НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО **Декан факультету**	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ** **Завідувач кафедри інформаційних систем і технологій**
_____ Ігор БОЛБОТ	**_____ Михайло ШВИДЕНКО**
«___» _____ 2026 р.	«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Інформаційна система підтримки надання освітніх послуг з повторного вивчення дисциплін студентами структурного підрозділі університету**»**

Спеціальність 126 «Інформаційні системи та технології»
Освітньо-професійна програма «Інформаційні системи та технології»

Гарант освітньої програми
к.е.н., доцент **_____ Максим МОКРІЄВ**

Керівник бакалаврської кваліфікаційної роботи
д.пед.н., професор **_____ Олена КУЗЬМІНСЬКА**

Виконав **_____ Владислав КИРИЧЕНКО**

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри інформаційних систем і технологій

к.е.н., доцент _____ Михайло ШВИДЕНКО

«___» _____ 2026 р.

ЗАВДАННЯ ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ ЗДОБУВАЧУ

Кириченку Владиславу Сергійовичу

Спеціальність 126«Інформаційні системи та технології»

ОП «Інформаційні системи та технології»

Тема бакалаврської кваліфікаційної роботи «Інформаційна система підтримки надання освітніх послуг з повторного вивчення дисциплін студентами структурного підрозділу університету» затверджена наказом від «19» грудень 2025 р. № 3205 «С2».

Термін подання завершеної роботи на кафедру «25» травня 2026 р.

Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту): система забезпечує централізований облік заяв, контроль відповідності встановленим вимогам, фіксацію управлінських рішень, координацію взаємодії між підрозділами та формування супровідної документації.

Перелік питань, що підлягають дослідженню:

- Аналіз предметної області організації повторного вивчення дисциплін у закладах вищої освіти та проаналізувати існуючі програмні рішення;
- Визначення вимог до інформаційної системи та моделювання її основних процесів
- Проєктування структури бази даних, архітектури та програмних модулів програми

Дата видачі завдання «06» січня 2026 р.

Керівник бакалаврської кваліфікаційної роботи _____ **Олена КУЗЬМІНСЬКА**

Завдання взяв до виконання _____ **Владислав КИРИЧЕНКО**

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

1. API – Application Programming Interface, програмний інтерфейс прикладного програмування
2. BPMN – Business Process Model and Notation, нотація моделювання бізнес-процесів
3. CSV – Comma-Separated Values, текстовий формат подання табличних даних
4. CRUD – Create, Read, Update, Delete, базові операції створення, перегляду, редагування та видалення даних
5. DB – Database, база даних
6. DFD – Data Flow Diagram, діаграма потоків даних
7. ER – Entity-Relationship, модель «сутність–зв’язок»
8. FK – Foreign Key, зовнішній ключ
9. GUI – Graphical User Interface, графічний інтерфейс користувача
10. PK – Primary Key, первинний ключ
11. QSS – Qt Style Sheets, механізм стилізації інтерфейсу в Qt-застосунках
12. RBAC – Role-Based Access Control, рольова модель керування доступом
13. SQL – Structured Query Language, мова структурованих запитів
14. SQLite – вбудована реляційна система керування базами даних
15. UML – Unified Modeling Language, уніфікована мова моделювання
16. БД – база даних
17. ЗВО – заклад вищої освіти
18. ІС – інформаційна система
19. ОС – операційна система
20. ПЗ – програмне забезпечення
21. СКБД – система керування базами даних
22. UI – User Interface, користувацький інтерфейс

ВСТУП

Академічна мобільність студентів, різноманітність форм контролю результатів навчання та необхідність дотримання внутрішніх регламентів університету зумовлюють підвищені вимоги до управління освітніми послугами, що надаються студентам у разі повторного опанування навчальних компонентів.

Процес повторного вивчення дисциплін охоплює комплекс взаємопов’язаних дій, включаючи фіксацію академічної заборгованості, ініціювання студентом відповідної процедури, перевірку підстав для повторного навчання, організацію навчального процесу та контроль результатів повторного оцінювання. За відсутності спеціалізованих інформаційних засобів зазначені процеси часто виконуються із застосуванням фрагментарних рішень або вручну, що призводить до неузгодженості даних, ускладнення комунікації між учасниками освітнього процесу та зниження оперативності прийняття управлінських рішень.

Традиційні підходи до підтримки повторного вивчення дисциплін, як правило, не забезпечують цілісного інформаційного супроводу на всіх етапах надання відповідної освітньої послуги. Відсутність централізованої системи обліку заяв, навчальних навантажень, фінансових аспектів та результатів повторного контролю ускладнює моніторинг виконання регламентів і створює додаткове навантаження на адміністративний персонал структурного підрозділу університету.

Використання сучасних інформаційних систем, побудованих на основі веб-технологій та клієнт–серверної архітектури, відкриває можливості для автоматизації процесів підтримки повторного вивчення дисциплін. Інтеграція механізмів збору, зберігання та оброблення освітніх даних дозволяє забезпечити прозорість процедур, зменшити кількість помилок, а також підвищити ефективність взаємодії між студентами, викладачами та адміністрацією структурного підрозділу.

Метою кваліфікаційної роботи є проєктування та розробка інформаційної системи підтримки надання освітніх послуг з повторного вивчення дисциплін студентами структурного підрозділу університету з використанням сучасних інформаційних технологій.

Для досягнення поставленої мети в роботі передбачається розв'язання таких **завдань:**

1. дослідити предметну область організації повторного вивчення дисциплін у закладах вищої освіти та проаналізувати існуючі програмні рішення;
2. визначити функціональні, нефункціональні та вимоги до розроблюваної системи;
3. спроєктувати архітектуру програмного забезпечення з урахуванням вимог до масштабованості, надійності та модульності;
4. розробити UML-моделі, що описують основні процеси та компоненти інформаційної системи;
5. реалізувати програмний прототип інформаційної системи для автоматизації процедур повторного вивчення дисциплін;
6. провести тестування системи та оцінити ефективність її використання в діяльності структурного підрозділу університету.

Об'єктом дослідження є процес організації та надання освітніх послуг з повторного вивчення дисциплін у закладах вищої освіти.

Предметом дослідження є методи, моделі та програмні засоби створення інформаційних систем підтримки повторного вивчення навчальних дисциплін.

Методи дослідження включають застосування системного аналізу для вивчення предметної області, UML-моделювання для формалізації процесів і структури системи, а також методи об'єктно-орієнтованого проєктування при реалізації програмного забезпечення. Для забезпечення взаємодії між компонентами системи використано клієнт–серверну архітектуру та REST-підхід, а оцінювання результатів здійснюється за допомогою методів експериментального тестування.

Практична значущість одержаних результатів полягає у можливості впровадження розробленої інформаційної системи в діяльність структурних підрозділів закладів вищої освіти. Система забезпечує автоматизацію обліку заяв на повторне вивчення дисциплін, контроль виконання регламентів та формування аналітичної інформації для прийняття управлінських рішень. Результати роботи можуть бути використані для підвищення якості організації освітніх послуг та оптимізації внутрішніх процесів університету.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

У межах предметної області система розглядається як програмно-організаційний комплекс, призначений для автоматизації процесів подання, розгляду, погодження та супроводу повторного вивчення навчальних дисциплін студентами структурного підрозділу університету. Система забезпечує централізований облік заяв, контроль відповідності встановленим вимогам, фіксацію управлінських рішень, координацію взаємодії між підрозділами та формування супровідної документації.

Функціональне призначення системи полягає у реалізації повного життєвого циклу освітньої послуги з повторного вивчення дисципліни: від моменту подання студентом заяви до завершення повторного контролю та фіксації результатів. Система підтримує перевірку академічних підстав, погодження з навчальною частиною, взаємодію з кафедрою або викладачем, а також, за необхідності, облік фінансових операцій. Інтеграція з внутрішніми інформаційними ресурсами університету забезпечує актуальність даних щодо студентів, навчальних планів і результатів оцінювання.

На рис. 1.1 подано DFD-діаграму 0-го рівня інформаційної системи підтримки повторного вивчення дисципліни, яка відображає взаємодію зовнішніх сутностей із центральним процесом «P0 – ІС підтримки повторного вивчення дисципліни». До зовнішніх сутностей належать Студент, Навчальна частина (Деканат), Викладач / Кафедра та Фінансовий відділ (Бухгалтерія). Основними потоками даних є заява на повторне вивчення, рішення щодо її розгляду, запити на погодження, підтвердження оплати (за потреби), організаційні параметри та результати повторного контролю.

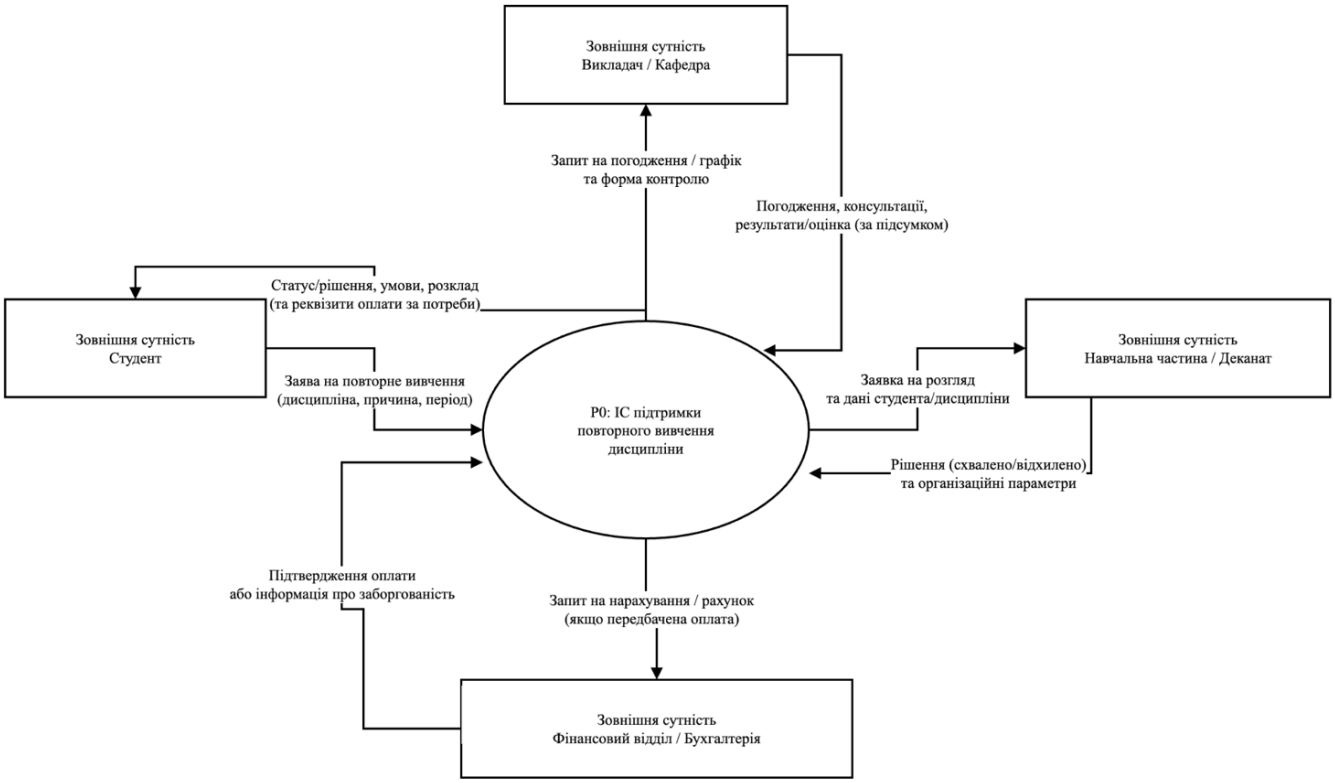

Рисунок 1.1 – DFD 0-рівня інформаційної системи підтримки повторного вивчення дисципліни

На рис. 1.2 наведено BPMN-діаграму бізнес-процесу надання освітньої послуги з повторного вивчення дисципліни. Процес ініціюється студентом шляхом подання електронної заяви. Далі здійснюється перевірка умов і даних, після чого приймається рішення про допуск до процедури. У разі позитивного рішення формується відповідний наказ або запис у системі. При оплаті, система генерує запит до фінансового підрозділу та очікує підтвердження платежу. Після виконання всіх умов призначається повторне вивчення дисципліни із визначенням графіка, форми контролю та відповідального викладача. У разі відмови студент отримує відповідне повідомлення із зазначенням причин.

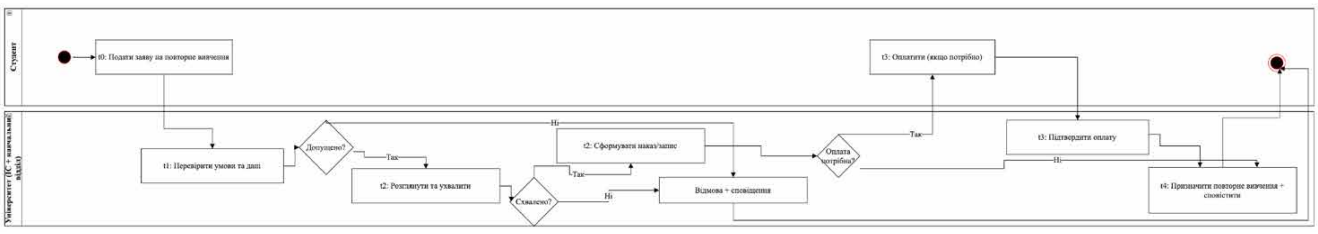

Рисунок 1.2 – BPMN-діаграма процесу підтримки повторного вивчення дисципліни

Для формалізації предметної області в табл. 1.1 наведено ключові сутності системи, їх функціональне призначення та типи даних, що обробляються.

Таблиця 1.1 – Основні сутності предметної області інформаційної системи підтримки повторного вивчення дисципліни

Сутність	Опис	Функціональне призначення	Типи даних
Студент	Особа, яка навчається у структурному підрозділі університету та має академічну заборгованість	Подання заяви на повторне вивчення, отримання статусу розгляду, виконання фінансових зобов'язань (за потреби), проходження повторного контролю	Персональні дані, дані про дисципліну, статус заяви, результати оцінювання
Навчальна частина / Деканат	Адміністративний підрозділ, відповідальний за організацію освітнього процесу	Перевірка підстав для повторного вивчення, прийняття рішення (схвалено/відхилено), визначення організаційних параметрів	Дані про навчальні плани, академічну заборгованість, управлінські рішення
Викладач / Кафедра	Підрозділ або особа, що забезпечує проведення повторного навчання та контролю	Погодження графіка, визначення форми контролю, проведення консультацій, фіксація результатів	Дані дисципліни, розклад, результати повторного оцінювання
Фінансовий відділ / Бухгалтерія	Підрозділ, що здійснює облік фінансових операцій	Нарахування оплати (за наявності), підтвердження платежу або інформування про заборгованість	Платіжні реквізити, статус оплати, фінансові записи
Центральний процес P0	Програмна підсистема підтримки повторного вивчення	Координація взаємодії між сутностями, оброблення заяв, контроль етапів процедури	Реєстри заяв, журнали подій, статуси, організаційні параметри

Предметна область інформаційної системи підтримки повторного вивчення дисципліни характеризується необхідністю чіткої регламентації процедур, координації взаємодії між адміністративними та навчальними підрозділами, а також забезпечення прозорості прийняття рішень. Автоматизація зазначених процесів дозволяє скоротити час розгляду заяв, мінімізувати ризик помилок, забезпечити контроль дотримання внутрішніх нормативів та підвищити якість надання освітніх послуг у структурному підрозділі університету.

1.2 Аналіз існуючих рішень

Особливу увагу приділено системам, що забезпечують електронний документообіг, управління академічною заборгованістю, організацію додаткових освітніх послуг та взаємодію між студентами, кафедрами й адміністрацією.

Серед найбільш поширених рішень, які можуть бути використані або адаптовані для підтримки процесів повторного вивчення дисциплін, доцільно виділити Moodle, Canvas LMS, Blackboard Learn, PeopleSoft Campus Solutions та Odoo Education. Кожне із зазначених рішень реалізує власний підхід до автоматизації освітнього циклу та управління студентськими сервісами.

На рис. 1.3 подано інтерфейс системи Moodle, яка є однією з найпоширеніших LMS із відкритим кодом. Moodle забезпечує модульну архітектуру, підтримку плагінів та гнучке налаштування навчальних курсів. У контексті повторного вивчення дисциплін система дозволяє створювати окремі курси для студентів із академічною заборгованістю, призначати додаткові завдання та контролювати результати повторного оцінювання. Водночас Moodle не містить спеціалізованого механізму комплексної автоматизації процесу подання та погодження заяв на повторне вивчення дисципліни.

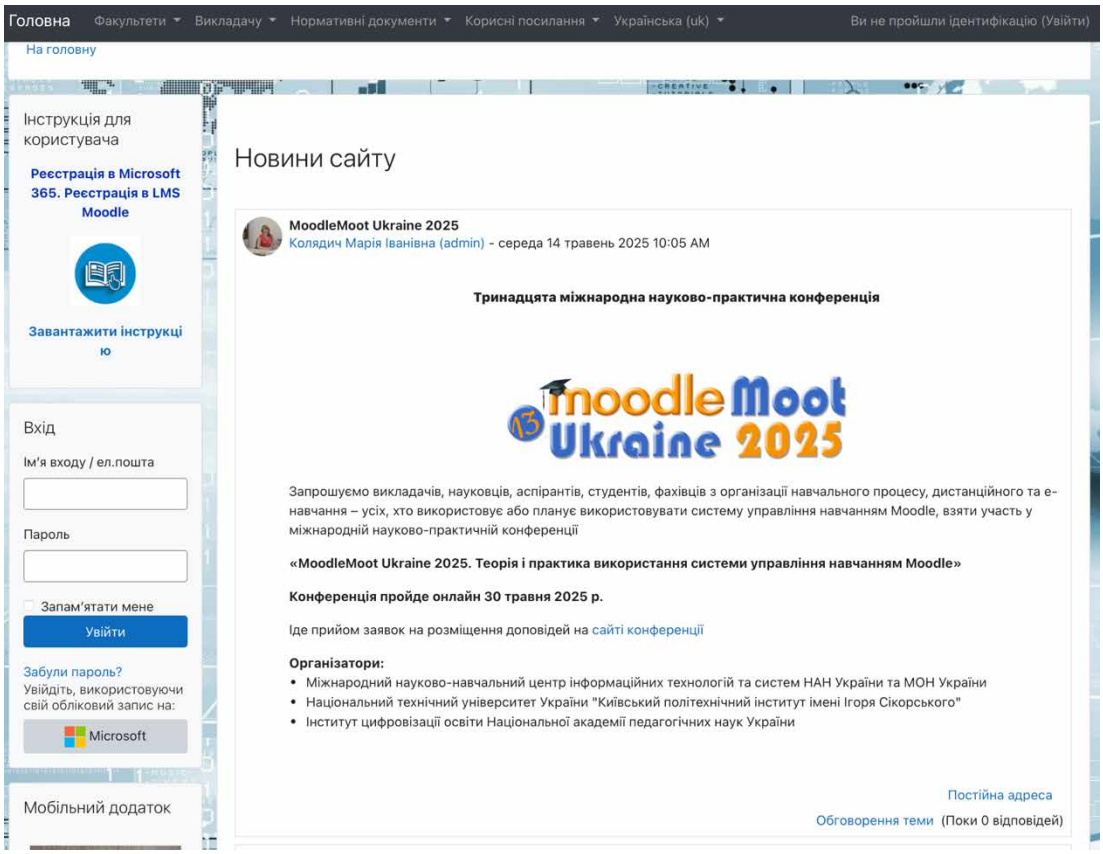

Рисунок 1.3 – Інтерфейс системи Moodle (приклад організації навчального середовища)

На рис. 1.4 представлено інтерфейс Canvas LMS — хмарної системи управління навчанням, що підтримує REST-орієнтовану архітектуру та розширені засоби аналітики. Canvas дозволяє формувати індивідуальні освітні траєкторії, управляти завданнями та контролювати прогрес студентів. Однак функціональність системи переважно орієнтована на навчальний контент і не передбачає детального регламентованого процесу адміністративного погодження повторного вивчення дисциплін у межах структурного підрозділу.

