабораторна робота 2**

**Опис**

**Завдання:** Магазин чарівних зілля  
 Ти працюєш у крамниці магичних зілля. Твоя задача — написати програму, яка рахує вартість покупки залежно від обраного зілля та кількості. І робить знижку, якщо клієнт купує багато.

**Умови:**  
 У магазині є 3 види зілля:

Зілля здоров'я — 50 золотих за одиницю

Зілля мани — 30 золотих

Зілля невидимості — 120 золотих

**Твоя програма має:**  
 Запитати у користувача назву зілля (одне з трьох).

Запитати кількість пляшечок.

Обчислити загальну суму.

Якщо кількість більша або дорівнює 5 — дати знижку 10%.

Вивести фінальну суму до оплати.

**Деталі завдання**

Тип: Лабораторна робота

Тип подання: file

Кінцевий термін: 21.06.2025, 00:00:00

Максимальний бал: 100

Коефіцієнт: 1

Ліміт повторних подань: 1

Статус: Активний

**Ваші відповіді**

Статус: graded

Відправлено: 14.06.2025

Роботу перевірено. Надіслати нову відповідь більше не можна.

Оцінка: 60 / 100

Детальніше

Рис. 3.20 – Завдання для уроку

Вчитель та адміністратор маю право на створення нового завдання (рис 3.21)

The form is titled "Створити нове завдання" (Create new task). It contains several input fields and dropdown menus:

- Назва\*** (Name): A text input field.
- Опис\*** (Description): A larger text input field.
- Тип завдання\*** (Task type): A dropdown menu with "Лабораторна робота" (Laboratory work) selected.
- Максимальний бал\*** (Maximum score): A text input field with "100" entered.
- Коефіцієнт\*** (Coefficient): A text input field with "1" entered.
- Дедлайн\*** (Deadline): A date picker showing "28.06.2025".
- Коефіцієнт протермінування** (Coefficient of preterm): A text input field with "0,5" entered.
- Обов'язкове** (Mandatory): A checked checkbox.
- Тип подання\*** (Submission type): A dropdown menu with "Файл" (File) selected.
- Ліміт повторних подань** (Limit of re-submissions): A text input field with "1" entered.
- PDF файл (завдання)** (PDF file (task)): A button labeled "Вибрати файл" (Select file).
- Критерії оцінювання** (Grading criteria): A large text input field.
- Створити завдання** (Create task): A blue button at the bottom right.

Рис. 3.21 – Створення завдання

Студенту доступний інтерфейс для здачі завдання (рис 3.22)

The interface shows a task submission page with the following sections:

- Назад** (Back): A link at the top left.
- Привітання** (Greeting): A section with a welcome message and task details.
- Опис** (Description): A text area containing the task description and instructions.
- Деталі завдання** (Task details): A table showing task parameters.
- Ваші відповіді** (Your answers): A section for submitting answers.
- Надіслати відповідь** (Submit answer): A button to submit the answer.
- Завантажити файл** (Upload file): A button to upload a file.
- Коментар (необов'язково)** (Comment (optional)): A text input field for a comment.
- Надіслати роботу** (Submit work): A blue button at the bottom.

| Деталі завдання             |                      |
|-----------------------------|----------------------|
| Тип:                        | lab                  |
| Коефіцієнт протермінування: | 0.5                  |
| Кінцевий термін:            | 27.06.2025, 00:00:00 |
| Максимальний бал:           | 100                  |
| Коефіцієнт:                 | 1                    |
| Ліміт повторних подань:     | 3                    |
| Статус:                     | Активний             |

Рис. 3.22 – Інтерфейс для надання відповіді

Після надання відповіді студентом, вчитель отримує їх у себе в завданнях, він може виставити оцінку, повернути роботу на доопрацювання .

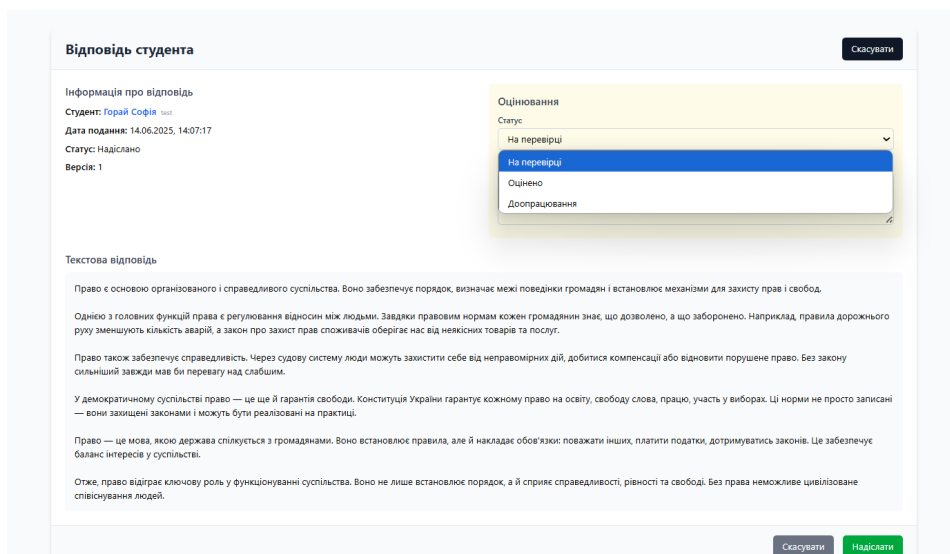


Рис. 3.23 – Статуси перевірки завдання

У нижній частині сторінки студент може переглянути статус своєї відповіді: дату подання, отриману оцінку, кількість спроб і доступність повторної подачі. Додатково може відображатися коментар викладача з рекомендаціями або зауваженнями до виконаної роботи, що сприяє покращенню якості наступних спроб. Якщо завдання перевірено, вказується оцінка та інформація про неможливість подати нову відповідь. У випадку, коли робота студента була повернута на доопрацювання (рис. 3.24).



Рис. 3.24 – Статуси робіт

Вчитель має доступ до створення тесту для завдання. Коли вчитель створює завдання для тесту він має підв'язати до нього тест. Завдання відображається як приховане від студентів (рис. 3.25).

← Назад

**ТЕМТ** Редагувати завдання

Опис

Деталі завдання

Тип: test

Коефіцієнт протермінування: 0.5

Кінцевий термін: 27.06.2025, 00:00:00

Максимальний бал: 100

Коефіцієнт: 1

Ліміт повторних подань: 22

Статус: Приховано від студентів

До завдання не прив'язано жодного тесту Створити тест

Результати тесту

| СТУДЕНТ | СТАТУС | БАЛИ | ЧАС | ЧАС ПОЗА ВКЛАДКОЮ |
|---------|--------|------|-----|-------------------|
|---------|--------|------|-----|-------------------|

Рис. 3.25 – Створення завдання для тесту

Створення тесту здійснюється через спеціальний інтерфейс, у якому викладач визначає основні параметри: назву, опис, час на виконання, спосіб відображення результатів, а також додаткові опції, такі як перемішування відповідей і показ правильних відповідей після завершення тестування (рис. 3.26). Далі формується перелік питань, які додаються у відповідному розділі. Для кожного питання встановлюється тип (з однією або кількома правильними відповідями), кількість балів та порядок. Після завершення налаштувань тест зберігається та стає доступним для використання у курсі.

Створення нового тесту Закрити

Основні налаштування тесту

Назва тесту

Опис тесту

Показувати результати: Після спроби

Час на тест (хвилин)

☐ Перемішувати відповіді

☐ Показувати правильні відповіді

Питання тесту Додати питання

Нове питання

Тип питання

Бали

Порядок

Активне питання

Варіанти відповідей

Розподілення варіантів відповідей

Правильні відповіді

Зберегти

Зберегти варіант

Зберегти питання

Зберегти тест

Рис. 3.26 – Конструктор тесту

Для кожного питання доступні функції редагування або видалення (рис. 3.27). Після внесення змін викладач зберігає тест, і він одразу стає доступним для проходження студентами у межах відповідного курсу.

Рис. 3.27 – Редагування тесту

Після відкриття тесту студент отримує інформацію про параметри завдання: тип, коефіцієнт, термін виконання, максимальний бал, кількість дозволених спроб і статус. Відображається назва тесту та короткий опис, а також інформація про вже використані спроби (рис. 3.28).

Рис. 3.28 – Можливість продовжити тест

Під час виконання тесту студента супроводжує таймер, що відраховує час до завершення (рис 3.29).

Вступний тест: Основи PHP Час: 49:03

Питання 1 з 3

Що таке PHP?

- ☐ Текстовий редактор
- ☐ Мова розмітки для вебсторінок
- ☐ Програма для редагування зображень
- ☐ Мова програмування на стороні сервера

Наступне питання

Рис. 3.29 – Проходження тесту

Кожне питання відображається окремо, причому є можливість повернутися до попереднього питання для перегляду або зміни відповіді (рис. 3.30). Після завершення роботи над питанням студент може надіслати свої відповіді на перевірку, скориставшись відповідною кнопкою.

Вступний тест: Основи PHP Час: 48:38

Питання 3 з 3

Як вивести текст "Hello, world!" у PHP?

- ☐ console.log("Hello, world!");
- ☐ echo "Hello, world!";
- ☐ print("Hello, world!");

Попереднє питання Відправити тест

Рис. 3.30 – Можливість повернутись до попереднього питання

Після здачі тесту студенту відображаються результати проходження: кількість набраних балів, статус виконання, а також, залежно від налаштувань тесту, перелік правильних відповідей для кожного питання (рис. 3.31). Якщо в параметрах

тесту дозволено показ правильних відповідей, студент може одразу порівняти свої відповіді з ними, проаналізувати допущені помилки та зрозуміти причини нарахування чи втрати балів. Для завершених спроб доступний детальний перегляд.

**Огляд тесту: Вступний тест: Основи PHP**

1. Що таке PHP?

Програма для редагування зображень

Текстовий редактор

Мова розмітки для вебсторінок (Ваша відповідь)

Мова програмування на стороні сервера (Правильна)

2. ким тегом відкривається PHP-код у HTML-документі?

<?php ... ?> (Правильна) (Ваша відповідь)

<php>

<!--php-->

<script type="php">

3. Як вивести текст "Hello, world!" у PHP?

print("Hello, world!"); (Правильна) (Ваша відповідь)

echo "Hello, world!"; (Правильна) (Ваша відповідь)

console.log("Hello, world!");

Сума балів: 2

Рис. 3.31 – Перегляд результатів

Після завершення тестування викладач має доступ до підсумкової таблиці результатів, у якій для кожного студента відображається статус виконання, кількість набраних балів, загальний час, витрачений на проходження тесту, а також додатковий показник - “час поза вкладкою” (рис. 3.32). Ця функція дозволяє визначити, скільки часу студент перебував поза тестом (наприклад, перемикався на інші вкладки браузера або програми) під час виконання завдання. Завдяки такому контролю можна об’єктивніше аналізувати самостійність виконання роботи, виявляти підозрілі випадки порушення академічної доброчесності або отримання підказок

сторонніми способами. За необхідності викладач має змогу переглянути детальну історію спроб студента, а також оперативно реагувати на нетипову поведінку під час тестування. Це підвищує рівень довіри до результатів оцінювання та сприяє забезпеченню чесних умов проходження тесту.

— Назад

**Тест** Редагувати завдання

Опис  
Тест

Деталі завдання

Тип: test

Коефіцієнт: 0.5

Протермінування:

Кінцевий термін: 27.06.2025, 00:00:00

Максимальний бал: 100

Коефіцієнт: 1

Ліміт повторних подань: 2

Статус: Активний

Тест: Вступний тест: Основи РНР  
Без опису Редагувати тест

**Результати тесту**

| студент                               | СТАТУС  | БАЛИ | ЧАС    | ЧАС ПОЗА ВКЛАДКОЮ |
|---------------------------------------|---------|------|--------|-------------------|
| Горай Софія Віталіївна                | Закрито | 0    | 0 сек  | 0 сек             |
| Красовіловий2 Михайло2 Олександрович2 | Закрито | 2    | 37 сек | 0 сек             |

Рис. 3.32 – Перегляд результатів для вчителя

Після отримання студентської відповіді викладач має можливість переглянути текст роботи, залишити власний коментар для студента, а також оцінити виконане завдання, вказавши відповідну кількість балів (рис 3.33).

**Відповідь студента** Оцінка: 8

Інформація про відповідь

Студент: Горай Софія test

Дата подання: 14.06.2025, 14:00:23

Статус: graded

Версія: 1

Коментар студента

Було цікаво замислитися про це у повсякденному контексті

Ваш коментар

Молодець

Текстова відповідь

Право відіграє ключову роль у моєму повсякденному житті, оскільки воно регулює всі аспекти взаємодії з іншими людьми, державними органами та навіть у межах родини. Завдяки правовим нормам я знаю, що мої права захищені — наприклад, у школі чи лікарні. Я можу звернутися до поліції або суду, якщо хтось порушить мої права. Крім того, знання права допомагає уникати конфліктів та вирішувати суперечки цивілізовано. Право формує почуття справедливості, відповідальності та поваги до інших. У моєму щоденному житті воно проявляється навіть у дрібницях — коли я переходжу дорогу на зелене світло або купую товар з гарантією.

— Назад

Рис. 3.33 – Оцінка роботи студента

Для студента є календар завдань, який відображає всі заплановані тести, лабораторні та практичні роботи у вигляді інтерактивної сітки за місяць. Для кожного дня відзначаються всі завдання, що мають бути виконані до певної дати, із зазначенням типу роботи та короткими деталями. При виборі конкретного дня у нижній частині сторінки відображається перелік завдань на цю дату з додатковою інформацією щодо кожного з них (рис. 3.34).