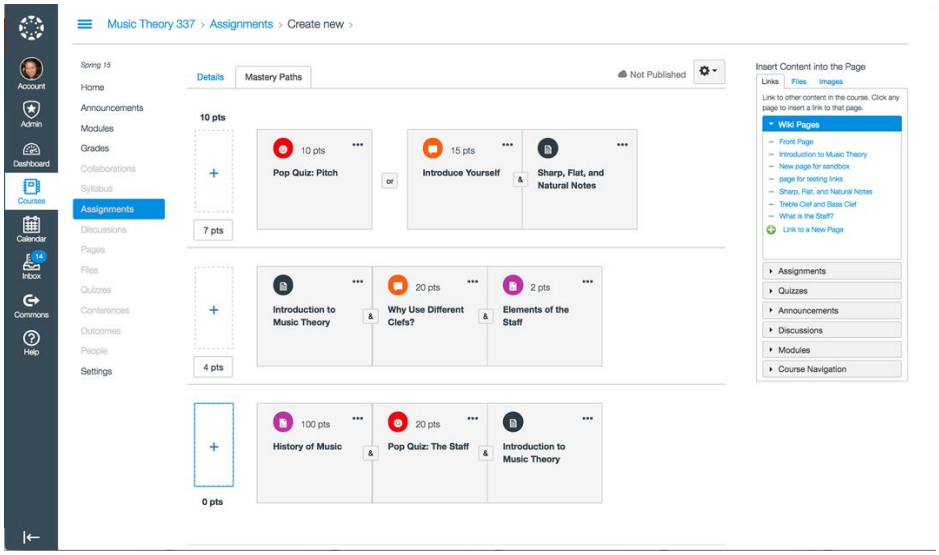

Рисунок 1.4 – Панель керування курсами в Canvas LMS

Система Blackboard Learn, зображена на рис. 1.5, орієнтована на великі освітні установи та забезпечує централізоване управління курсами, оцінюванням і аналітичними звітами. Вона підтримує інтеграцію з адміністративними модулями та системами управління даними студентів. Разом із тим складність налаштування та висока вартість ліцензування обмежують її використання як спеціалізованого рішення для автоматизації окремої послуги — повторного вивчення дисципліни.

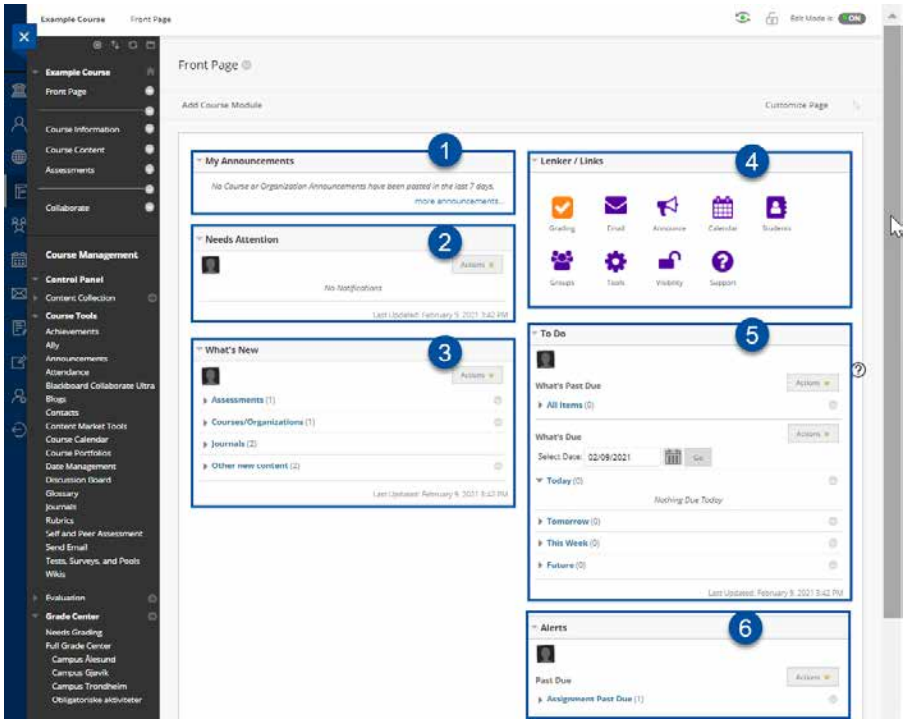

Рисунок 1.5 – Інтерфейс Blackboard Learn з аналітичними модулями

На рис. 1.6 наведено приклад ERP-рішення PeopleSoft Campus Solutions, яке використовується для комплексного управління студентськими даними, розкладом, оплатою навчання та академічними статусами. Система дозволяє адмініструвати процедури реєстрації, академічного контролю та обліку заборгованостей. Проте її впровадження потребує значних фінансових і технічних ресурсів, а адаптація під специфіку структурного підрозділу університету є складною.

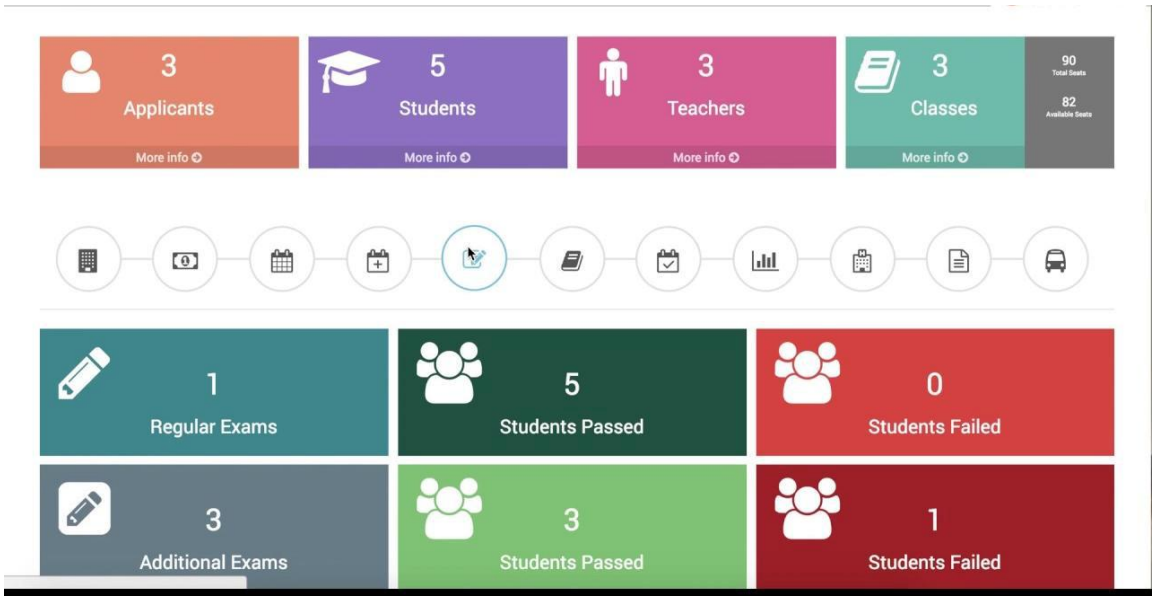

Рисунок 1.6 – Модуль управління академічними записами в PeopleSoft Campus Solutions

На рис. 1.7 представлено Odoo Education - модуль ERP-системи Odoo, який підтримує управління студентами, курсами, оплатою та розкладом занять. Система має модульну структуру та може бути адаптована для організації додаткових освітніх послуг. Однак для реалізації повноцінного процесу повторного вивчення дисципліни необхідне суттєве доопрацювання бізнес-логіки та інтеграція з внутрішніми регламентами університету.

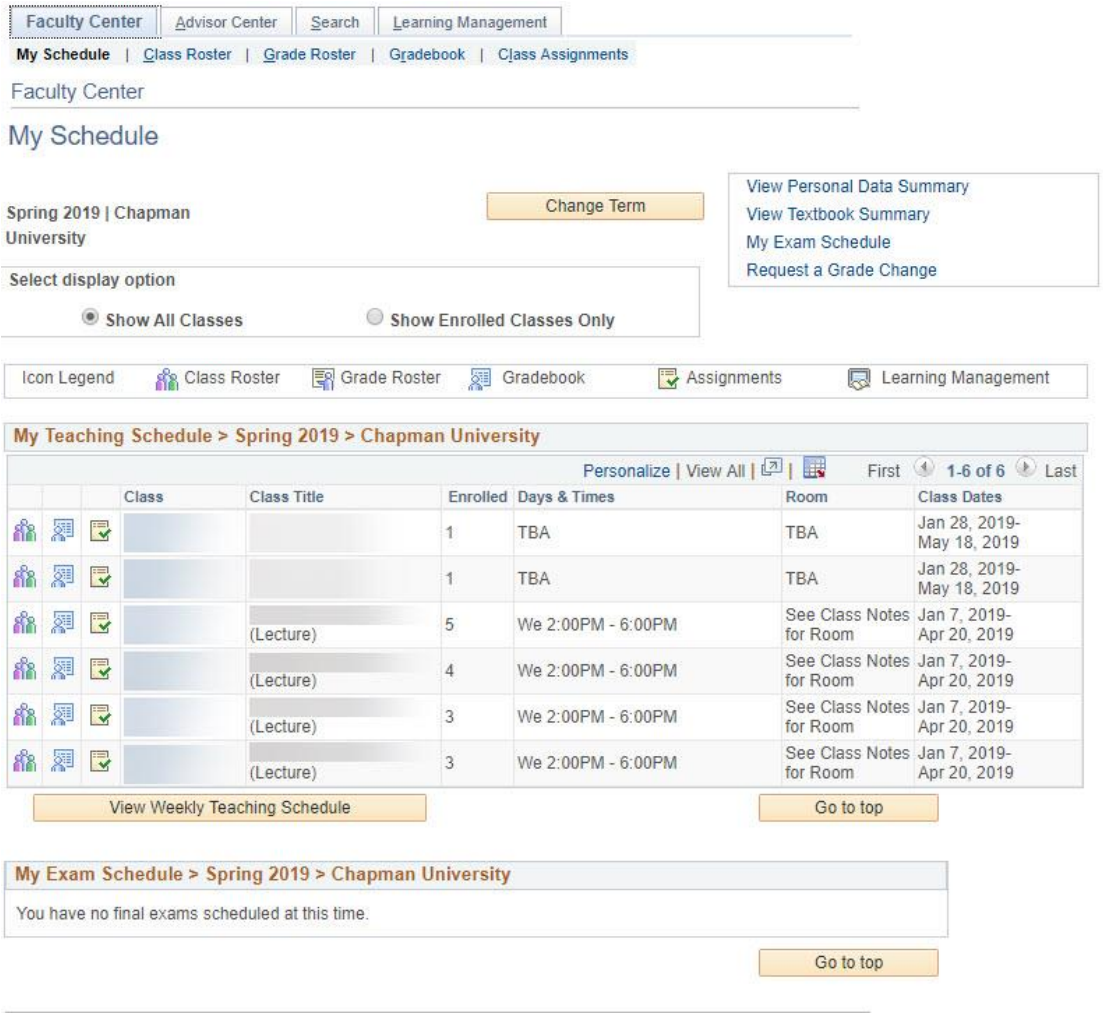

Рисунок 1.7 – Інтерфейс модуля управління студентами в Odoo Education

Порівняльний аналіз існуючих рішень (табл. 1.2) свідчить, що більшість систем орієнтована на загальне управління навчальним процесом або електронне навчання, тоді як спеціалізованої автоматизації процедури повторного вивчення дисципліни вони не забезпечують.

Таблиця 1.2 – Порівняння існуючих систем та розроблюваної інформаційної системи

Система	Тип архітектури	Підтримка повторного вивчення	Інтеграція з фінансовими модулями	Орієнтація
Moodle	Веб / клієнт–сервер	Часткова (через курси)	Обмежена (через плагіни)	Організація дистанційного навчання
Canvas LMS	Хмарна / REST	Часткова	Обмежена	Управління онлайн-курсами
Blackboard Learn	Централізована	Часткова	Так	Великі університети

Продовження таблиці 1.2

PeopleSoft Campus Solutions	ERP	Так (загально)	Так	Комплексне адміністрування ЗВО
Odoo Education	ERP / модульна	Часткова	Так	Освітні та корпоративні структури
Розроблювана система	Веб / клієнт–сервер	Повна (спеціалізована)	Так	Підтримка повторного вивчення дисциплін

Результати проведеного аналізу демонструють, що існуючі рішення забезпечують автоматизацію окремих аспектів освітнього процесу, проте не орієнтовані безпосередньо на регламентований процес надання освітньої послуги з повторного вивчення дисципліни. Запропонована інформаційна система розробляється з урахуванням специфіки діяльності структурного підрозділу університету, включає механізми подання та погодження заяв, обліку фінансових операцій, координації взаємодії між кафедрою, деканатом і бухгалтерією та забезпечує повний контроль життєвого циклу відповідної освітньої послуги.

1.3 Моделювання програмної системи

Моделювання предметної області інформаційної системи підтримки надання освітніх послуг з повторного вивчення дисципліни студентами структурного підрозділу університету передбачає формалізацію основних суб’єктів, функціональних процесів та інформаційних взаємозв’язків, що забезпечують повний життєвий цикл відповідної освітньої послуги. Основною метою моделювання є побудова структурованого опису логіки функціонування системи - від моменту подання студентом заяви до завершення повторного навчання та фіксації результатів оцінювання.

Для формалізації предметної області використано UML-моделювання, зокрема діаграму прецедентів, діаграму послідовності та діаграму активності. Застосування UML забезпечує уніфіковане представлення функціональних

можливостей системи, визначення ролей учасників процесу та відображення динаміки взаємодії між компонентами програмного забезпечення.

На рис. 1.8 наведено діаграму прецедентів інформаційної системи підтримки повторного вивчення дисципліни.

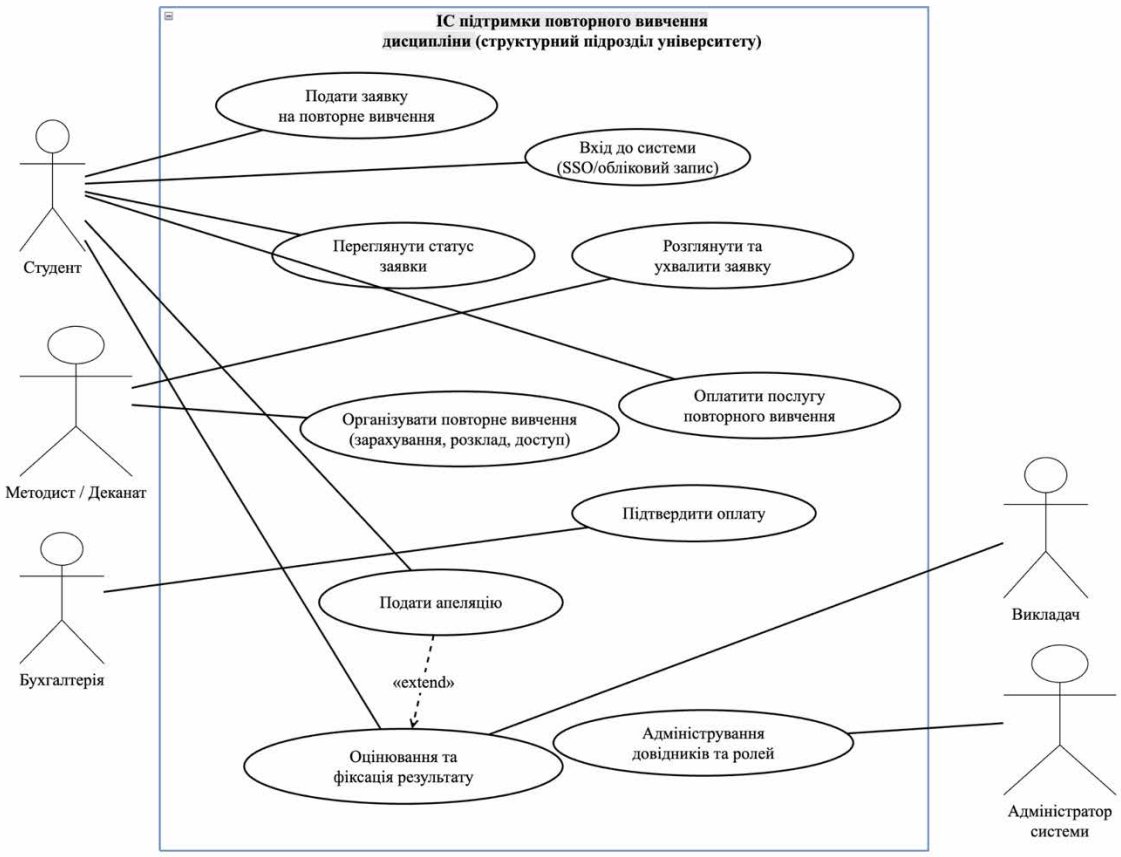

Рисунок 1.8 – Діаграма прецедентів інформаційної системи підтримки повторного вивчення дисципліни

Основними акторами виступають: Студент, Методист/Деканат, Бухгалтерія, Викладач та Адміністратор системи. Діаграма відображає ключові варіанти використання: «Подати заяву на повторне вивчення», «Переглянути статус заявки», «Розглянути та ухвалити заявку», «Організувати повторне вивчення», «Оплатити послугу», «Підтвердити оплату», «Оцінювання та фіксація результату», а також «Адміністрування довідників та ролей».