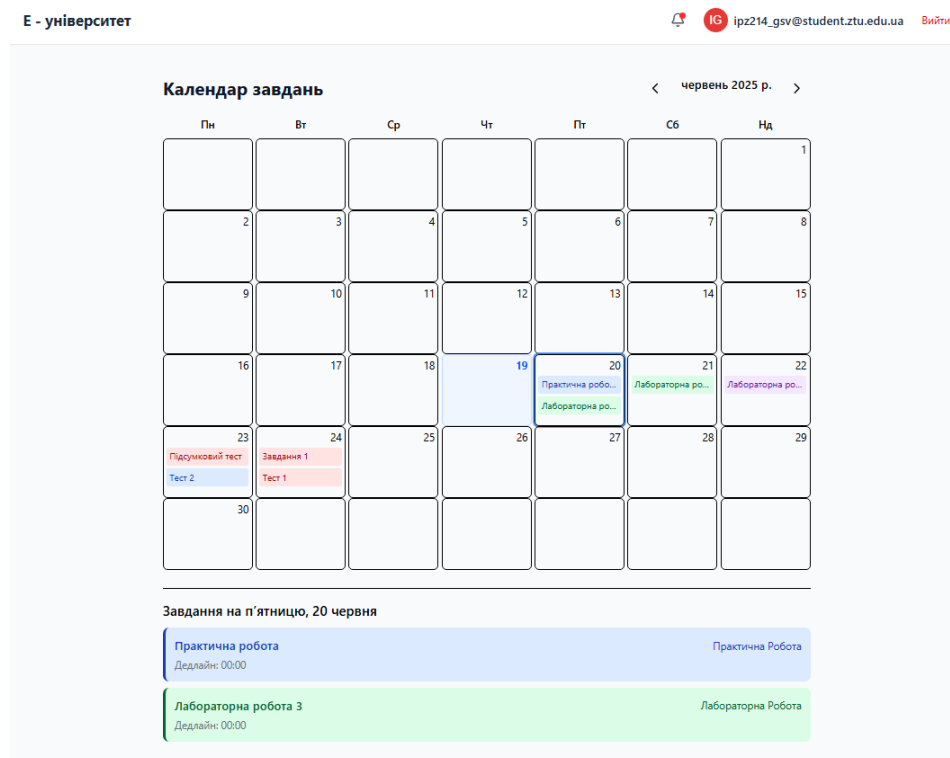


Рис. 3.34 – Календар запланованих робіт

У студента є можливість переглядати свої оцінки за курс (рис. 3.35). Журнал оцінок містить зведену інформацію про всі отримані результати за різними видами завдань і курсів. Для кожної лабораторної, тесту чи іншої роботи відображається оцінка, максимальний бал, дата оцінювання, а також ім'я викладача, який перевіряв завдання. Якщо оцінка була знижена через запізнення, це також зазначається поряд із відповідним коментарем. Додатково надається можливість переглянути деталі кожної роботи та ознайомитися з коментарем викладача щодо виконання. Даний формат дозволяє студенту контролювати власний прогрес і своєчасно реагувати на зауваження або результати перевірки.

**Журнал оцінок** За курсами За типами завдань

Програмування мовою Python ▲

Лабораторна робота 2 Оцінено

**Оцінка: 60 / 100**

Викладач: Горай Софія Віталіївна Оцінено: 15.06.2025

[Переглянути роботу](#)

Тест 1

**Оцінка: 100 / 100**

Коментар викладача:  
Оцінка за тестову спробу

Оцінено: 17.06.2025

[Переглянути роботу](#)

Лабораторна робота 3 Оцінено

**Оцінка: 98 / 100** (3 запізненням: -2 б)

Викладач: Горай Софія Віталіївна Оцінено: 15.06.2025

[Переглянути роботу](#)

Рис. 3.35 – Особисті оцінки студента

Розділ “Інформаційна система навчального закладу” містить основні довідники освітнього процесу, включаючи факультети, кафедри, спеціальності, освітні програми та академічні групи (рис. 3.36).

**Інформаційна система навчального закладу**

[Факультети](#)
[Кафедри](#)
[Спеціальності](#)
[Освітні програми](#)
[Групи](#)
Додати спеціальність

**Розробка комп'ютерних ігор (122)**

Факультет: ФІКТ  
Кафедра: ІПЗ

**Інженерія програмного забезпечення (121)**

Факультет: ФІКТ  
Кафедра: ІПЗ

**Правоохоронна діяльність (093)**

Факультет: ФПУП  
Кафедра: ПР

**Право (081)**

Факультет: ФПУП  
Кафедра: ПУУ

Рис. 3.36 – Сторінка з інформаційною системою навчального закладу

Доступ до редагування цього розділу мають лише адміністратори системи. Для кожної вкладки передбачена можливість додавання нових записів, редагування або видалення існуючих (рис. 3.37). Такий інтерфейс дозволяє адміністраторам

своєчасно підтримувати актуальність та повноту інформації про організаційну структуру закладу, забезпечуючи коректну роботу навчальної системи та автоматизацію академічних процесів.