Зв’язок типу «extend» застосовано для сценарію «Подати апеляцію», який розширює прецедент «Оцінювання та фіксація результату». Така модель дозволяє відобразити можливість оскарження результатів повторного контролю та забезпечує гнучкість процесу.

На рис. 1.9 представлено діаграму послідовності, що деталізує сценарій оброблення заяви на повторне вивчення дисципліни.

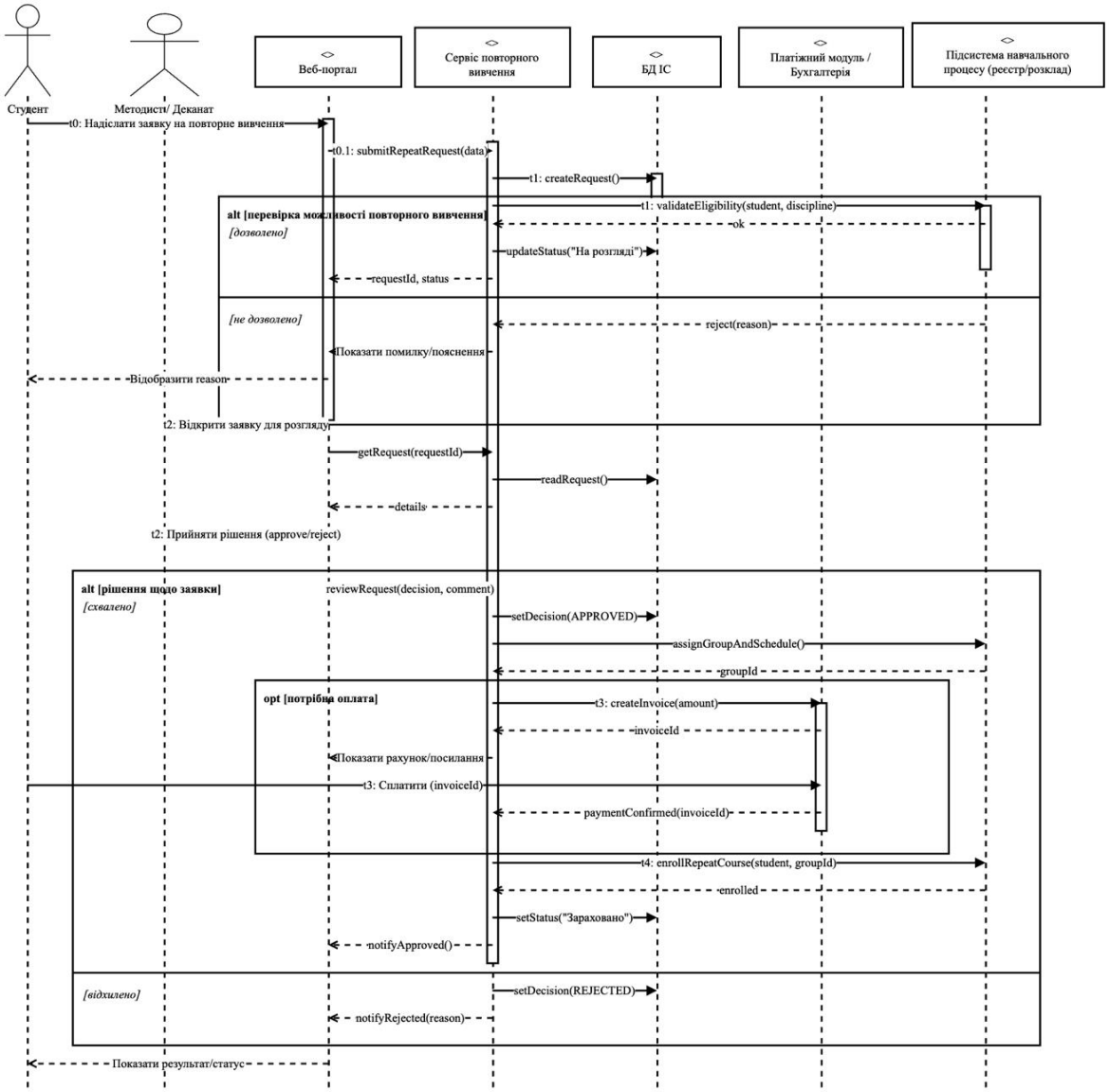

Рисунок 1.9 – Діаграма послідовності процесу оброблення заяви на повторне вивчення дисципліни

Процес ініціюється студентом через портал. Система створює запис звернення у базі даних та здійснює перевірку відповідності встановленим критеріям (академічний статус, кількість спроб, строки оформлення, наявність фінансових зобов’язань). У разі відповідності вимог заяві присвоюється статус «На розгляді» та вона передається методисту або деканату для прийняття рішення. Якщо заявку схвалено, система визначає необхідність оплати. За наявності фінансової складової формується рахунок через платіжний модуль,

після чого очікується підтвердження платежу. Після підтвердження оплати або за відсутності потреби в оплаті здійснюється зарахування студента на повторне вивчення та оновлення статусу заявки.

У випадку відмови система надсилає студенту повідомлення із зазначенням причини відхилення.

Узагальнений бізнес-процес надання освітньої послуги представлено на діаграмі активності (рис. 1.10).

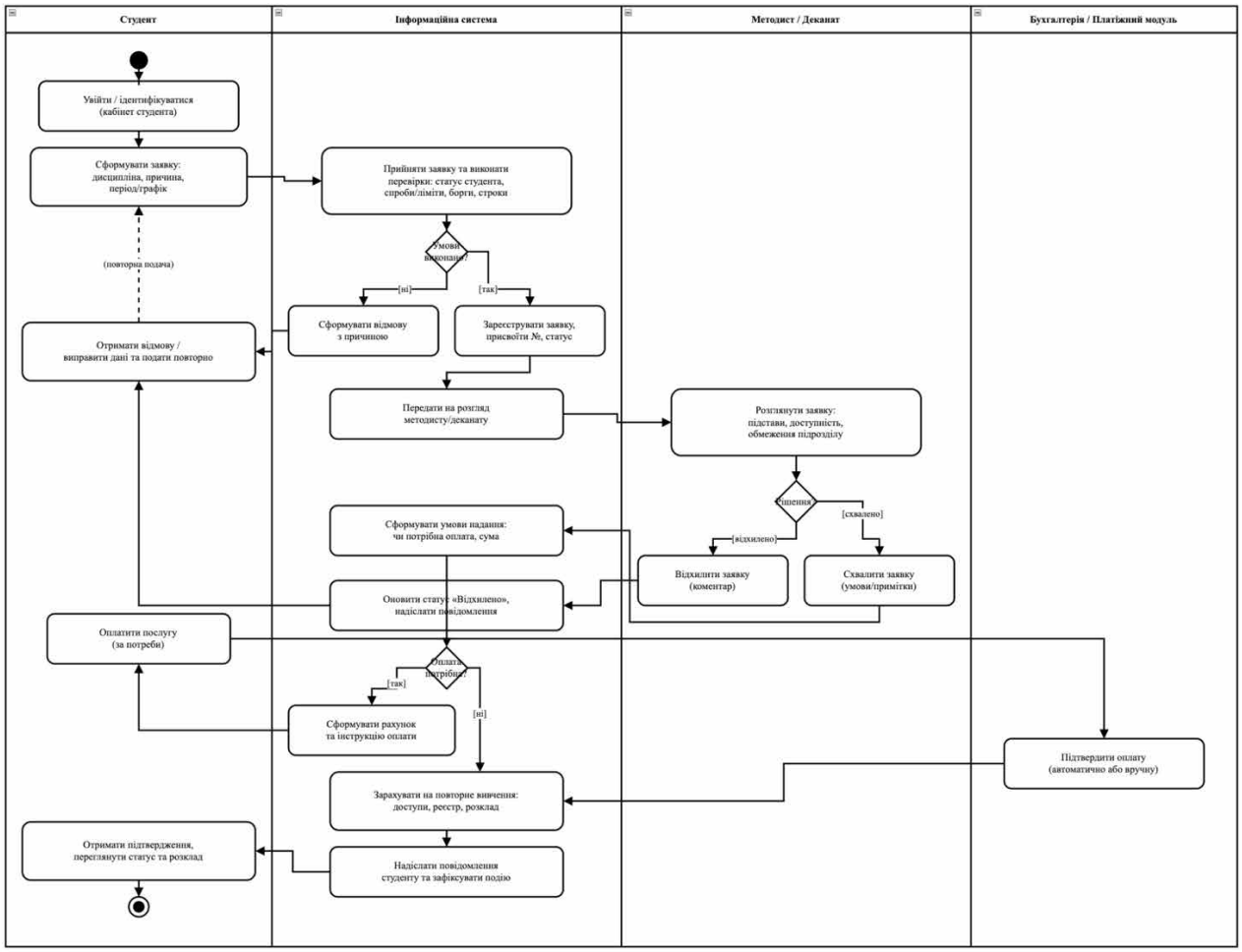

Рисунок 1.10 – Діаграма активності процесу надання освітньої послуги з повторного вивчення дисципліни

Діаграма побудована із використанням смуг відповідальності (swimlane): «Студент», «Інформаційна система», «Методист / Деканат», «Бухгалтерія / Платіжний модуль».

Процес розпочинається з автентифікації студента та формування заяви. Інформаційна система виконує перевірку умов допуску. У разі невідповідності

вимог формується відмова з поясненням та можливістю повторного подання заявки. Якщо умови виконані, заявка реєструється та передається на розгляд.

Після ухвалення позитивного висновку визначається необхідність оплати. У разі потреби формується документ, а після підтвердження платежу здійснюється зарахування студента на повторне проходження дисципліни, формуються доступи та оновлюється система.

Завершальним етапом є фіксація результату повторного оцінювання та інформування студента.

Узагальнення результатів моделювання дозволяє визначити такі ключові особливості системи:

– підтримка рольової взаємодії між студентом, адміністративним персоналом та викладачем;

– централізований контроль статусів заяв та регламентованість прийняття рішень;

– інтеграція з фінансовими та навчальними підсистемами університету;

– забезпечення повного життєвого циклу освітньої послуги з повторного вивчення дисципліни.

Побудована модель предметної області формує основу для подальшого проєктування архітектури програмного забезпечення та реалізації інформаційної системи, орієнтованої на автоматизацію процедур повторного навчання в межах структурного підрозділу університету.

1.4 Аналіз вимог інформаційної системи

Аналіз вимог інформаційної системи підтримки надання освітніх послуг з повторного вивчення дисципліни є ключовим етапом проєктування, що визначає функціональні можливості системи, її експлуатаційні характеристики та вимоги до безпеки оброблення персональних даних. Формування вимог здійснювалося з урахуванням результатів моделювання предметної області, нормативних

документів закладу вищої освіти та регламентів організації повторного вивчення дисциплін.

Проєктована система розглядається як веб-орієнтований клієнт–серверний програмний продукт, призначений для автоматизації процесів подання, розгляду, погодження та супроводу заяв на повторне вивчення дисциплін. Система забезпечує інтеграцію з підсистемами навчального процесу та, за необхідності, з фінансовим модулем університету.

Функціональні вимоги визначають перелік операцій, які система повинна виконувати для реалізації повного життєвого циклу освітньої послуги з повторного вивчення дисципліни. Основними користувачами системи є студент, методист/деканат, бухгалтерія, викладач та адміністратор системи.

Узагальнений перелік функціональних вимог наведено в таблиці 1.3.

Таблиця 1.3 – Функціональні вимоги інформаційної системи підтримки повторного вивчення дисципліни

№	Вимога	Опис	Роль користувача
1	Автентифікація користувача	Вхід до системи з використанням облікового запису	Усі користувачі
2	Подання заявки	Формування та надсилання заявки на повторне вивчення (дисципліна, причина, період)	Студент
3	Перевірка умов допуску	Автоматична перевірка академічного статусу, строків, наявності заборгованостей	Система
4	Реєстрація заявки	Присвоєння ідентифікатора та статусу «На розгляді»	Система
5	Розгляд заявки	Прийняття рішення щодо схвалення або відхилення	Методист / Деканат
6	Формування умов надання послуги	Визначення необхідності оплати та організаційних параметрів	Методист / Деканат
7	Формування рахунку	Генерація рахунку на оплату (за потреби)	Система
8	Підтвердження оплати	Підтвердження факту оплати	Бухгалтерія

9	Організація повторного навчання	Зарахування студента, формування розкладу, надання доступів	Система
10	Фіксація результатів	Внесення результатів повторного контролю	Викладач

Продовження таблиці 1.3

11	Подання апеляції	Оскарження результатів повторного оцінювання	Студент
12	Адміністрування довідників	Керування ролями, параметрами та нормативними довідниками	Адміністратор

Нефункціональні вимоги визначають якісні характеристики системи, що впливають на стабільність її роботи, зручність використання та масштабованість. З огляду на роль системи в адміністративному управлінні освітнім процесом особлива увага приділяється продуктивності, надійності та сумісності з існуючими інформаційними ресурсами університету.

Узагальнення нефункціональних вимог наведено в таблиці 1.4.

Таблиця 1.4 – Нефункціональні вимоги інформаційної системи підтримки повторного вивчення дисципліни

№	Категорія	Вимога	Показник
1	Продуктивність	Час оброблення заявки	≤ 2 с
2	Надійність	Збереження даних при збоях	≥ 99,9 %
3	Масштабованість	Підтримка одночасної роботи користувачів	≥ 100 користувачів
4	Сумісність	Інтеграція з підсистемою навчального процесу	Передбачено
5	Зручність	Інтуїтивний веб-інтерфейс	Висока
6	Розширюваність	Можливість додавання нових регламентів та параметрів	Передбачено

Окрему групу формують вимоги до безпеки, оскільки система обробляє персональні відомості студентів, інформацію про академічну заборгованість та фінансові процедури. Система повинна забезпечувати розмежування доступу, захист конфіденційної інформації та моніторинг цілісності даних.

Основні вимоги щодо безпеки наведено в таблиці 1.5.

Таблиця 1.5 – Вимоги до безпеки інформаційної платформи підтримки повторного вивчення дисципліни

№	Вимога	Опис
1	Контроль доступу	Обов’язкова автентифікація користувачів

Продовження таблиці 1.5

2	Рольова модель	Розмежування прав доступу відповідно до ролі
3	Захист персональних даних	Обмеження доступу до персональної інформації
4	Реєстрація подій	Фіксація дій користувачів і змін статусів заяв
5	Цілісність даних	Контроль коректності записів у базі даних
6	Захист фінансових операцій	Контроль підтвердження оплат і обмеження несанкціонованих змін

Проведений аналіз вимог дозволяє сформувати специфікацію інформаційної системи, орієнтованої на автоматизацію та регламентацію процесу повторного вивчення дисциплін у межах структурного підрозділу університету. Запропонована система забезпечує централізований контроль заяв, прозорість прийняття управлінських рішень та інтеграцію з фінансовими й навчальними підсистемами. Такий підхід створює передумови для підвищення ефективності організації освітніх послуг та зменшення адміністративного навантаження.

1.5 Постановка завдання

На основі проведеного аналізу предметної області, сформульованих функціональних і нефункціональних вимог, а також результатів UML-моделювання було сформовано постановку завдання розроблення інформаційної системи підтримки надання освітніх послуг з повторного вивчення дисципліни студентами структурного підрозділу університету.

Завдання полягає у проєктуванні та розробленні інформаційної системи, призначеної для автоматизації процесів подання, розгляду, погодження та супроводу заяв на повторне вивчення дисциплін, забезпечення контролю

виконання регламентів та інтеграції з підсистемами навчального процесу і фінансового обліку.

Проєктована платформа розглядається як програмний комплекс із клієнт–серверною архітектурою, що функціонує у межах інформаційної інфраструктури університету. Система повинна забезпечувати повний життєвий цикл освітньої послуги з повторного вивчення дисципліни — від ініціювання студентом звернення до завершення повторного контролю та фіксації результатів у відповідних реєстрах.Для досягнення поставленої мети необхідно спроєктувати архітектуру системи, що складається з взаємопов’язаних функціональних підсистем:

– підсистема користувацької взаємодії, яка забезпечує роботу студентів, методистів/деканату, бухгалтерії, викладачів і адміністратора через веб-інтерфейс;

– підсистема управління заявами, що відповідає за реєстрацію, перевірку, зміну статусів і зберігання заяв на повторне вивчення дисциплін;

– підсистема прийняття управлінських рішень, яка реалізує механізм розгляду та погодження заяв відповідно до встановлених регламентів;

– платіжна підсистема, що забезпечує формування рахунків та підтвердження оплати освітніх послуг (за наявності фінансової складової);

– підсистема інтеграції з навчальним процесом, яка забезпечує зарахування студента до групи повторного вивчення, формування доступів та фіксацію результатів повторного контролю.

Вхідними даними системи є:

– персональні та облікові дані користувачів (ідентифікатор, роль, параметри доступу);

– інформація про дисципліну (назва, семестр, форма контролю);

– дані про академічну заборгованість студента;

– параметри регламенту повторного вивчення (строки подання, кількість спроб, необхідність оплати);

– інформація про фінансові операції (рахунок, статус оплати).

Дані можуть надходити як безпосередньо від користувача через веб-інтерфейс, так і шляхом інтеграції з внутрішніми інформаційними системами університету (реєстр студентів, навчальні плани, підсистема фінансового обліку).

Вихідними даними системи є:

– статуси заяв на повторне вивчення (зареєстровано, на розгляді, схвалено, відхилено, зараховано);

– сформовані управлінські рішення та коментарі;

– інформація про необхідність та підтвердження оплати

– дані про зарахування студента до групи повторного вивчення;

– результати повторного контролю та їх фіксація у відповідних реєстрах;

– аналітичні звіти щодо кількості поданих заяв, прийнятих рішень та навантаження підрозділів.

Основною метою постановки завдання є створення інформаційної системи, яка забезпечує регламентовану, прозору та контрольовану процедуру надання освітньої послуги з повторного вивчення дисципліни. Автоматизація відповідних процесів дозволяє мінімізувати ризик помилок, скоротити час оброблення заяв, зменшити адміністративне навантаження на працівників структурного підрозділу та підвищити якість управління освітнім процесом.

Результатом виконання поставленого завдання має стати програмний продукт, який забезпечує централізований облік заяв, підтримку прийняття управлінських рішень, інтеграцію з фінансовими та навчальними підсистемами університету та формування єдиного інформаційного середовища для супроводу процесу повторного вивчення дисциплін.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних у вигляді ER-діаграми

Логічна модель даних інформаційної системи підтримки повторного вивчення дисципліни призначена для формалізації структури зберігання даних, визначення взаємозв’язків між основними сутностями системи та забезпечення цілісності інформаційних процесів у межах повторного вивчення навчальних компонентів. Модель побудована відповідно до функціональних вимог системи та орієнтована на підтримку процесів подання заявок, перевірки умов допуску, організації повторного навчання, обліку оплат, фіксації результатів оцінювання та журналювання дій користувачів.

На рисунку 2.1 наведено логічну модель даних системи у вигляді ER-діаграми.

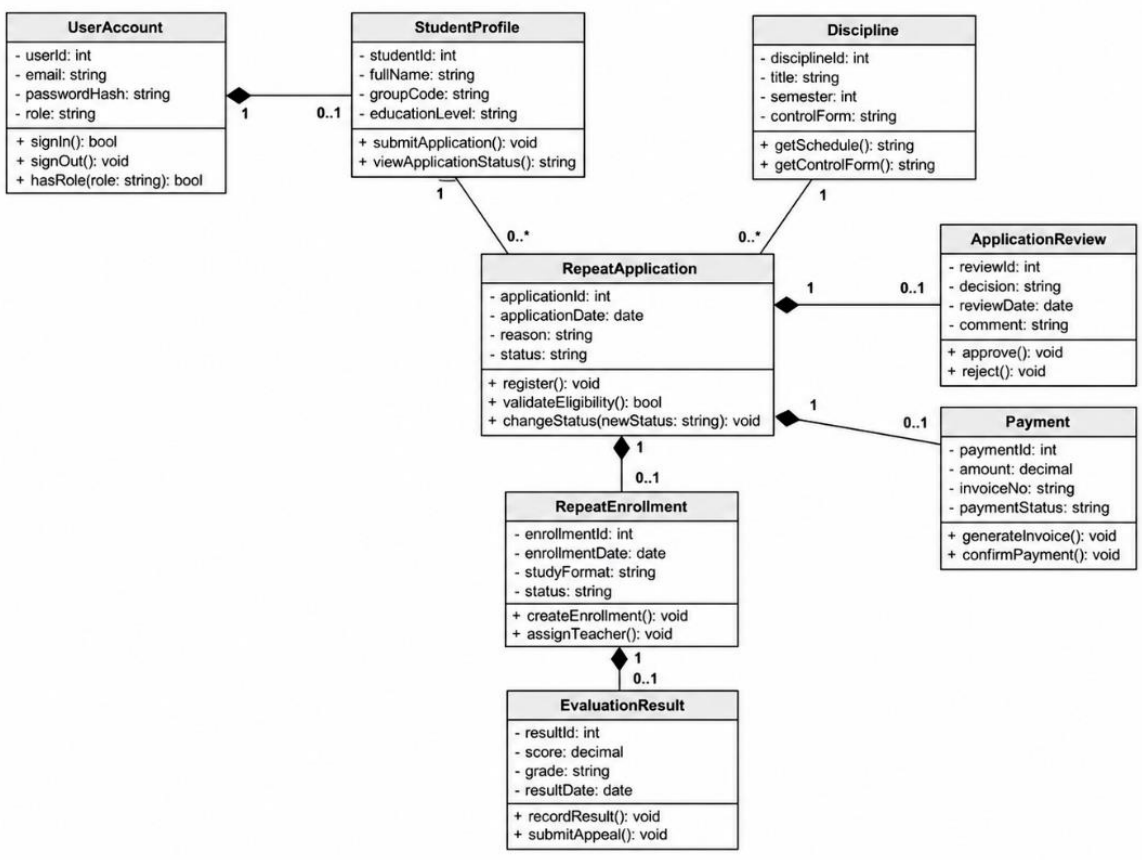

Рисунок 2.1 – Логічна модель даних інформаційної системи підтримки повторного вивчення дисципліни

Основу моделі становить сутність REPEAT_APPLICATION, яка забезпечує централізований облік заяв студентів на повторне вивчення дисциплін. Вона пов’язана із сутностями STUDENT_PROFILE, DISCIPLINE, PAYMENT, REPEAT_ENROLLMENT та EVALUATION_RESULT, що забезпечує підтримку повного життєвого циклу освітньої послуги. Для реалізації механізмів автентифікації та розмежування доступу використовується сутність USER_ACCOUNT, а для контролю змін і реєстрації подій - AUDIT_LOG.

Під час проєктування логічної моделі враховано необхідність підтримки зв’язків типу «один до багатьох» між користувачами, студентами, дисциплінами та заявками, а також зв’язків «один до одного» для сутностей, що відображають результати проходження повторного вивчення та фінансові операції. Такий підхід забезпечує мінімізацію дублювання даних, спрощує подальшу фізичну реалізацію бази даних і підвищує узгодженість інформації.

Основні сутності логічної моделі даних наведено у таблиці 2.1.

Таблиця 2.1 – Основні сутності логічної моделі даних

Сутність	Призначення
USER_ACCOUNT	Зберігання облікових записів користувачів системи
STUDENT_PROFILE	Зберігання інформації про студентів
DISCIPLINE	Зберігання даних про навчальні дисципліни
REPEAT_APPLICATION	Облік заяв на повторне вивчення дисциплін
PAYMENT	Облік фінансових операцій та оплат
REPEAT_ENROLLMENT	Облік зарахування студентів на повторне вивчення
EVALUATION_RESULT	Зберігання результатів повторного контролю
AUDIT_LOG	Журналювання подій і дій користувачів

Сформована логічна модель даних забезпечує структуроване представлення предметної області, підтримує реалізацію основних бізнес-процесів системи та створює основу для подальшого проєктування фізичної моделі бази даних і програмної реалізації інформаційної системи.

2.2 Діаграма класів і кооперації

Для формалізації внутрішньої структури програмного забезпечення та моделювання взаємодії між основними об’єктами інформаційної системи підтримки повторного вивчення дисципліни було побудовано діаграму класів і діаграми кооперації. Використання UML-моделювання дозволило визначити склад програмних сутностей, їх атрибути, операції та взаємозв’язки, а також описати механізми обміну повідомленнями між об’єктами системи під час виконання ключових бізнес-процесів.

На рисунку 2.2 наведено діаграму класів інформаційної системи.

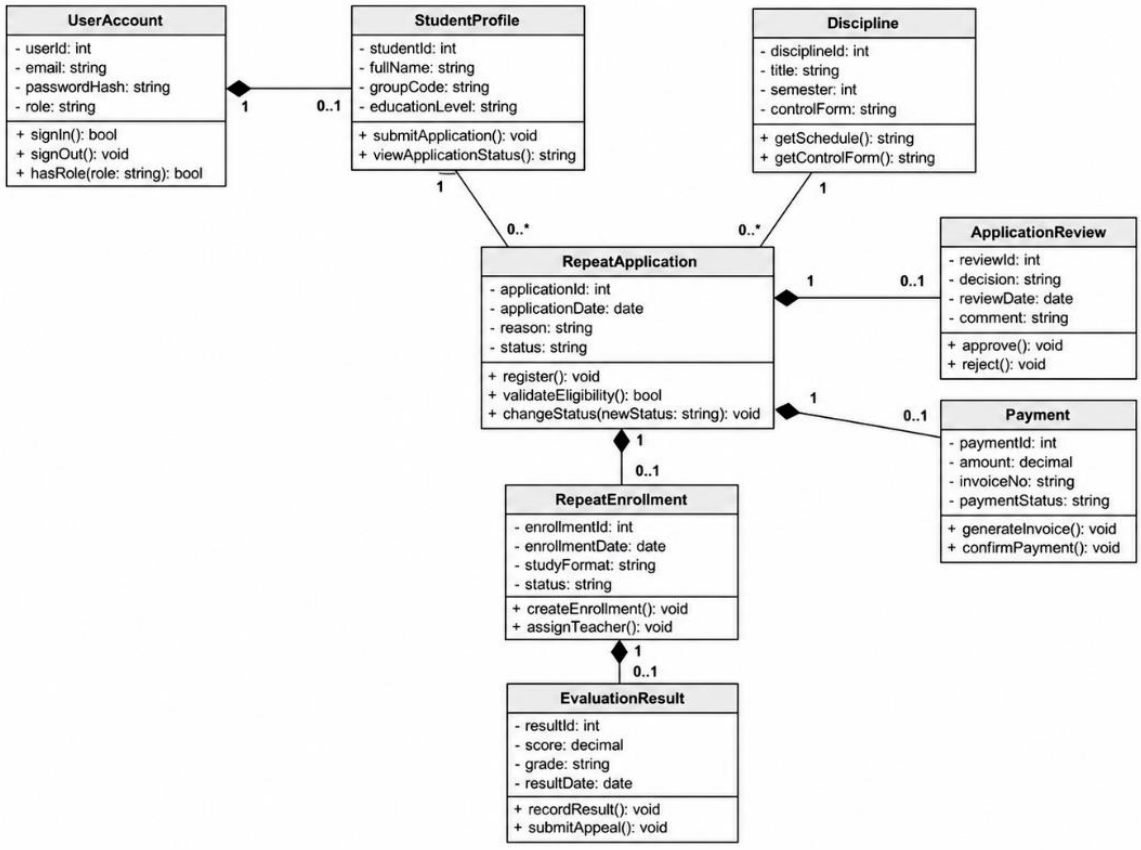

Рисунок 2.2 – Діаграма класів інформаційної системи підтримки повторного вивчення дисципліни

Діаграма класів відображає основні програмні сутності системи та їх логічну взаємодію. Центральним елементом моделі є клас RepeatApplication, який реалізує функції реєстрації заявки, перевірки умов допуску та керування

статусами проходження повторного вивчення дисципліни. Взаємодія із користувачами реалізується через класи UserAccount та StudentProfile, а організація повторного навчання та контролю знань забезпечується класами RepeatEnrollment, ApplicationReview, Payment і EvaluationResult.

Побудована модель забезпечує підтримку механізмів рольового доступу, обліку оплат, управління процесом зарахування студентів та фіксації результатів повторного контролю. Використання асоціацій і композицій між класами дозволяє підтримувати цілісність об’єктної структури та узгодженість даних у програмній системі.

Основні класи програмної системи наведено у таблиці 2.2.

Таблиця 2.2 – Основні класи інформаційної системи

Клас	Призначення
UserAccount	Автентифікація користувачів і контроль ролей
StudentProfile	Робота з профілем студента
Discipline	Зберігання параметрів навчальної дисципліни
RepeatApplication	Керування заявками на повторне вивчення
ApplicationReview	Розгляд та погодження заяв
Payment	Формування та підтвердження оплат
RepeatEnrollment	Організація повторного вивчення дисципліни
EvaluationResult	Фіксація результатів оцінювання

Для моделювання динамічної взаємодії між об’єктами системи побудовано діаграми кооперації, що описують послідовність обміну повідомленнями між екземплярами класів у межах основних сценаріїв роботи системи.

На рисунку 2.3 наведено кооперацію процесу розгляду заявки та підтвердження оплати.

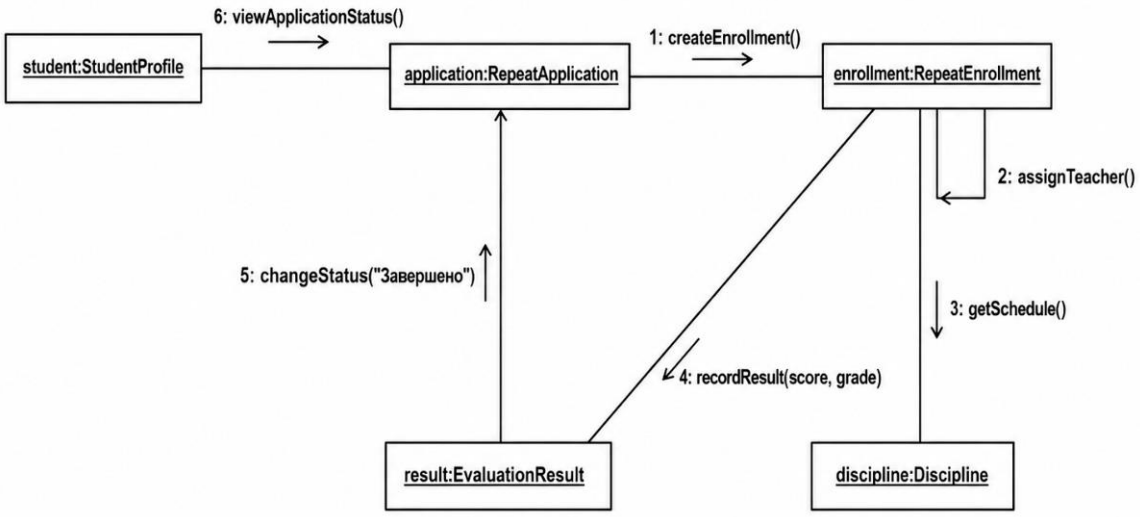

Рисунок 2.3 – Кооперація процесу розгляду заявки та підтвердження оплати

У межах цієї кооперації реалізовано механізм перевірки умов допуску, погодження заявки, формування рахунку та підтвердження оплати з подальшим оновленням статусу заявки. Взаємодія між класами RepeatApplication, ApplicationReview та Payment забезпечує централізоване управління процесом повторного вивчення дисципліни та контроль фінансових операцій.

На рисунку 2.4 наведено кооперацію процесу зарахування студента на повторне вивчення дисципліни та фіксації результату.

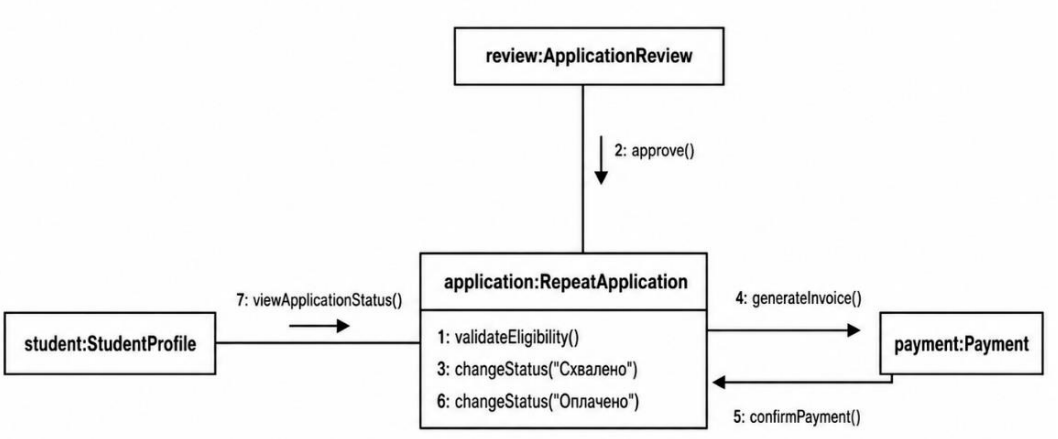

Рисунок 2.4 – Кооперація процесу зарахування та фіксації результату повторного вивчення

У межах даного сценарію здійснюється створення запису про зарахування студента, призначення викладача, отримання параметрів дисципліни та реєстрація результатів повторного контролю. Після завершення оцінювання

система автоматично оновлює статус заявки та забезпечує доступ до результатів повторного навчання.

На рисунку 2.5 наведено кооперацію процесу формування результату повторного контролю.

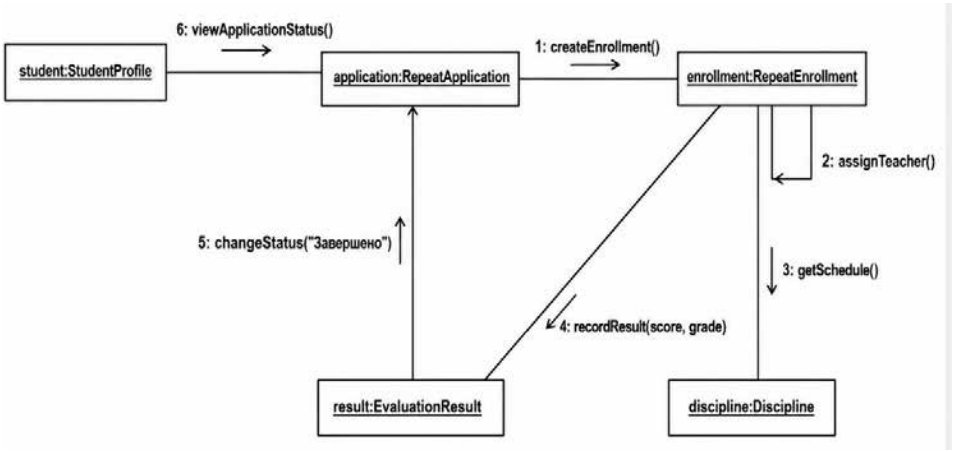

Рисунок 2.5 – Кооперація процесу формування результатів повторного контролю

У процесі кооперації реалізується взаємодія між об’єктами RepeatEnrollment, Discipline, EvaluationResult та RepeatApplication, що забезпечує передачу даних про результати оцінювання, оновлення статусів проходження повторного навчання та відображення підсумкової інформації для студента.

Сформовані UML-моделі забезпечують структуроване представлення архітектури програмної системи, формалізують взаємодію між основними програмними компонентами та створюють основу для подальшої реалізації програмного забезпечення інформаційної системи підтримки повторного вивчення дисципліни.

2.3 Діаграма компонентів розроблювальної системи

Для формалізації архітектурної структури програмного забезпечення інформаційної системи підтримки повторного вивчення дисципліни було побудовано діаграму компонентів, яка визначає склад основних програмних модулів, їх функціональне призначення та взаємозв’язки між окремими частинами системи. Компонентний підхід дозволяє реалізувати модульну

структуру програмного забезпечення, забезпечити розподіл функціональних обов’язків між підсистемами та спростити подальший супровід і масштабування програмного продукту.

На рисунку 2.6 наведено діаграму компонентів інформаційної системи підтримки повторного вивчення дисципліни.