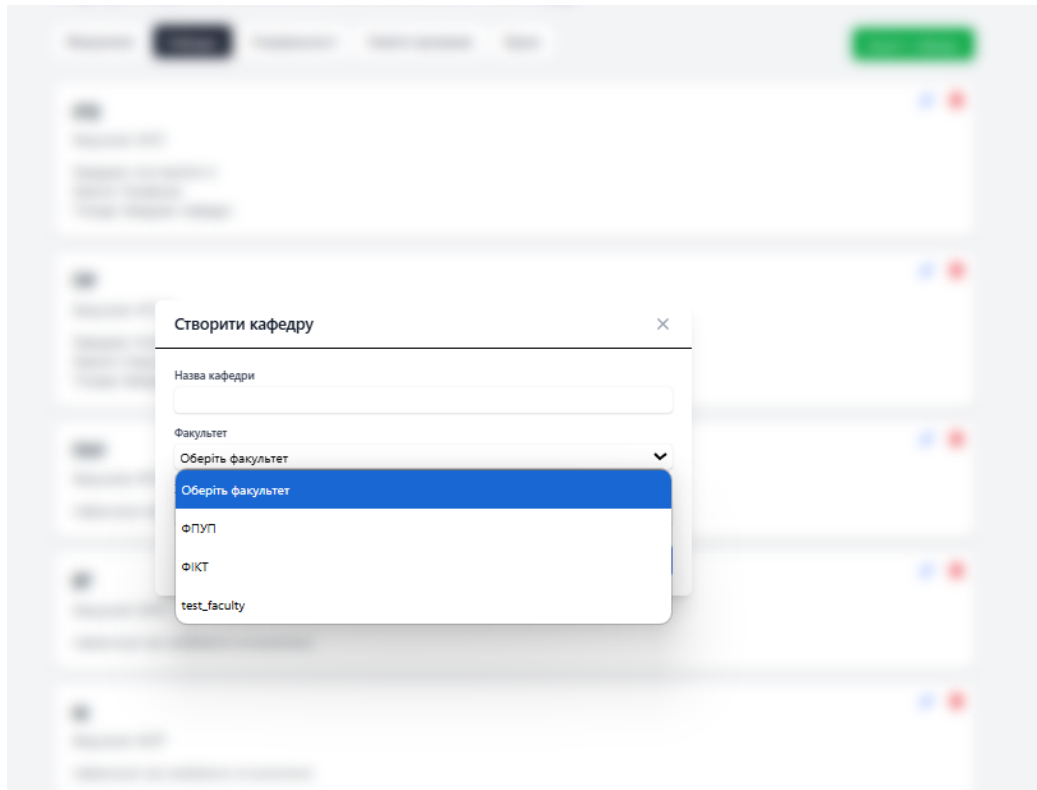


Рис. 3.37 – Створення нової кафедри

### 3.3 Тестування роботи програмного комплексу

Тестування програмного комплексу є невід’ємною частиною процесу розробки та впровадження системи управління академічними ресурсами студентів. Його основна мета – впевнитися, що система функціонує коректно, відповідає технічним і бізнес-вимогам, а також забезпечує надійну й безпечну взаємодію з користувачами. Важливим є також забезпечення стабільної роботи в умовах реального навантаження та багатокористувацького середовища.

Тестування відбувалося у кілька етапів. На початковому рівні проводилося модульне тестування [30], яке передбачає перевірку окремих функцій, методів та компонентів програми. Це дозволило локалізувати помилки на ранньому етапі й переконатися у правильності логіки роботи окремих елементів – зокрема, створення курсів, перевірки прав доступу, збереження оцінок, обробки запитів REST API.

Далі здійснювалося інтеграційне тестування, під час якого перевірялася взаємодія між окремими модулями – наприклад, між модулем подання завдань, їх оцінюванням та журналом оцінок. Особливу увагу приділяли коректному обробленню викликів API, цілісності даних у базі, а також відповідності результатів очікуваній поведінці системи. Також перевірялися сценарії з різними ролями користувачів – студента, викладача, адміністратора – що дозволило виявити та усунути недоліки в авторизації, фільтрації даних та логіці доступу.

На наступному етапі проводилося системне тестування на локальному сервері та в середовищі Docker із симуляцією реальної експлуатації. Під час тестування перевірялась робота всього комплексу в цілому: від входу користувача до виконання ним повного циклу дій (вивчення матеріалу, проходження тесту, отримання оцінки тощо). Також перевірялась поведінка системи за умов пікового навантаження: запускалося одночасно кілька сесій з різними обліковими записами, що взаємодіяли з платформою паралельно.

Окремо було виконано тестування інтерфейсу користувача. Перевірялась коректність відображення елементів, зручність навігації, адаптивність інтерфейсу до різних типів пристроїв. Здійснювались перевірки відповідності UX-рішень очікуванням студентів та викладачів: доступність кнопок, логічність переходів, зрозумілі повідомлення про помилки тощо.

Завершальний етап включав регресійне тестування, яке проводилося після внесення змін або додавання нових функцій. Його завданням було переконатися, що нововведення не спричинили порушення вже існуючої функціональності[31].

Результати тестування підтвердили працездатність системи в основних сценаріях використання, а також виявили низку незначних технічних і логічних помилок, які були оперативно усунені. Система продемонструвала стабільність, високу швидкість відгуку при базовому та середньому навантаженні, а також коректне функціонування у середовищі з реалістичними умовами доступу.

Завдяки комплексному підходу до тестування забезпечено відповідність розробленого програмного забезпечення вимогам користувачів, стандартам якості та критеріям продуктивності.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 84   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

## Висновки до третього розділу

У результаті аналізу інтерфейсу та порядку роботи з системою управління академічними ресурсами студентів можна стверджувати, що застосований підхід контейнеризації на базі Docker Compose значно спрощує розгортання середовища, об'єднуючи серверну частину на FastAPI, клієнтський фронтенд на React, базу даних PostgreSQL і проксі-сервер – Traefik в єдине інтегроване рішення. Така архітектура дозволяє мінімізувати помилки при налаштуванні, забезпечити ідентичність середовища розробки та експлуатації в різних інфраструктурних умовах і гарантувати безпечний доступ за ролями: адмін, викладач, студент.

Інтуїтивно зрозумілий персональний кабінет, що включає блоки персональної інформації, навчальних даних та налаштувань, дає користувачам можливість швидко орієнтуватися в платформі, відстежувати дедлайни, переглядати результати й обмінюватися повідомленнями. Викладачі отримують гнучкі інструменти для побудови курсів, модулів, завдань і тестів із налаштуванням дедлайнів, вагових коефіцієнтів і лімітів спроб, тоді як студенти – чітке відображення завдань і результатів із можливістю повторного проходження та аналізу помилок.

Багаторівневе тестування програмного комплексу – від модульного й інтеграційного до системного, інтерфейсного та регресійного – підтвердило стабільність роботи всіх компонентів при реальному навантаженні, коректність обробки даних і відповідність очікуванням різних ролей користувачів. Це дозволило не лише переконатися в надійності та продуктивності платформи, але й своєчасно усунути виявлені недоліки, що суттєво підвищило якість кінцевого рішення.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 85   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

## ВИСНОВКИ

У ході виконання комплексної кваліфікаційної роботи було спроектовано та розроблено інформаційну систему управління академічними ресурсами студентів, яка відповідає сучасним вимогам до веб-застосунків у сфері освіти. Основною метою проєкту стало створення інтегрованої платформи, що забезпечує ефективну взаємодію між усіма учасниками освітнього процесу: студентами, викладачами та адміністрацією навчального закладу.

На етапі аналітики було сформульовано бізнес-вимоги та функціональні потреби системи, що дало змогу чітко визначити основні модулі: управління курсами, завданнями, тестами, навчальними матеріалами, поданням документів, оцінюванням і моніторингом прогресу студентів. Було застосовано сучасні підходи до архітектури програмного забезпечення – клієнт-серверна модель, яка дозволила чітко розмежувати обов’язки між клієнтською частиною на базі React і серверною логікою на базі FastAPI. Для зберігання даних було обрано СУБД PostgreSQL, яка забезпечила високу надійність та ефективну обробку складних запитів.

У розробці було використано об’єктно-орієнтований підхід, що сприяв формуванню логічної й гнучкої структури системи, а також полегшив підтримку та розвиток. За допомогою ORM-бібліотеки SQLAlchemy реалізовано моделі, які відображають усі основні освітні об’єкти, а для обробки запитів клієнта – REST API із відповідною авторизацією та перевіркою прав доступу. У системі передбачено підтримку ролей, перевірку прав на доступ до курсів та дій, пов’язаних із навчальними матеріалами.

Завдяки використанню Docker і супутніх інструментів (Docker Compose, Traefik) забезпечено просте розгортання системи в різних середовищах, що є важливою умовою масштабованості та підтримки в умовах реального використання. Протягом розробки активно застосовувалися принципи інкапсуляції, повторного використання коду та поділу відповідальностей.

Робота над проєктом виконувалась у команді з використанням системи керування завданнями Jira, що дозволило ефективно планувати спринти, відстежувати статус задач і забезпечити контроль за прогресом. Взаємодія між

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 86   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

учасниками команди та синхронізація змін здійснювалися через систему контролю версій Git, яка дозволила уникати конфліктів і забезпечити стабільну інтеграцію функціоналу.

У межах дипломного проєкту реалізовано комплексну, функціонально насичену й технічно стійку систему, що здатна підтримувати навчальний процес у вищому навчальному закладі. Створена платформа демонструє приклад поєднання сучасних технологій і методик розробки, орієнтованих на підвищення ефективності освітнього середовища.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 87   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Moodle. Система управління навчанням. URL: <https://moodle.org/>
2. Blackboard Learn. Система управління навчанням. URL: <https://www.blackboard.com/>
3. Sakai. Відкрита система управління навчанням. URL: <https://www.sakaiproject.org/>
4. Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley, 2003. 560 с.
5. FastAPI Documentation. Official Site. URL: <https://fastapi.tiangolo.com/>
6. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994. 395 с.
7. React. A JavaScript library for building user interfaces. URL: <https://reactjs.org/>
8. PostgreSQL Global Development Group. PostgreSQL: The World's Most Advanced Open Source Relational Database. URL: <https://www.postgresql.org/>
9. Docker Inc. What is a Container? URL: <https://www.docker.com/resources/what-container>
10. Moat A. Using Docker: Developing and Deploying Software with Containers. 1-е вид. Бірмінгем: O'Reilly Media, 2015. 354 с.
11. Docker Inc. Overview of Docker Compose. URL: <https://docs.docker.com/compose/>
12. Turnbull J. The Docker Book: Containerization is the new virtualization. James Turnbull, 2014. 312 с.
13. Docker Inc. Swarm mode overview. URL: <https://docs.docker.com/engine/swarm/>
14. Traefik Labs. Introduction to Traefik. URL: <https://doc.traefik.io/traefik/>
15. Hüttel H.-J. Foundations of Software Science and Computational Structures. Springer, 2018. 376 с.
16. Wiegers K., Beatty J. Software Requirements. 3rd ed. Microsoft Press, 2013. 560 с.
17. Sommerville I. Software Engineering. 10th ed. Pearson, 2015. 880 с.
18. Larman C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. 3rd ed. Prentice Hall, 2004. 736 с.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 88   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

19. Obe R. O., Hsu L. S. PostgreSQL: Up and Running. 3rd ed. O'Reilly Media, 2020. 322 c.
20. Tiangolo S. FastAPI documentation. URL: <https://fastapi.tiangolo.com/tutorial/sql-databases/>
21. Pydantic Documentation: BaseModel. URL: <https://docs.pydantic.dev/latest/usage/models/>
22. Colvin S. Pydantic: Data validation and settings management using Python type annotations. URL: <https://pydantic.dev/>
23. Ramírez S. FastAPI Documentation – Path Operation Decorators. URL: <https://fastapi.tiangolo.com/tutorial/path-params/>
24. Atlassian. Jira Software – Project and Issue Tracking. URL: <https://www.atlassian.com/software/jira>
25. Erl T. Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall, 2005. 792 c.
26. GeeksforGeeks. Client-Server Architecture. URL: <https://www.geeksforgeeks.org/system-design/client-server-model>
27. Richardson C. Microservices Patterns: With examples in Java. Manning Publications, 2018. 520 c.
28. Microsoft Learn. Introduction to REST APIs. URL: <https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design>
29. Myers G. J., Sandler C., Badgett T. The Art of Software Testing. 3rd ed. Wiley, 2011. 256 c.
30. Meszaros G. xUnit Test Patterns: Refactoring Test Code. Addison-Wesley, 2007. 896 c.
31. Joshi S. Software Testing Essentials: Foundation Guide. 2nd ed. Packt Publishing, 2024. 310 c.

ДОДАТКИ

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 90   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

## Моделі освітньої структури системи

```

class Faculty(Base):
    __tablename__ = "faculties"

    id = Column(UUID(as_uuid=True), primary_key=True, default=uuid4)
    name = Column(String, unique=True, nullable=False)

    dean_id = Column(UUID(as_uuid=True), ForeignKey("teachers.user_id", ondelete="SET NULL"), nullable=True)
    dean = relationship("Teacher", foreign_keys=[dean_id], backref="headed_faculty", uselist=False)
    departments = relationship("Department", back_populates="faculty")

class Department(Base):
    __tablename__ = "departments"

    id = Column(UUID(as_uuid=True), primary_key=True, default=uuid4)
    name = Column(String, unique=True, nullable=False)

    faculty_id = Column(UUID(as_uuid=True), ForeignKey("faculties.id", ondelete="CASCADE"), nullable=False)
    faculty = relationship("Faculty", back_populates="departments")
    head_id = Column(UUID(as_uuid=True), ForeignKey("teachers.user_id"), nullable=True)
    head = relationship("Teacher", foreign_keys=[head_id], lazy="joined")

class EducationalDegree(Base):
    __tablename__ = "educational_degrees"

    id = Column(UUID(as_uuid=True), primary_key=True, default=uuid4)
    name = Column(String, unique=True, nullable=False)

class EducationForm(Base):
    __tablename__ = "education_forms"

    id = Column(UUID(as_uuid=True), primary_key=True, default=uuid4)
    name = Column(String, unique=True, nullable=False)

class Specialty(Base):
    __tablename__ = "specialties"

    id = Column(UUID(as_uuid=True), primary_key=True, default=uuid4)
    code = Column(String, unique=True, nullable=True)
    name = Column(String, unique=True, nullable=False)

    department_id = Column(UUID(as_uuid=True), ForeignKey("departments.id", ondelete="CASCADE"), nullable=True)
    department = relationship("Department")

    programs = relationship("EducationProgram", back_populates="specialty")

class Group(Base):