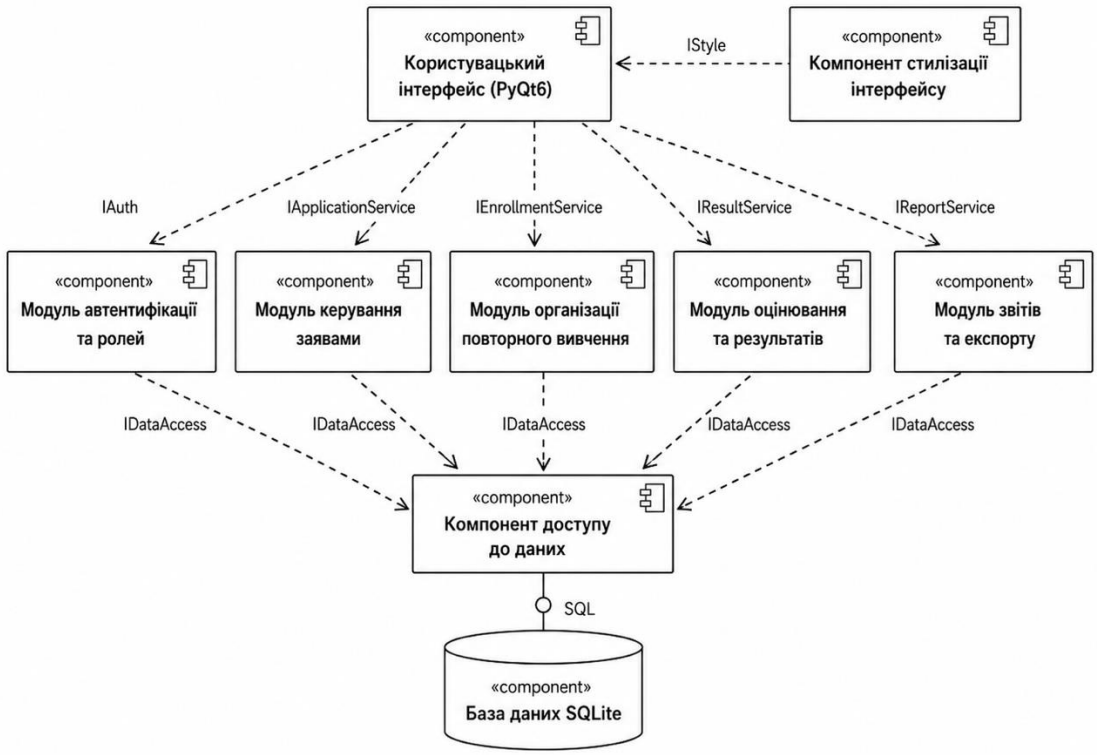

Рисунок 2.6 – Діаграма компонентів інформаційної системи підтримки повторного вивчення дисципліни

Архітектура системи побудована за принципом розподілу функціональності між окремими компонентами прикладного рівня та компонентом доступу до даних. Центральним елементом взаємодії із користувачем є компонент користувацького інтерфейсу, реалізований засобами PyQt6. Через інтерфейс користувачі здійснюють автентифікацію, подання заяв, перегляд статусів, роботу з результатами повторного контролю та формування звітної інформації.

Функціональна логіка системи розподілена між окремими модулями: модулем автентифікації та ролей, модулем керування заявами, модулем організації повторного вивчення, модулем оцінювання та результатів, а також

модулем звітів та експорту. Обмін даними між прикладними компонентами та базою даних реалізується через спеціалізований компонент доступу до даних, що забезпечує централізовану роботу із SQL-запитами та підтримку цілісності інформації.

Для забезпечення уніфікованого візуального оформлення програмного забезпечення використовується окремий компонент стилізації інтерфейсу, який взаємодіє з користувацьким інтерфейсом через інтерфейс IStyle. Взаємодія прикладних модулів із компонентом доступу до даних реалізується через інтерфейс IDataAccess, що дозволяє забезпечити незалежність бізнес-логіки від механізмів фізичного зберігання даних.

Основні компоненти системи наведено у таблиці 2.3.

Таблиця 2.3 – Основні компоненти програмної системи

Компонент	Призначення
Користувацький інтерфейс (PyQt6)	Взаємодія користувача із системою
Компонент стилізації інтерфейсу	Формування єдиного стилю програмного інтерфейсу
Модуль автентифікації та ролей	Авторизація користувачів і контроль доступу
Модуль керування заявами	Робота із заявками на повторне вивчення
Модуль організації повторного вивчення	Керування процесом зарахування та навчання
Модуль оцінювання та результатів	Формування та облік результатів повторного контролю
Модуль звітів та експорту	Формування звітної документації та експорту даних
Компонент доступу до даних	Взаємодія із базою даних
База даних SQLite	Зберігання інформації системи

Сформована компонентна структура забезпечує модульність програмного забезпечення, розмежування функціональних обов’язків між підсистемами та підтримку централізованої взаємодії з базою даних. Використання компонентного підходу створює основу для подальшого розширення функціональності системи, спрощує інтеграцію нових модулів і підвищує ефективність супроводу програмного забезпечення.

2.4 Діаграма пакетів

Для структуризації програмної архітектури інформаційної системи підтримки повторного вивчення дисципліни було побудовано діаграму пакетів. Вона відображає логічне групування програмних модулів за функціональним призначенням і показує залежності між прикладною та інфраструктурною частинами системи. Такий підхід спрощує супровід програмного забезпечення, зменшує зв’язаність між окремими частинами програми та забезпечує зрозумілу організацію проєкту.

На рисунку 2.7 наведено діаграму пакетів інформаційної системи.

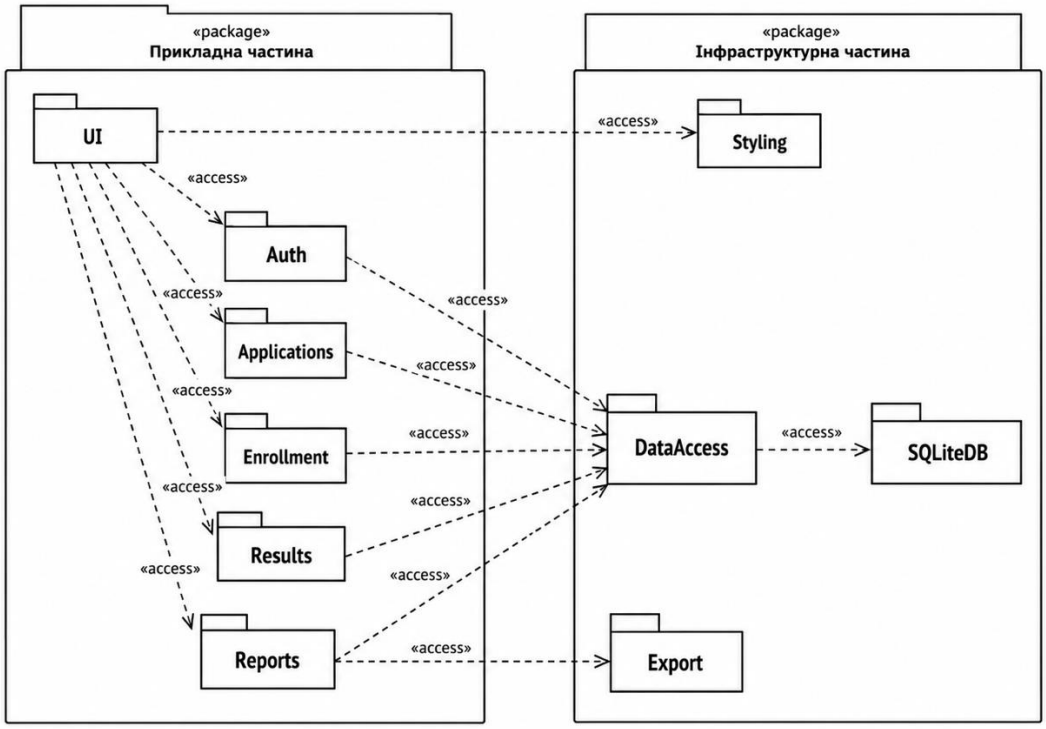

Рисунок 2.7 – Діаграма пакетів інформаційної системи підтримки повторного вивчення дисципліни

Пакетна структура системи поділена на дві основні частини: прикладну та інфраструктурну. До прикладної частини належать пакети UI, Auth, Applications, Enrollment, Results і Reports, які забезпечують взаємодію з користувачем, автентифікацію, роботу із заявами, організацію повторного вивчення, фіксацію

результатів та формування звітів. Інфраструктурна частина містить пакети DataAccess, SQLiteDB, Export і Styling, що відповідають за доступ до даних, зберігання інформації, експорт результатів і стилізацію інтерфейсу.

Основні пакети програмної архітектури наведено у таблиці 2.4.

Таблиця 2.4 – Основні пакети програмної архітектури системи

Пакет	Частина системи	Призначення
UI	Прикладна	Взаємодія користувача з функціями системи
Auth	Прикладна	Автентифікація та перевірка ролей
Applications	Прикладна	Оброблення заяв на повторне вивчення
Enrollment	Прикладна	Організація зарахування на повторне навчання
Results	Прикладна	Облік результатів повторного контролю
Reports	Прикладна	Формування звітів і передача даних на експорт
DataAccess	Інфраструктурна	Централізований доступ до бази даних
SQLiteDB	Інфраструктурна	Зберігання даних інформаційної системи
Export	Інфраструктурна	Збереження експортованих файлів і звітів
Styling	Інфраструктурна	Оформлення графічного інтерфейсу

Залежності між пакетами побудовані за принципом одностороннього доступу: прикладні пакети використовують інфраструктурні сервіси, але не містять механізмів фізичного зберігання даних. Пакет UI звертається до функціональних пакетів системи та пакета Styling, а пакети Auth, Applications, Enrollment, Results і Reportsпрацюють із даними через DataAccess. Окремо пакет Reports взаємодіє з Export, що забезпечує формування вихідних файлів і звітної інформації.

Сформована діаграма пакетів показує логічну організацію програмної системи та підтверджує її модульність. Запропонована структура забезпечує чітке розмежування прикладної логіки, доступу до даних і допоміжних інфраструктурних засобів, що є важливим для подальшої реалізації, тестування та супроводу інформаційної системи.

2.5 Висновки до другого розділу

У другому розділі виконано проєктування інформаційного та програмного забезпечення інформаційної системи підтримки повторного вивчення дисципліни студентами структурного підрозділу університету. У процесі проєктування сформовано логічну модель даних, UML-моделі класів, кооперацій, компонентів і пакетної структури системи, що дозволило формалізувати архітектуру програмного забезпечення та визначити основні механізми взаємодії між підсистемами.

У результаті побудови ER-діаграми визначено основні сутності предметної області та зв’язки між ними. Логічна модель даних забезпечує підтримку процесів подання заяв, організації повторного навчання, обліку оплат, фіксації результатів повторного контролю та журналювання дій користувачів. Запропонована структура даних мінімізує дублювання інформації та створює основу для подальшої фізичної реалізації бази даних.

Побудована діаграма класів дозволила визначити об’єктну структуру програмної системи, основні атрибути та операції класів, а також взаємозв’язки між програмними сутностями. Діаграми кооперації формалізували сценарії взаємодії об’єктів під час оброблення заявок, підтвердження оплат, організації повторного вивчення та фіксації результатів оцінювання.

У межах компонентного проєктування сформовано модульну структуру програмного забезпечення, яка включає користувацький інтерфейс, модулі автентифікації, керування заявами, організації повторного вивчення, оцінювання результатів, формування звітів та компонент доступу до даних. Це забезпечує розподіл функціональності між окремими частинами системи та спрощує подальший супровід програмного продукту.

Розроблена діаграма пакетів дозволила структурувати програмну систему на прикладну та інфраструктурну частини, що забезпечує логічне розмежування бізнес-логіки, механізмів доступу до даних і допоміжних сервісів. Такий підхід підвищує масштабованість та підтримуваність програмного забезпечення.

Отримані результати другого розділу створюють основу для подальшої реалізації програмного забезпечення, розроблення фізичної бази даних, реалізації функціональних модулів системи та проведення тестування інформаційної системи підтримки повторного вивчення дисципліни.

3 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ ТА РОЗРОБЛЕННЯ АЛГОРИТМІЧНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Обґрунтування вибору технологічних засобів та інструментів реалізації

Розроблення інформаційної системи підтримки повторного вивчення дисципліни студентами структурного підрозділу університету потребує вибору технологічних засобів, які забезпечують підтримку процесів автентифікації користувачів, подання та оброблення заяв, організації повторного навчання, фіксації результатів повторного контролю, формування звітів і локального зберігання даних. Оскільки проєктована система орієнтована на роботу у форматі автономного desktop-застосунку без окремого серверного розгортання, основними критеріями вибору технологій є простота впровадження, стабільність роботи, підтримка графічного інтерфейсу, можливість локального зберігання даних і подальшого розширення функціональності.

Для реалізації системи було обрано стек Python + PyQt6 + SQLite. Використання мови Python дозволяє швидко реалізувати прикладну логіку, механізми оброблення заяв, перевірки статусів, формування звітів і алгоритми взаємодії між модулями системи. Графічний інтерфейс реалізовано на основі PyQt6, оскільки Qt Widgets підтримують створення класичних desktop-інтерфейсів із багатовіконною структурою, таблицями, формами введення, панелями навігації та діалоговими вікнами. Зберігання даних реалізовано за допомогою SQLite через стандартний модуль sqlite3, що дозволяє працювати з реляційною базою даних без встановлення окремого серверу баз даних.

Основні технологічні засоби реалізації системи наведено у таблиці 3.1.

Таблиця 3.1 – Порівняльна характеристика технологічних засобів реалізації системи

Технологія / інструмент	Призначення	Переваги	Обґрунтування вибору
Python 3.11 / 3.12	Реалізація прикладної логіки	Простий синтаксис, кросплатформеність, підтримка бібліотек	Дає змогу реалізувати модулі керування заявами, контролю статусів, роботи з результатами та звітами
PyQt6	Побудова графічного інтерфейсу	Підтримка форм, таблиць, вкладок і діалогових вікон	Забезпечує створення повноцінного desktop-застосунку для роботи користувачів системи
SQLite	Локальна база даних	Не потребує окремого серверу, підтримує SQL-запити та транзакції	Доцільна для автономної інформаційної системи структурного підрозділу
Qt Style Sheets / QSS	Оформлення інтерфейсу	Підтримка стилізації елементів інтерфейсу	Використовується для формування єдиного стилю програмної системи
hashlib / PBKDF2-HMAC	Захист паролів користувачів	Підтримка криптографічного хешування	Забезпечує безпечне зберігання облікових даних
csv	Експорт звітної інформації	Простий формат табличного обміну даними	Дає змогу експортувати результати роботи системи
PyInstaller	Пакування програмного забезпечення	Формування виконуваного файлу	Використовується для створення готової версії програми
Git / GitHub	Контроль версій	Підтримка історії змін і командної розробки	Забезпечує керованість процесом розроблення

SQLite використовує реляційну модель даних із підтримкою SQL-запитів, індексів, зовнішніх ключів і транзакцій, що є достатнім для реалізації локальної інформаційної системи підтримки повторного вивчення дисципліни. У межах проєкту база даних використовується для зберігання інформації про користувачів, студентів, дисципліни, заявки, результати повторного контролю, оплати та журнал дій користувачів. Такий підхід забезпечує централізоване зберігання інформації та підтримку цілісності даних.

Для реалізації графічного інтерфейсу використовується PyQt6, оскільки бібліотека підтримує побудову desktop-застосунків із багатовіконною структурою та забезпечує інтеграцію з елементами управління Qt Widgets. Це дозволяє реалізувати форми подання заяв, таблиці перегляду статусів, панелі адміністрування, модулі роботи з результатами повторного контролю та вікна формування звітів у межах єдиного інтерфейсу користувача.

Для захисту облікових записів користувачів застосовується модуль hashlib із підтримкою механізму PBKDF2-HMAC. У системі це використовується для зберігання не відкритих паролів, а їхніх криптографічних хешів, що підвищує безпеку авторизації та зменшує ризик компрометації персональних даних користувачів.

Оформлення графічного інтерфейсу реалізовано за допомогою Qt Style Sheets, що дозволяє централізовано налаштовувати зовнішній вигляд кнопок, таблиць, полів введення та інших елементів інтерфейсу. Це забезпечує уніфікований стиль програмної системи та підвищує зручність взаємодії користувача із застосунком.

Для експорту результатів роботи системи використовується модуль csv, який дозволяє формувати табличні звіти у форматі CSV для подальшого відкриття у табличних редакторах або передачі адміністративним підрозділам університету. Для забезпечення стабільного запуску застосунку на різних комп’ютерах використовується ізольоване середовище venv, а для підготовки завершеної версії програмного забезпечення - PyInstaller, який дозволяє сформувати виконуваний файл із вбудованими залежностями.

Отже, обраний стек технологій є доцільним для реалізації інформаційної системи підтримки повторного вивчення дисципліни, оскільки забезпечує локальне зберігання даних, підтримку повноцінного графічного інтерфейсу, механізми безпечної автентифікації користувачів, роботу з реляційною базою даних та формування звітної інформації. Використання Python, PyQt6 та SQLite створює основу для подальшого розширення функціональності системи,

інтеграції нових модулів і підвищення ефективності автоматизації процесів повторного вивчення дисциплін у структурному підрозділі університету.

3.2 Представлення архітектури програмної системи

Архітектура інформаційної системи підтримки повторного вивчення дисципліни побудована за модульним принципом із використанням desktop-орієнтованого підходу. Основною метою архітектурного проєктування є забезпечення централізованого керування процесами подання та оброблення заяв, організації повторного навчання, контролю результатів повторного оцінювання та взаємодії користувачів із локальною базою даних у межах єдиного програмного середовища.

На рисунку 3.1 наведено діаграму компонентного розгортання інформаційної системи підтримки повторного вивчення дисципліни.