```

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 91   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

```
__tablename__ = "groups"
```

```
id = Column(UUID(as_uuid=True), primary_key=True, default=uuid4)
```

```
name = Column(String, unique=True, nullable=False)
```

```
education_program_id = Column(
```

```
    UUID(as_uuid=True), ForeignKey("education_programs.id", ondelete="CASCADE"), nullable=False
)
```

```
year = Column(Integer, nullable=True)
```

```
education_program = relationship("EducationProgram", back_populates="groups")
```

```
class EducationProgram(Base):
```

```
    __tablename__ = "education_programs"
```

```
id = Column(UUID(as_uuid=True), primary_key=True, default=uuid4)
```

```
name = Column(String, unique=True, nullable=False)
```

```
specialty_id = Column(UUID(as_uuid=True), ForeignKey("specialties.id", ondelete="CASCADE"), nullable=False)
```

```
courses = relationship("Course", back_populates="education_program", cascade="all, delete")
```

```
specialty = relationship("Specialty", back_populates="programs")
```

```
groups = relationship("Group", back_populates="education_program")
```

```
class Test(Base):
```

```
    __tablename__ = "tests"
```

```
id = Column(UUID(as_uuid=True), primary_key=True, default=uuid4)
```

```
assignment_id = Column(UUID(as_uuid=True), ForeignKey("assignments.id"), nullable=False)
```

```
title = Column(String, nullable=False)
```

```
description = Column(Text, nullable=True)
```

```
time_limit_minutes = Column(Integer, nullable=True)
```

```
attempts_allowed = Column(Integer, nullable=True)
```

```
shuffle_questions = Column(Boolean, default=False, nullable=False)
```

```
shuffle_answers = Column(Boolean, default=False, nullable=False)
```

```
show_correct_answers = Column(
    Boolean, default=False, nullable=False
)
```

```
show_score_after = Column(String, default="never", nullable=False)
```

```
show_answers_after = Column(String, default="never", nullable=False)
```

```
browser_restrictions = Column(Boolean, default=False, nullable=False)
```

```
result_view_type = Column(String, default="detailed", nullable=False)
```

```
schedule_type = Column(String, default="flexible", nullable=False)
```

```
available_from = Column(DateTime, nullable=True)
```

```
available_to = Column(DateTime, nullable=True)
```

```
questions = relationship("Question", back_populates="test", cascade="all, delete")
```

```
attempts = relationship("TestAttempt", back_populates="test", cascade="all, delete")
```

```
assignment = relationship("Assignment", back_populates="test")
```

```
class TestAttempt(Base):
```

```
    __tablename__ = "test_attempts"
```

```
id = Column(UUID(as_uuid=True), primary_key=True, default=uuid4)
```

```
started_at = Column(DateTime, default=datetime_now_sec, nullable=False)
```

```
finished_at = Column(DateTime, nullable=True)
```

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 92   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

```

score = Column(Integer, nullable=True)
answers_data = Column(JSONB, nullable=True)
time_spent_seconds = Column(Integer, nullable=True)
time_spent_out_of_tab = Column(Integer, nullable=True)
status = Column(String, nullable=False, default="in_progress")
suspicious_activity = Column(Boolean, default=False, nullable=False)
manual_review = Column(Boolean, default=False, nullable=False)
review_notes = Column(Text, nullable=True)
questions_order = Column(Text, nullable=True)

test_id = Column(UUID(as_uuid=True), ForeignKey("tests.id"), nullable=False)
student_id = Column(UUID(as_uuid=True), ForeignKey("students.user_id"), nullable=False)

test = relationship("Test", back_populates="attempts")
student = relationship("Student")

```

```

class Question(Base):
    __tablename__ = "test_questions"

    id = Column(UUID(as_uuid=True), primary_key=True, default=uuid4)
    question_text = Column(Text, nullable=False)
    question_type = Column(String, nullable=False)
    points = Column(Integer, nullable=False)
    order = Column(Integer, nullable=False)
    explanation = Column(Text, nullable=True)
    media_url = Column(String, nullable=True)
    media_type = Column(String, nullable=True)
    difficulty = Column(String, nullable=True)
    hint = Column(Text, nullable=True)
    is_active = Column(Boolean, default=True, nullable=False)

    test_id = Column(UUID(as_uuid=True), ForeignKey("tests.id"), nullable=False)

    test = relationship("Test", back_populates="questions")
    answer_options = relationship("AnswerOption", back_populates="question", cascade="all, delete")

```

```

class AnswerOption(Base):
    __tablename__ = "answer_options"

    id = Column(UUID(as_uuid=True), primary_key=True, default=uuid4)
    option_text = Column(Text, nullable=False)
    is_correct = Column(Boolean, default=False, nullable=False)
    order = Column(Integer, nullable=False)
    media_url = Column(String, nullable=True)
    partial_score = Column(Float, nullable=True)
    (Text, nullable=True)

    question_id = Column(UUID(as_uuid=True), ForeignKey("test_questions.id"), nullable=False)

    question = relationship("Question", back_populates="answer_options")

```

```

class Assignment(Base):
    __tablename__ = "assignments"

    id = Column(UUID(as_uuid=True), primary_key=True, default=uuid4)

```

```

title = Column(String, nullable=False)
description = Column(Text, nullable=True)
pdf_url = Column(String, nullable=True)
assignment_type = Column(String, nullable=False)
max_score = Column(Integer, nullable=False)
koefcicient = Column(Float, default=1.0, nullable=False)
deadline = Column(DateTime, nullable=True)
miss_deadline_koefcicient = Column(Float, default=0.8, nullable=False)
is_required = Column(Boolean, default=True, nullable=False)
order = Column(Integer, nullable=False)
submission_type = Column(String, nullable=False, default="text")

light_remove = Column(Boolean, default=False, nullable=False)
resubmission_limit = Column(Integer, nullable=True)
grading_criteria = Column(Text, nullable=True)
is_shown = Column(Boolean, default=True, nullable=False)

lesson_id = Column(
    UUID(as_uuid=True), ForeignKey("lessons.id"), nullable=True
)
course_id = Column(UUID(as_uuid=True), ForeignKey("courses.id"), nullable=True)

lesson = relationship("Lesson", back_populates="assignments")
course = relationship("Course")
test = relationship("Test", back_populates="assignment", uselist=False, cascade="all, delete-orphan")
submissions = relationship("Submission", back_populates="assignment")
grades = relationship("Grade", back_populates="assignment", cascade="all, delete")

```

```
class Submission(Base):
```

```
    __tablename__ = "submissions"
```

```

id = Column(UUID(as_uuid=True), primary_key=True, default=uuid4)
student_id = Column(UUID(as_uuid=True), ForeignKey("students.user_id"), nullable=False)
assignment_id = Column(UUID(as_uuid=True), ForeignKey("assignments.id"), nullable=False)
submitted_at = Column(DateTime, default=datetime.utcnow, nullable=False)
text_answer = Column(Text, nullable=True)
file_urls = Column(Text, nullable=True) # JSON
comment_student = Column(Text, nullable=True)
comment_teacher = Column(Text, nullable=True)
version = Column(Integer, default=1, nullable=False)
status = Column(String, default="submitted", nullable=False)
light_remove = Column(Boolean, default=False, nullable=False)

```

```

assignment = relationship("Assignment")
student = relationship("Student")

```

```
class Role(Base):
```

```
    __tablename__ = "roles"
```

```

id = Column(UUID(as_uuid=True), primary_key=True, default=uuid4, index=True)
name = Column(String, unique=True, nullable=False)

```

```
users = relationship("User", back_populates="role")
```

```

def __repr__(self):
    return f"<Role(id={self.id}, name={self.name})>"

```

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 94   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

```

class DocumentTemplate(Base):
    __tablename__ = "document_templates"

    id = Column(UUID(as_uuid=True), primary_key=True, default=uuid4)
    name = Column(String, nullable=False)
    description = Column(Text, nullable=True)
    file_name = Column(String, nullable=False)
    requires_approval = Column(Boolean, default=False)
    document_type = Column(String, nullable=False,
                             default="certificate")
    template_variables = Column(ARRAY(String), nullable=True)
    created = Column(DateTime, default=datetime_now_sec)

class UserDocumentRequest(Base):
    __tablename__ = "document_requests"

    id = Column(UUID(as_uuid=True), primary_key=True, default=uuid4)

    user_id = Column(UUID(as_uuid=True), ForeignKey(
        "users.id"), nullable=False)
    user = relationship("User", foreign_keys=[user_id], backref="document_requests")

    template_id = Column(UUID(as_uuid=True), ForeignKey(
        "document_templates.id"), nullable=False)
    template = relationship("DocumentTemplate")

    status = Column(PgEnum(DocumentStatus),
                    default=DocumentStatus.PENDING, nullable=False)

    generation_status = Column(String, default="pending")

    filled_data = Column(JSONB, nullable=True)
    comment = Column(Text, nullable=True)
    rejection_reason = Column(Text, nullable=True)

    document_url = Column(String, nullable=True)
    viewed_by_user = Column(Boolean, default=False)

    created = Column(DateTime, default=datetime_now_sec)
    updated = Column(DateTime, default=datetime_now_sec,
                     onupdate=datetime_now_sec)
    generated_at = Column(DateTime, nullable=True)

    approved_by_id = Column(
        UUID(as_uuid=True), ForeignKey("users.id"), nullable=True)
    approved_by = relationship("User", foreign_keys=[approved_by_id])

```

## Базовий клас роботи з БД

```

class CRUDBase(Generic[ModelType, CreateSchemaType, UpdateSchemaType]):
    def __init__(self, model: Type[ModelType]):

        self.model = model

    async def get(self, db: AsyncSession, id: Any, load_relations: list = None) -> Optional[ModelType] | None:
        stmt = select(self.model).filter(self.model.id == id)

        if load_relations:
            for relation in load_relations:
                relation_property = getattr(self.model, relation, None)
                if relation_property:
                    stmt = stmt.options(selectinload(relation_property))

        result = await db.execute(stmt)
        return result.scalar_one_or_none()

    async def get_by_name(self, db: AsyncSession, name: str) -> Optional[ModelType] | None:
        stmt = select(self.model).filter(self.model.name == name)
        result = await db.execute(stmt)
        return result.scalar_one_or_none()

    async def get_by_user_id(self, db: AsyncSession, user_id: str) -> Optional[ModelType] | None:
        stmt = select(self.model).filter(self.model.user_id == user_id)
        result = await db.execute(stmt)
        return result.scalar_one_or_none()

    async def get_multi(self, db: AsyncSession, *, page: int = 0, page_break: bool = False) -> list[ModelType]:
        stmt = select(self.model)

        if page_break:
            skip = page * settings.MULTI_MAX
            limit = settings.MULTI_MAX
            stmt = stmt.offset(skip).limit(limit)

        result = await db.execute(stmt)
        return result.scalars().all()

    async def create(self, db: AsyncSession, *, obj_in: CreateSchemaType, load_relations: list = None) -> ModelType:
        obj_in_data = obj_in.dict(exclude_unset=True)
        db_obj = self.model(**obj_in_data)
        db.add(db_obj)

        await db.commit()
        await db.refresh(db_obj)

```

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 96   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

```

if load_relations:
    query = select(self.model).where(self.model.id == db_obj.id)

    for relation in load_relations:
        relation_property = getattr(self.model, relation, None)
        query = query.options(selectinload(relation_property))

    result = await db.execute(query)
    db_obj = result.scalars().first()

return db_obj

async def update(
    self, db: AsyncSession, *, db_obj: ModelType, obj_in: Union[UpdateSchemaType, Dict[str, Any]],
    load_relations: list = None
) -> ModelType:
    if isinstance(obj_in, dict):
        update_data = obj_in
        obj_data = obj_in
    else:
        update_data = obj_in.dict(exclude_unset=True)
        obj_data = update_data

    for field in obj_data:
        if field in update_data:
            setattr(db_obj, field, update_data[field])

    db.add(db_obj)
    await db.commit()
    await db.refresh(db_obj)

    if load_relations:
        query = select(self.model).where(self.model.id == db_obj.id)

        for relation in load_relations:
            relation_property = getattr(self.model, relation, None)
            query = query.options(selectinload(relation_property))

        result = await db.execute(query)
        db_obj = result.scalars().first()

    return db_obj

async def remove(self, db: AsyncSession, *, id: int) -> Optional[ModelType]:
    db_obj = await self.get(db, id)

    if db_obj:
        await db.delete(db_obj)
        await db.commit()

```

```
return db_obj
```

```
async def check_existing(self, db: AsyncSession, **kwargs) -> bool:
```

```
    query = select(self.model).filter_by(**kwargs)
```

```
    result = await db.execute(query)
```

```
    return result.scalar_one_or_none() is not None
```

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 98   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

## Файл конфігурації Docker Compose

```

services:
  proxy:
    image: traefik:v2.2
    networks:
      - ${TRAEFIK_PUBLIC_NETWORK?Variable not set}
      - default
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    command:
      - --providers.docker
      - --providers.docker.constraints=Label(`traefik.constraint-label-stack`, `${TRAEFIK_TAG?Variable not set}`)
      - --providers.docker.exposedbydefault=false
      - --providers.docker.swarmmode
      - --accesslog
      - --log
      - --api
    deploy:
      placement:
        constraints:
          - node.role == manager
      labels:
        - traefik.enable=true
        - traefik.docker.network=${TRAEFIK_PUBLIC_NETWORK?Variable not set}
        - traefik.constraint-label=${TRAEFIK_PUBLIC_TAG?Variable not set}
        - traefik.http.middlewares.${STACK_NAME?Variable not set}-https-redirect.redirect-scheme=https
        - traefik.http.middlewares.${STACK_NAME?Variable not set}-https-redirect.redirectschemepermanent=true

        - traefik.http.routers.${STACK_NAME?Variable not set}-proxy-http.rule=Host(`${DOMAIN?Variable not set}`) || Host(`www.${DOMAIN?Variable not set}`)
        - traefik.http.routers.${STACK_NAME?Variable not set}-proxy-http.entrypoints=http
        - traefik.http.routers.${STACK_NAME?Variable not set}-proxy-https.rule=Host(`${DOMAIN?Variable not set}`) || Host(`www.${DOMAIN?Variable not set}`)
        - traefik.http.routers.${STACK_NAME?Variable not set}-proxy-https.entrypoints=https
        - traefik.http.routers.${STACK_NAME?Variable not set}-proxy-https.tls=true
        - traefik.http.routers.${STACK_NAME?Variable not set}-proxy-https.tls.certresolver=le
        - traefik.http.services.${STACK_NAME?Variable not set}-proxy.loadbalancer.server.port=80
        - traefik.http.middlewares.${STACK_NAME?Variable not set}-www-redirect.redirectregex.regex=https://(www.)?(${DOMAIN?Variable not set})/(.*)

        - traefik.http.middlewares.${STACK_NAME?Variable not set}-www-redirect.redirectregex.replace=https://${DOMAIN?Variable not set}/${3}
        - traefik.http.routers.${STACK_NAME?Variable not set}-proxy-https.middlewares=${STACK_NAME?Variable not set}-www-redirect
        - traefik.http.routers.${STACK_NAME?Variable not set}-proxy-http.middlewares=${STACK_NAME?Variable not set}-www-redirect,${STACK_NAME?Variable not set}-https-redirect