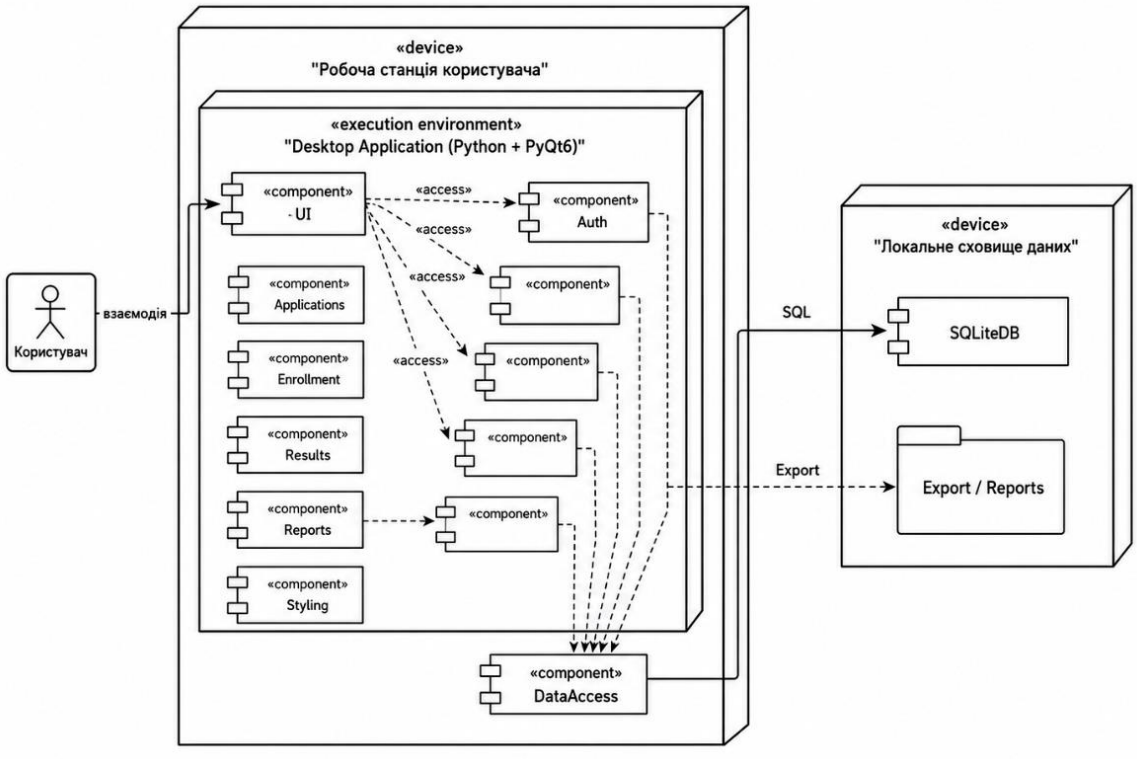

Рисунок 3.1 – Діаграма компонентного розгортання інформаційної системи підтримки повторного вивчення дисципліни

Архітектура системи реалізована у форматі desktop-застосунку, який функціонує на робочій станції користувача в середовищі Python та PyQt6.

Взаємодія користувача із системою здійснюється через графічний інтерфейс, який забезпечує доступ до функціональних модулів керування заявами, організації повторного вивчення, роботи з результатами оцінювання, формування звітів та авторизації користувачів.

Основні програмні компоненти функціонують у межах середовища виконання Desktop Application (Python + PyQt6) та взаємодіють між собою через внутрішні інтерфейси доступу. Для підтримки логіки оброблення даних використовується окремий компонент DataAccess, який реалізує централізовану взаємодію з локальною базою даних SQLite. Такий підхід забезпечує відокремлення бізнес-логіки від механізмів фізичного зберігання даних та підвищує підтримуваність програмного забезпечення.

Для зберігання структурованої інформації використовується локальне сховище даних SQLiteDB, яке містить дані про користувачів, студентів, дисципліни, заявки, результати повторного контролю, оплату та журнал дій системи. Взаємодія із базою даних здійснюється через SQL-запити, що формуються компонентом доступу до даних.

Окремо в архітектурі передбачено файлове сховище Export / Reports, яке використовується для формування та збереження звітної документації, експортованих результатів і службових файлів системи. Це дозволяє забезпечити підтримку формування звітів для деканату, викладачів і адміністративного персоналу структурного підрозділу університету.

Основні елементи архітектури системи наведено у таблиці 3.2.

Таблиця 3.2 – Основні елементи архітектури програмної системи

Елемент архітектури	Призначення
Робоча станція користувача	Середовище запуску desktop-застосунку
Desktop Application (Python + PyQt6)	Виконання прикладної логіки системи
UI	Взаємодія користувача із системою
Auth	Авторизація та перевірка ролей користувачів
Applications	Оброблення заяв на повторне вивчення
Enrollment	Організація повторного навчання
Results	Робота з результатами оцінювання
Reports	Формування звітів і експорту

Продовження таблиці 3.2

Styling	Оформлення графічного інтерфейсу
DataAccess	Взаємодія з базою даних
SQLiteDB	Зберігання структурованих даних
Export / Reports	Збереження експортованих файлів і звітів

Запропонована архітектура забезпечує модульність програмної системи, централізовану взаємодію з базою даних, підтримку локального зберігання інформації та можливість подальшого розширення функціональності. Використання компонентного підходу дозволяє спростити супровід програмного забезпечення, забезпечити ізоляцію функціональних модулів та підвищити ефективність реалізації процесів підтримки повторного вивчення дисципліни у структурному підрозділі університету.

3.3 Реалізація програмного забезпечення та функціональних модулів системи

Розроблення програмного забезпечення інформаційної системи підтримки повторного вивчення дисципліни виконано з використанням модульного підходу, що забезпечує розподіл функціональності між окремими програмними компонентами та спрощує подальший супровід і розширення системи. Реалізація програмної системи орієнтована на підтримку процесів авторизації користувачів, подання та оброблення заяв, організації повторного навчання, роботи з результатами повторного контролю, формування звітів і ведення журналу подій.

На рисунку 3.2 наведено фрагмент програмної реалізації структури бази даних системи.

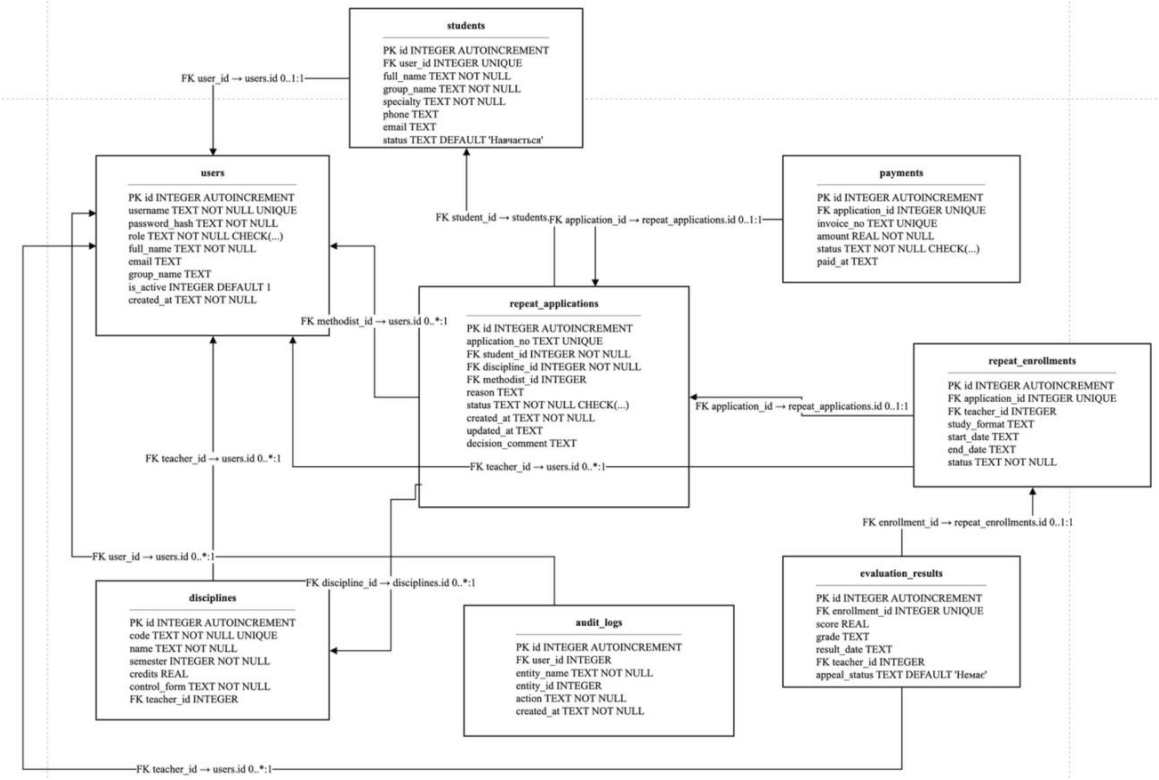

Рисунок 3.2 – Фрагмент програмної реалізації структури бази даних системи

Для зберігання інформації використовується локальна база даних SQLite, структура якої формується за допомогою SQL-скриптів у межах програмного модуля ініціалізації бази даних. Реалізовані таблиці забезпечують зберігання даних про користувачів, студентів, дисципліни, заявки на повторне вивчення, результати оцінювання, оплату та журналювання дій користувачів. Використання зовнішніх ключів і механізмів перевірки цілісності дозволяє підтримувати узгодженість даних між взаємопов’язаними сутностями системи.

Основні таблиці програмної системи наведено у таблиці 3.3.

Таблиця 3.3 – Основні таблиці бази даних системи

Таблиця	Призначення
users	Зберігання облікових записів користувачів
students	Зберігання інформації про студентів
disciplines	Дані про навчальні дисципліни
repeat_applications	Облік заяв на повторне вивчення
repeat_enrollments	Облік зарахування на повторне навчання

Продовження таблиці 3.3

evaluation_results	Зберігання результатів повторного контролю
payments	Облік фінансових операцій
audit_logs	Журналювання дій користувачів
notifications	Формування системних повідомлень

На рисунку 3.3 наведено програмну реалізацію SQL-структури окремих таблиць системи.

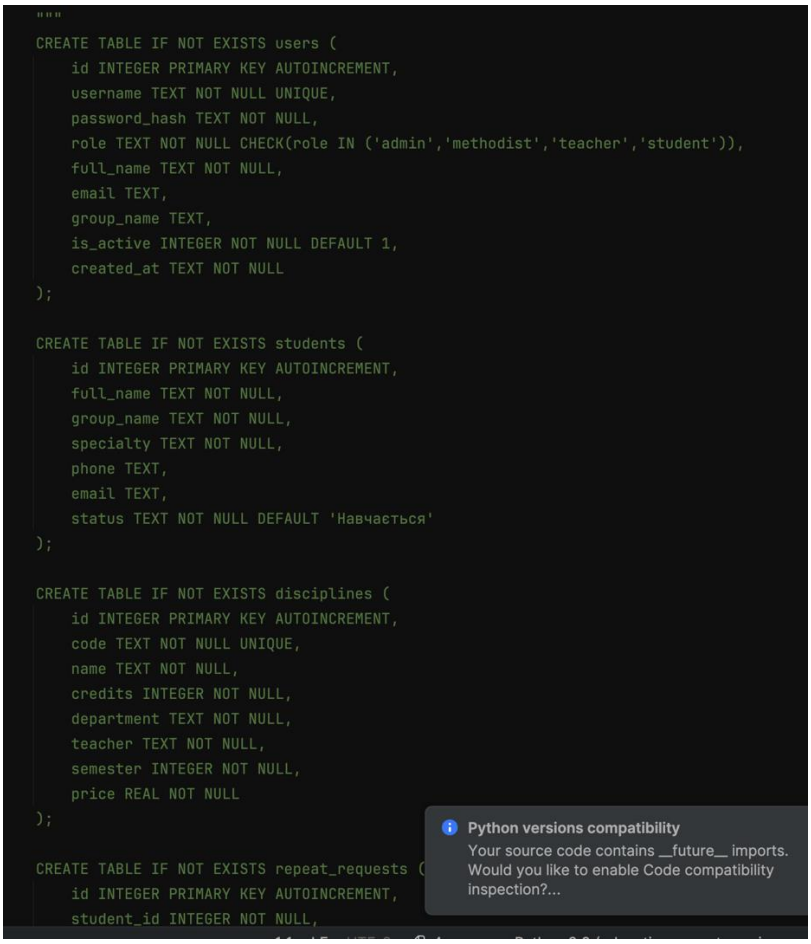

Рисунок 3.3 – Програмна реалізація SQL-структури таблиць системи

У процесі реалізації структури бази даних використано механізми PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK та DEFAULT, що дозволяють забезпечити унікальність записів, підтримку зв’язків між таблицями та контроль допустимих значень атрибутів. Для реалізації рольової моделі

доступу використовується поле role, яке визначає тип користувача системи: адміністратор, методист, викладач або студент.

Для підтримки процесу повторного навчання реалізовано модулі роботи із заявами, які забезпечують реєстрацію звернень студентів, перевірку умов допуску, зміну статусів заявок і формування рішень щодо повторного вивчення дисципліни. Після підтвердження заявки система автоматично створює запис про зарахування студента на повторне навчання та забезпечує можливість подальшого внесення результатів оцінювання.

На рисунку 3.4 наведено приклад SQL-запиту для отримання інформації про заявки на повторне вивчення дисципліни.

```
return self.query(
    sql: f"""SELECT rr.id, s.full_name AS student, s.group_name, d.code, d.name AS discipline, d.price,
                rr.reason, rr.status, rr.payment_status, rr.academic_year, rr.priority, rr.created_at, rr.notes
        FROM repeat_requests rr
        JOIN students s ON s.id=rr.student_id
        JOIN disciplines d ON d.id=rr.discipline_id
        {clause}
        ORDER BY rr.id DESC""",
    params,
)
```

Рисунок 3.4 – Реалізація SQL-запиту для формування інформації про заявки

Запити до бази даних реалізовано через окремий компонент доступу до даних, який формує SQL-вибірки, виконує об’єднання таблиць і повертає структуровані результати прикладним модулям системи. Для формування аналітичної інформації використовуються SQL-операції JOIN, ORDER BY та механізми фільтрації, що дозволяють отримувати дані про студентів, дисципліни, статуси заяв, результати повторного контролю та фінансові операції.

Для забезпечення безпеки облікових записів реалізовано механізм зберігання паролів у вигляді криптографічних хешів. Під час входу до системи здійснюється перевірка відповідності введених даних хешованому значенню, що знижує ризик компрометації облікової інформації користувачів.

Окремо в системі реалізовано модуль повідомлень і журналювання подій, який забезпечує фіксацію змін статусів заяв, результатів оцінювання та дій

користувачів. Це дозволяє підвищити контроль за роботою системи та забезпечити прозорість виконання процедур повторного вивчення дисципліни.

3.4 Алгоритмізація програмних модулів системи

Для забезпечення коректної роботи інформаційної системи підтримки повторного вивчення дисципліни було розроблено алгоритми функціонування основних програмних модулів, які реалізують процеси подання заяв, перевірки умов допуску, погодження повторного навчання, підтвердження оплат, організації повторного вивчення дисципліни та фіксації результатів повторного контролю.

Алгоритмізація програмних модулів дозволяє формалізувати послідовність виконання операцій у межах системи, забезпечити узгодженість оброблення даних та підтримати централізоване керування основними бізнес-процесами структурного підрозділу університету.

На рисунку 3.5 наведено блок-схему алгоритму подання та реєстрації заявки на повторне вивчення дисципліни.

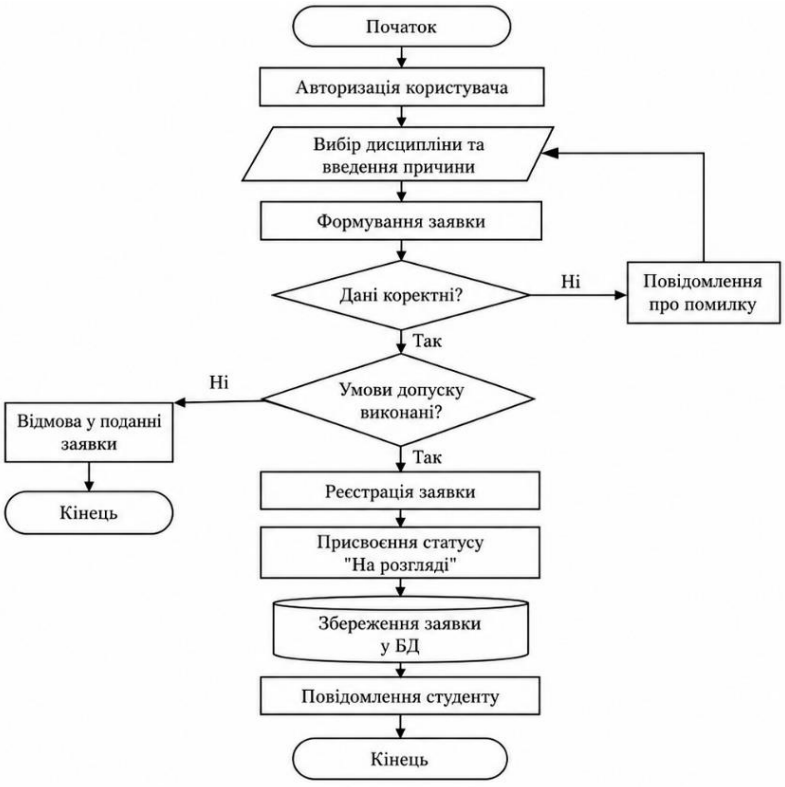

Рисунок 3.5 – Блок-схема алгоритму подання та реєстрації заявки

Подання заявки починається з авторизації користувача та вибору дисципліни для повторного вивчення. Після введення причини повторного проходження дисципліни система формує заявку та виконує перевірку коректності введених даних. У разі виявлення помилок користувач отримує відповідне повідомлення з можливістю повторного введення інформації.

Після успішної перевірки система аналізує виконання умов допуску до повторного вивчення дисципліни. Якщо студент не відповідає встановленим вимогам, заявка не реєструється та формується повідомлення про відмову. У разі успішного проходження перевірки заявка реєструється в базі даних, їй автоматично присвоюється статус «На розгляді», а студент отримує повідомлення про успішне подання звернення.

Основні етапи алгоритму подання заявки наведено у таблиці 3.4.

Таблиця 3.4 – Основні етапи алгоритму подання заявки

Етап	Призначення
Авторизація користувача	Перевірка облікових даних
Введення інформації	Формування параметрів заявки
Перевірка коректності	Контроль правильності введених даних
Перевірка умов допуску	Аналіз можливості повторного навчання
Реєстрація заявки	Збереження заявки у базі даних
Присвоєння статусу	Формування поточного стану заявки
Повідомлення студента	Інформування користувача

На рисунку 3.6 наведено блок-схему алгоритму розгляду заявки та підтвердження оплати.

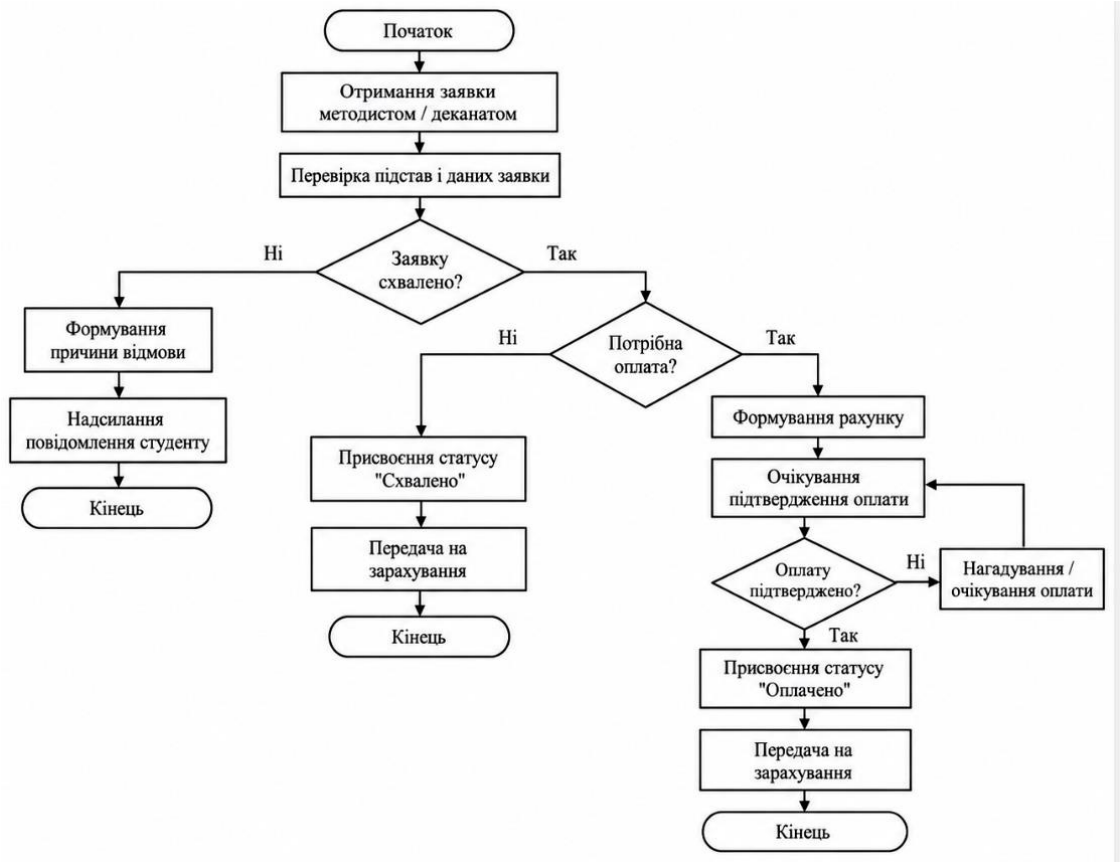

Рисунок 3.6 – Блок-схема алгоритму розгляду заявки та підтвердження оплати

Після надходження заявки методист або працівник деканату виконує перевірку поданих даних, аналізує підстави повторного вивчення дисципліни та приймає рішення щодо погодження або відмови. У випадку відмови система формує причину відхилення заявки та надсилає відповідне повідомлення студенту.

Якщо заявку погоджено, система перевіряє необхідність виконання оплати. У разі необхідності формується рахунок та здійснюється контроль підтвердження оплати. До моменту підтвердження платежу система може повторно формувати нагадування про оплату. Після підтвердження фінансової операції заявці присвоюється статус «Оплачено», після чого вона передається на етап зарахування студента на повторне вивчення дисципліни.

На рисунку 3.7 наведено блок-схему алгоритму зарахування студента на повторне вивчення дисципліни та фіксації результатів повторного контролю.

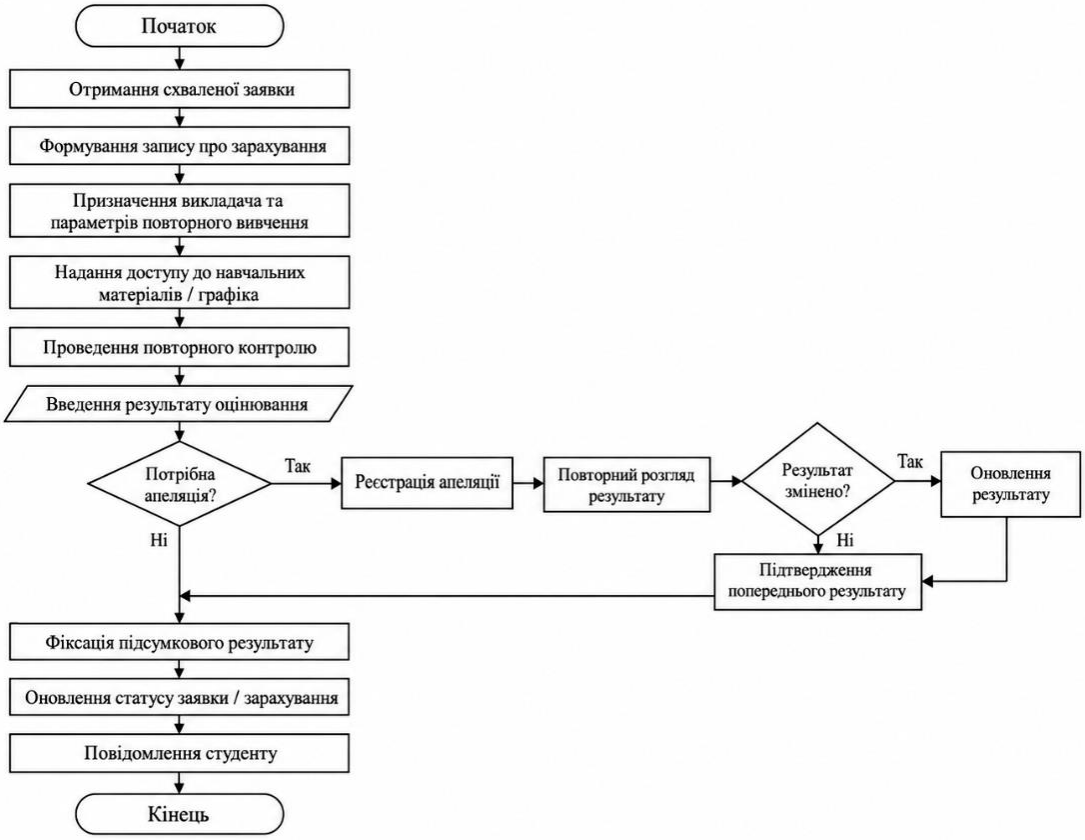

Рисунок 3.7 – Блок-схема алгоритму зарахування та фіксації результатів повторного контролю

Після погодження заявки система формує запис про зарахування студента та призначає викладача, відповідального за проведення повторного навчання і контролю знань. Далі студенту надається доступ до навчальних матеріалів, графіка проведення занять або консультацій, після чого виконується повторний контроль знань.

Результати оцінювання вводяться до системи та можуть бути передані на апеляційний розгляд. У разі подання апеляції здійснюється повторний перегляд результату оцінювання. Якщо результат змінено, система автоматично оновлює підсумковий результат; у протилежному випадку підтверджується попереднє значення оцінки. Після завершення процедури система фіксує остаточний результат, оновлює статус заявки та надсилає студенту відповідне повідомлення.

Розроблені алгоритми забезпечують формалізовану послідовність роботи програмних модулів системи, підтримують централізоване керування процесами повторного вивчення дисциплін та забезпечують узгоджену взаємодію між

користувачами, прикладними модулями й базою даних. Використання алгоритмічного підходу створює основу для подальшої реалізації програмного забезпечення та проведення функціонального тестування системи.

3.5 Висновки до третього розділу

У третьому розділі виконано проєктування архітектури програмної системи та розроблення алгоритмічного забезпечення інформаційної системи підтримки повторного вивчення дисципліни студентами структурного підрозділу університету. У процесі роботи обґрунтовано вибір технологічних засобів реалізації, сформовано архітектуру програмного забезпечення, реалізовано структуру бази даних і визначено алгоритми функціонування основних програмних модулів системи.

У результаті аналізу технологічних засобів для реалізації програмного забезпечення обрано стек Python, PyQt6 та SQLite, який забезпечує підтримку desktop-орієнтованої архітектури, локального зберігання даних, реалізацію графічного інтерфейсу користувача та централізовану роботу з реляційною базою даних. Використання модульного підходу дозволило розділити функціональність системи між окремими програмними компонентами, що спрощує подальший супровід і масштабування програмного забезпечення.

У межах архітектурного проєктування сформовано компонентну структуру системи, яка включає модулі автентифікації, керування заявами, організації повторного вивчення дисципліни, оцінювання результатів, формування звітів та централізованого доступу до даних. Розроблена архітектура забезпечує взаємодію між прикладними компонентами та локальною базою даних SQLite через окремий компонент доступу до даних, що підвищує підтримуваність програмної системи та ізолює бізнес-логіку від механізмів фізичного зберігання інформації.

У процесі реалізації програмного забезпечення створено структуру бази даних, реалізовано SQL-моделі таблиць, зовнішні ключі та механізми перевірки

цілісності даних. Програмна система забезпечує зберігання інформації про користувачів, студентів, дисципліни, заявки, результати повторного контролю, фінансові операції та журнал дій користувачів. Окремо реалізовано механізми авторизації, журналювання подій і формування повідомлень.

У межах алгоритмізації програмних модулів побудовано блок-схеми процесів подання заяв, розгляду та погодження повторного навчання, підтвердження оплат, зарахування студентів та фіксації результатів повторного контролю. Розроблені алгоритми забезпечують формалізовану послідовність виконання операцій, підтримку узгодженої взаємодії між користувачами та прикладними модулями системи, а також автоматизацію основних процесів підтримки повторного вивчення дисциплін.

Отримані результати третього розділу створюють основу для подальшого тестування програмного забезпечення, оцінювання ефективності роботи інформаційної системи та підготовки завершеної програмної реалізації для використання у структурному підрозділі університету.

4 ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ СИСТЕМИ

4.1 Тестування програмних модулів інформаційної системи підтримки повторного вивчення дисципліни

Тестування програмних модулів інформаційної системи підтримки повторного вивчення дисципліни студентами структурного підрозділу університету спрямоване на перевірку коректності функціонування основних підсистем, стабільності роботи програмного забезпечення, правильності оброблення даних користувачів, надійності роботи механізмів авторизації та достовірності формування результатів повторного навчання й звітної інформації. Методика тестування охоплює модульне, інтеграційне, функціональне та навантажувальне тестування, що забезпечує комплексну перевірку системи на різних рівнях її функціонування.

Загальний план тестування програмних модулів наведено у таблиці 4.1.

Таблиця 4.1 – План тестування основних програмних модулів системи

№	Модуль / компонент	Тип тестування	Тестові дії	Очікуваний результат
1	Модуль автентифікації	Модульне	Вхід у систему, перевірка ролей, оброблення помилок	Коректна авторизація та розмежування доступу
2	Модуль керування заявами	Функціональне	Подання, редагування та перевірка заяв	Коректна реєстрація заявок
3	Модуль перевірки допуску	Алгоритмічне	Аналіз умов повторного вивчення	Правильне визначення можливості подання заявки
4	Модуль організації повторного навчання	Інтеграційне	Формування запису про зарахування	Коректне створення записів повторного навчання
5	Модуль оцінювання результатів	Функціональне	Введення та оновлення результатів контролю	Правильне збереження результатів

Продовження таблиці 4.1

6	Модуль апеляцій	Функціональне	Реєстрація та повторний розгляд результатів	Коректне оновлення статусів і результатів
7	Модуль оплат	Інтеграційне	Формування рахунків і підтвердження оплат	Коректна зміна статусу заявки
8	Модуль звітності	Функціональне	Формування CSV-звітів і списків	Повна та структурована звітна інформація
9	Підсистема журналювання	Модульне	Запис дій користувачів і системних подій	Повнота та консистентність журналів
10	Продуктивність системи	Навантажувальне	Робота із великою кількістю заяв	Стабільна робота без втрати даних

Методика оцінювання результатів тестування базується на аналізі правильності роботи бізнес-логіки системи, стабільності взаємодії між прикладними модулями та коректності роботи з базою даних SQLite. Для функціональних модулів оцінюється правильність оброблення даних користувачів, формування заяв, оновлення статусів і підтримка процесів повторного навчання.

Особлива увага приділяється перевірці механізмів авторизації користувачів, контролю доступу до функціональних модулів та коректності оброблення помилок введення даних. У процесі тестування перевіряється стабільність роботи системи під час повторного подання заяв, оновлення результатів оцінювання, виконання апеляційних процедур і формування звітної документації.

Для інтеграційних компонентів оцінюється правильність взаємодії між прикладними модулями та компонентом доступу до даних. Контролюється коректність виконання SQL-запитів, збереження інформації у взаємопов’язаних таблицях і підтримка цілісності даних у базі SQLite. Додатково перевіряється правильність роботи механізмів зовнішніх ключів, оновлення статусів заяв та взаємодії між модулями повторного навчання, оплат і результатів оцінювання.

Навантажувальне тестування дозволяє оцінити стабільність роботи програмного забезпечення при одночасній роботі декількох користувачів та великій кількості заяв на повторне вивчення дисциплін. Під час тестування контролюється час відповіді системи, швидкість формування звітів, коректність роботи локальної бази даних і відсутність втрати інформації при виконанні послідовних операцій запису та оновлення даних.

У процесі тестування також перевіряється робота механізмів повідомлень і журналювання подій, які забезпечують фіксацію дій користувачів, змін статусів заяв і результатів повторного контролю. Це дозволяє забезпечити прозорість функціонування системи та підтримку контролю за виконанням процедур повторного навчання у структурному підрозділі університету.

4.2 Тестування інтерфейсних модулів інформаційної системи підтримки повторного вивчення дисципліни

Тестування інтерфейсних модулів інформаційної системи підтримки повторного вивчення дисципліни спрямоване на перевірку коректності роботи графічного інтерфейсу користувача, стабільності навігації між функціональними розділами системи, правильності відображення даних та зручності взаємодії користувача із програмним забезпеченням. Особлива увага приділяється перевірці роботи форм введення, таблиць, механізмів фільтрації, побудови аналітичних елементів та взаємодії GUI-компонентів із локальною базою даних SQLite.

Інтерфейс системи реалізовано у вигляді настільного застосунку на основі Python, PyQt6 та QSS-стилізації. Головною особливістю реалізованого інтерфейсу є підтримка єдиного стилю оформлення, централізованої навігації та структурованого подання інформації відповідно до ролей користувачів системи.

На рисунку 4.1 представлено форму авторизації користувача інформаційної системи.

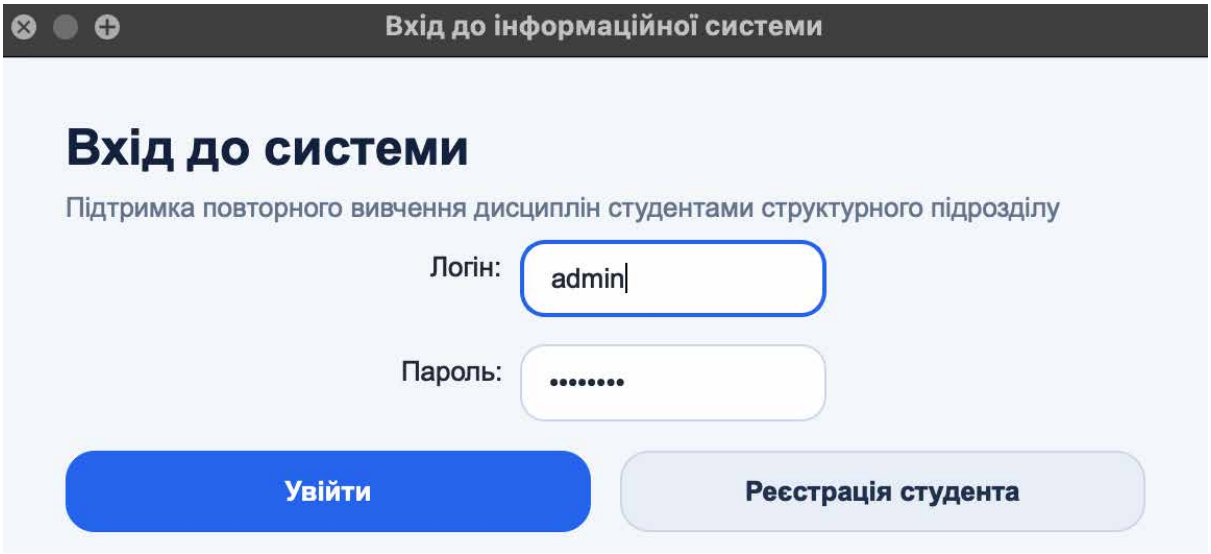

Рисунок 4.1 – Вікно авторизації користувача

Форма авторизації забезпечує введення логіна та пароля, перевірку облікових даних, оброблення помилок введення та контроль доступу до функціональних модулів системи. Під час тестування перевірялась коректність переходу між елементами форми, стабільність роботи кнопок входу та реєстрації, а також правильність повідомлень при помилковій автентифікації.

На рисунку 4.2 наведено форму реєстрації нового студента.

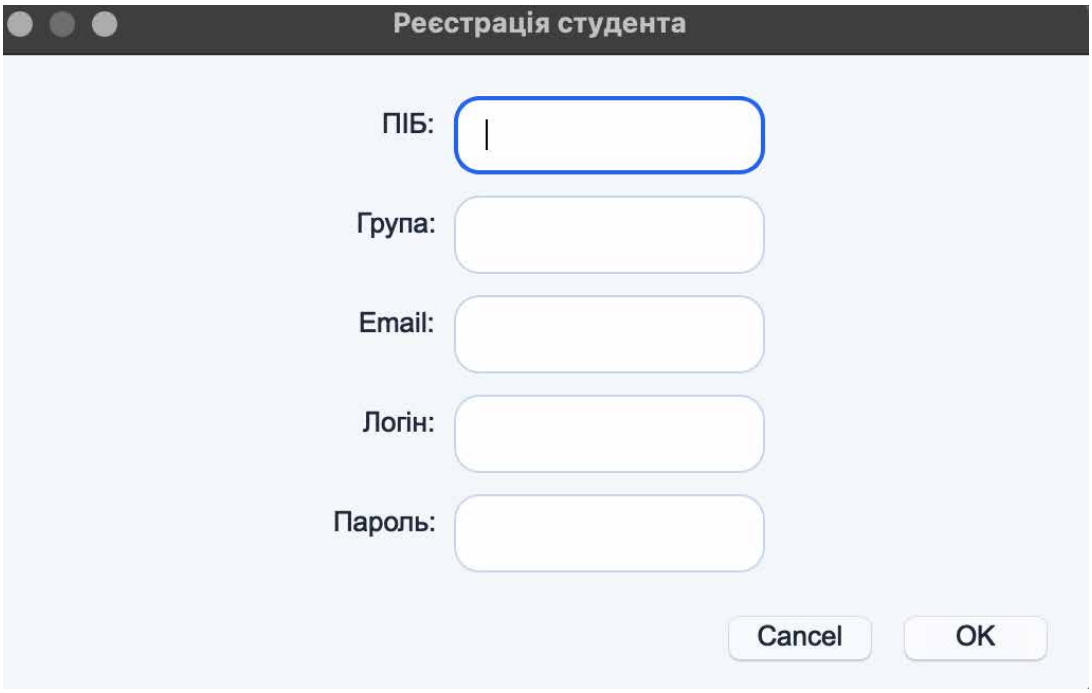

Рисунок 4.2 – Вікно реєстрації студента

Тестування модуля реєстрації включало перевірку введення персональних даних, валідацію полів, перевірку унікальності логінів та правильність

збереження інформації у таблицях бази даних. Додатково контролювалась стійкість інтерфейсу при введенні некоректних або неповних даних.