```

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 99   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

```

queue:
  image: rabbitmq:3
  volumes:
    - ./conf/rabbitmq.conf:/etc/rabbitmq/rabbitmq.conf

flower:
  image: mher/flower:0.9.7
  networks:
    - ${TRAEFIK_PUBLIC_NETWORK?Variable not set}
    - default
  env_file:
    - .env
  command:
    - "--broker=amqp://guest@queue:5672/"
  deploy:
    labels:
      - traefik.enable=true
      - traefik.docker.network=${TRAEFIK_PUBLIC_NETWORK?Variable not set}
      - traefik.constraint-label=${TRAEFIK_PUBLIC_TAG?Variable not set}
      - traefik.http.routers.${STACK_NAME?Variable not set}-flower-http.rule=Host(`flower.${DO-
MAIN?Variable not set}`)
      - traefik.http.routers.${STACK_NAME?Variable not set}-flower-http.entrypoints=http
      - traefik.http.routers.${STACK_NAME?Variable not set}-flower-http.middle-
wares=${STACK_NAME?Variable not set}-https-redirect
      - traefik.http.routers.${STACK_NAME?Variable not set}-flower-https.rule=Host(`flower.${DO-
MAIN?Variable not set}`)
      - traefik.http.routers.${STACK_NAME?Variable not set}-flower-https.entrypoints=https
      - traefik.http.routers.${STACK_NAME?Variable not set}-flower-https.tls=true
      - traefik.http.routers.${STACK_NAME?Variable not set}-flower-https.tls.certresolver=le
      - traefik.http.services.${STACK_NAME?Variable not set}-flower.loadbalancer.server.port=5555

backend:
  image: "${DOCKER_IMAGE_BACKEND?Variable not set}:${TAG-latest}"
  volumes:
    - ./backend/app/alembic:/app/alembic
    - ./backend:/backend
  logging:
    driver: "json-file"
    options:
      max-size: "200k"
      max-file: "3"
  env_file:
    - .env
  environment:
    - SERVER_NAME=${DOMAIN?Variable not set}
    - SERVER_HOST=https://${DOMAIN?Variable not set}
  build:
    context: ./backend
    dockerfile: backend.dockerfile

```

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 25 – ПЗ | Арк. |
|      |      |          |        |      |                          | 100  |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

```

args:
  BACKEND_APP_MODULE: ${BACKEND_APP_MODULE-app.main:app}
  BACKEND_PRE_START_PATH: ${BACKEND_PRE_START_PATH-/app/scripts/prestart.sh}
  BACKEND_PROCESS_MANAGER: ${BACKEND_PROCESS_MANAGER-gunicorn}
  BACKEND_WITH_RELOAD: ${BACKEND_WITH_RELOAD-false}
  INSTALL_DEV: ${INSTALL_DEV-false}
deploy:
  labels:
    - traefik.enable=true
    - traefik.constraint-label-stack=${TRAEFIK_TAG?Variable not set}
    - traefik.http.routers.${STACK_NAME?Variable not set}-backend-http.rule=PathPrefix(`/api`) || PathPrefix(`/docs`) || PathPrefix(`/redoc`)
    - traefik.http.services.${STACK_NAME?Variable not set}-backend.loadbalancer.server.port=80

celeryworker:
  image: "${DOCKER_IMAGE_CELERYWORKER?Variable not set}:${TAG-latest}"
  logging:
    driver: "json-file"
    options:
      max-size: "200k"
      max-file: "3"
  env_file:
    - .env
  environment:
    - SERVER_NAME=${DOMAIN?Variable not set}
    - SERVER_HOST=https://${DOMAIN?Variable not set}
  build:
    context: ./backend
    dockerfile: celeryworker.dockerfile
  args:
    INSTALL_DEV: ${INSTALL_DEV-false}

frontend:
  image: "${DOCKER_IMAGE_FRONTEND?Variable not set}:${TAG-latest}"
  env_file:
    - .env
  build:
    context: ./frontend
  deploy:
    labels:
      - traefik.enable=true
      - traefik.constraint-label-stack=${TRAEFIK_TAG?Variable not set}
      - traefik.http.routers.${STACK_NAME?Variable not set}-frontend-http.rule=PathPrefix(`/`)
      - traefik.http.services.${STACK_NAME?Variable not set}-frontend.loadbalancer.server.port=80

db:
  image: postgres:12
  volumes:
    - app-db-data:/var/lib/postgresql/data/pgdata

```

```

env_file:
- .env
environment:
- PGDATA=/var/lib/postgresql/data/pgdata
deploy:
placement:
constraints:
- node.labels.${STACK_NAME?Variable not set}.app-db-data == true

```

```

pgadmin:
image: dpage/pgadmin4
networks:
- ${TRAEFIK_PUBLIC_NETWORK?Variable not set}
- default
env_file:
- .env
deploy:
labels:
- traefik.enable=true
- traefik.docker.network=${TRAEFIK_PUBLIC_NETWORK?Variable not set}
- traefik.constraint-label=${TRAEFIK_PUBLIC_TAG?Variable not set}
- traefik.http.routers.${STACK_NAME?Variable not set}-pgadmin-http.rule=Host(`pgadmin.${DO-
MAIN?Variable not set}`)
- traefik.http.routers.${STACK_NAME?Variable not set}-pgadmin-http.entrypoints=http
- traefik.http.routers.${STACK_NAME?Variable not set}-pgadmin-http.middle-
wares=${STACK_NAME?Variable not set}-https-redirect
- traefik.http.routers.${STACK_NAME?Variable not set}-pgadmin-https.rule=Host(`pgadmin.${DO-
MAIN?Variable not set}`)
- traefik.http.routers.${STACK_NAME?Variable not set}-pgadmin-https.entrypoints=https
- traefik.http.routers.${STACK_NAME?Variable not set}-pgadmin-https.tls=true
- traefik.http.routers.${STACK_NAME?Variable not set}-pgadmin-https.tls.certresolver=le
- traefik.http.services.${STACK_NAME?Variable not set}-pgadmin.loadbalancer.server.port=5050

```

```

volumes:
app-db-data:

```

```

networks:
traefik-public:
external: ${TRAEFIK_PUBLIC_NETWORK_IS_EXTERNAL=true}

```