На рисунку 4.3 представлено головний інформаційний дашборд системи.

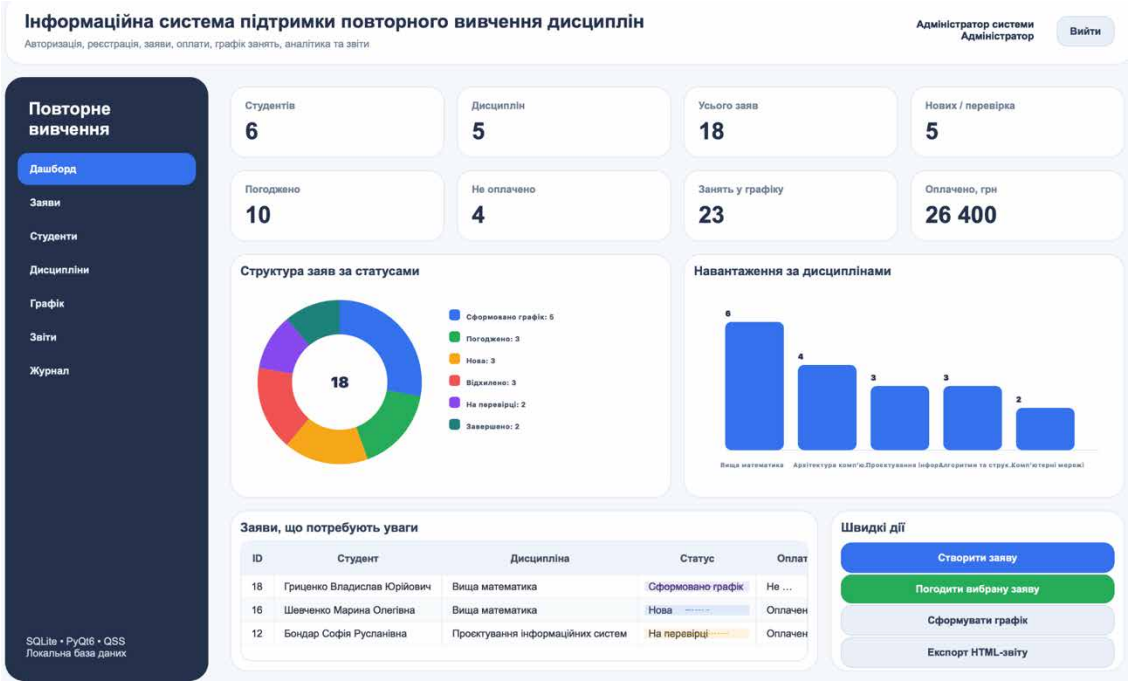

Рисунок 4.3 – Головне вікно інформаційної системи

Головний дашборд забезпечує відображення статистичних показників, аналітичних діаграм, навігаційного меню та швидкого доступу до основних функціональних модулів системи. Під час тестування перевірялась коректність оновлення статистичних даних, відображення графіків і стабільність роботи навігаційної панелі.

На рисунку 4.4 наведено інтерфейс модуля керування заявами на повторне вивчення дисциплін.

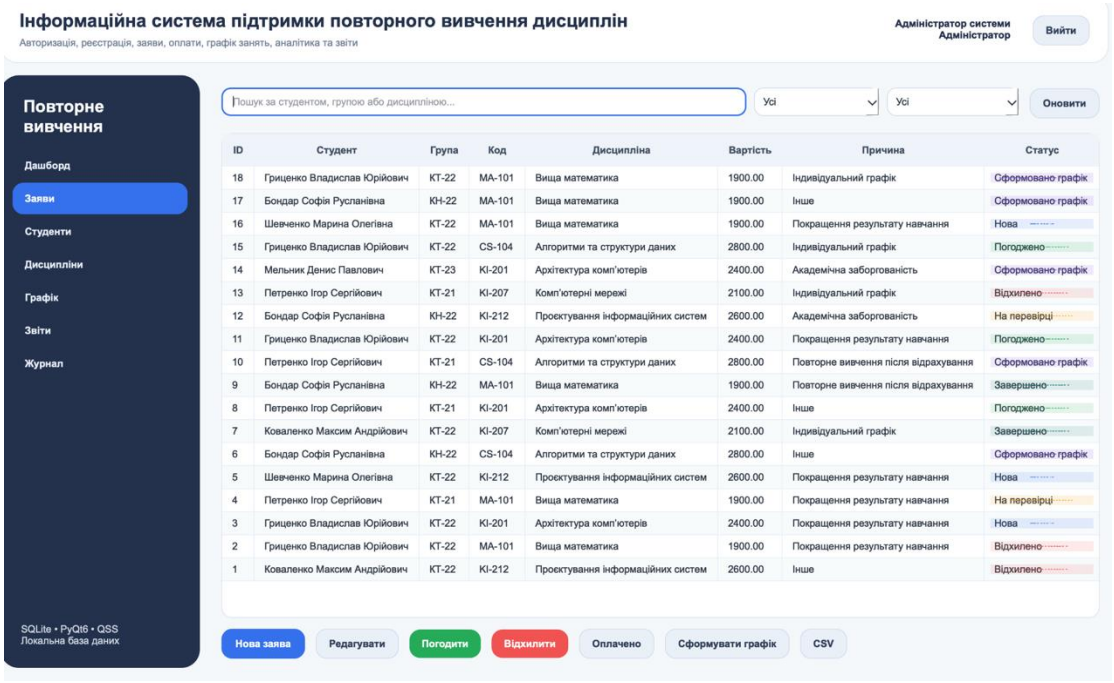

Рисунок 4.4 – Інтерфейс модуля керування заявами

Даний модуль забезпечує перегляд, пошук, фільтрацію, редагування та зміну статусів заяв. У процесі тестування перевірялась коректність роботи таблиці заяв, функцій сортування, фільтрів статусів, оновлення записів та взаємодії інтерфейсу із SQL-запитами до бази даних.

На рисунку 4.5 представлено інтерфейс модуля дисциплін.

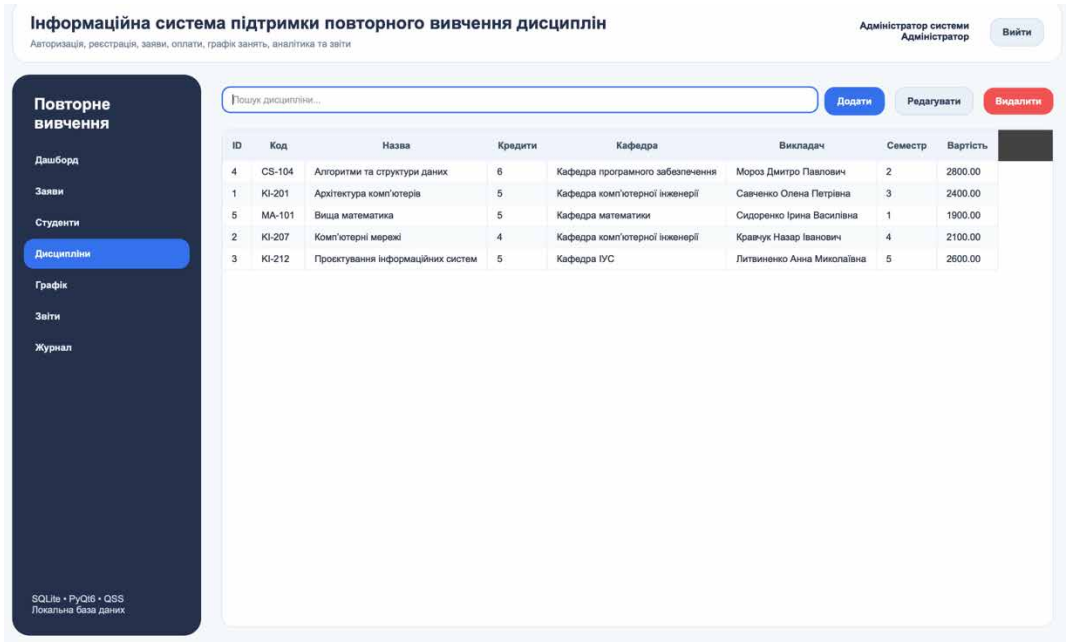

Рисунок 4.5 – Інтерфейс модуля дисциплін

Під час тестування модуля дисциплін перевірялась правильність додавання, редагування та видалення записів про дисципліни, а також коректність оновлення інформації у таблицях системи. Додатково оцінювалась стабільність роботи пошукового механізму та правильність відображення даних у табличній формі.

На рисунку 4.6 наведено інтерфейс модуля формування графіка занять.

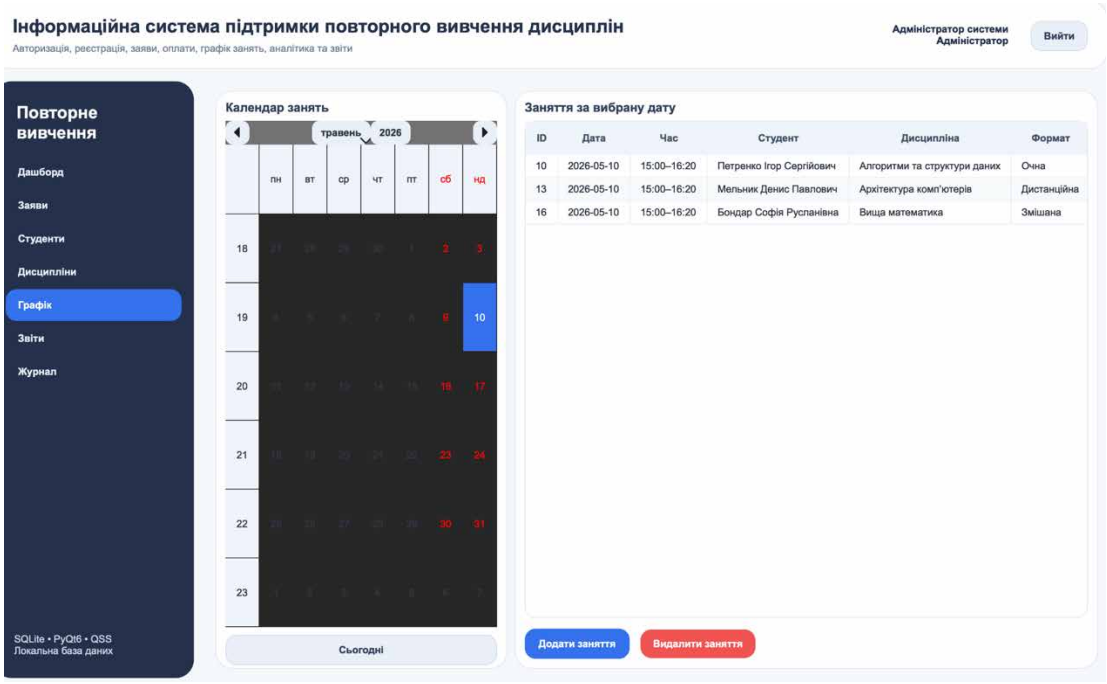

Рисунок 4.6 – Інтерфейс модуля графіка занять

Тестування даного модуля включало перевірку роботи календаря, формування записів занять, оновлення таблиці розкладу та правильності синхронізації графіка із заявами на повторне навчання. Окремо перевірялась стабільність роботи календарного компонента та швидкість оновлення інтерфейсу при зміні дати.

На рисунку 4.7 представлено модуль формування аналітичних звітів.

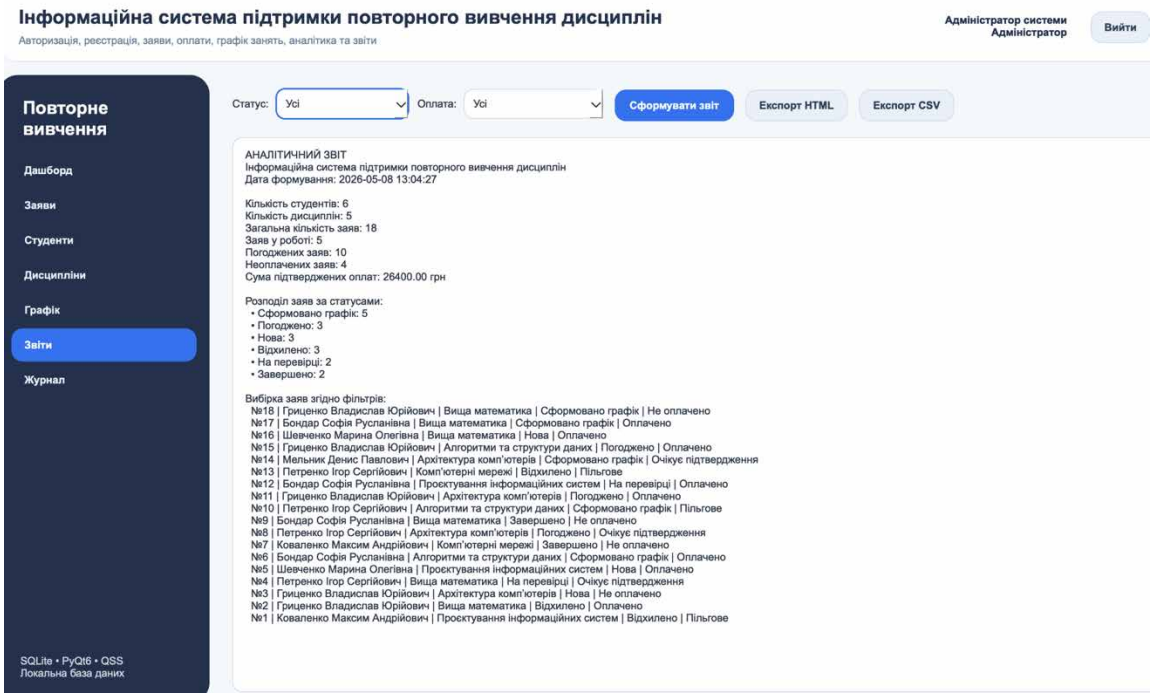

Рисунок 4.7 – Інтерфейс модуля звітності

Модуль звітності забезпечує формування текстових та табличних аналітичних звітів, експорт HTML і CSV-документів, а також відображення статистики щодо заяв і оплат. У процесі тестування перевірялась правильність формування звітних даних, коректність експорту та відповідність інформації даним бази SQLite.

На рисунку 4.8 наведено інтерфейс журналу дій користувачів.

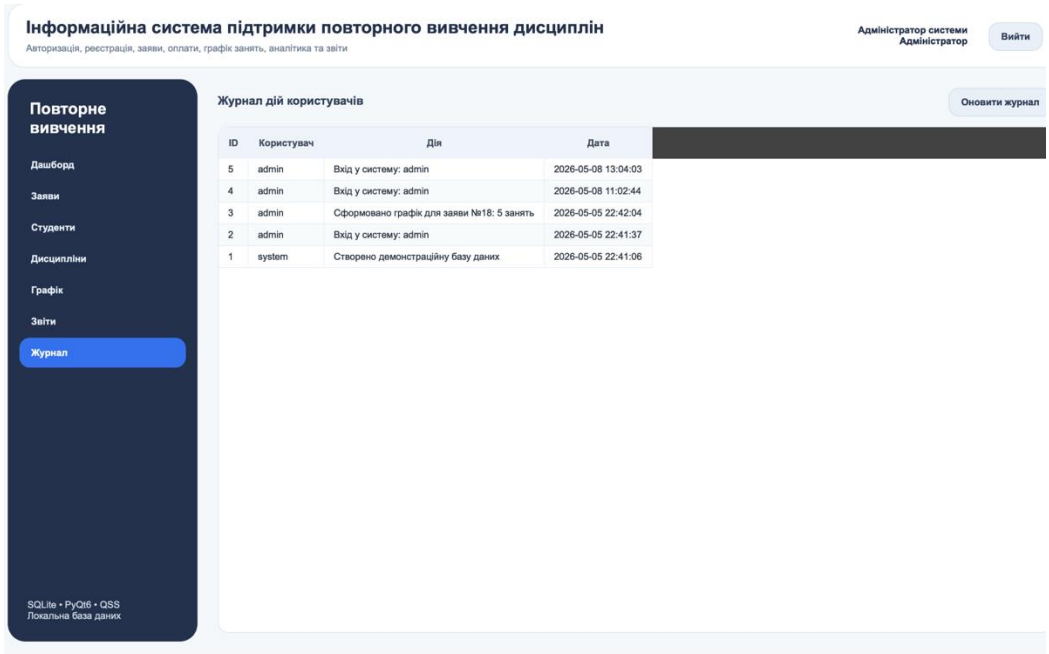

Рисунок 4.8 – Інтерфейс журналу подій системи

Журнал подій використовується для контролю дій користувачів, фіксації операцій у системі та забезпечення прозорості функціонування програмного забезпечення. Під час тестування перевірялась коректність запису подій, оновлення журналу та правильність відображення часових міток.

Основні результати тестування інтерфейсних модулів наведено у таблиці 4.2.

Таблиця 4.2 – Результати тестування інтерфейсних модулів системи

№	Інтерфейсний модуль	Перевірка	Результат
1	Авторизація	Вхід і перевірка доступу	Успішно
2	Реєстрація студентів	Валідація та збереження даних	Успішно
3	Дашборд	Оновлення статистики та графіків	Успішно
4	Модуль заяв	Фільтрація, зміна статусів	Успішно
5	Модуль дисциплін	CRUD-операції	Успішно
6	Модуль графіка	Формування занять	Успішно
7	Модуль звітності	Формування та експорт звітів	Успішно
8	Журнал подій	Запис та відображення подій	Успішно

Проведене тестування підтвердило коректність функціонування інтерфейсних модулів, стабільність роботи графічного інтерфейсу та правильність взаємодії між GUI-компонентами й локальною базою даних SQLite. Отримані результати засвідчують придатність інформаційної системи до практичного використання в умовах структурного підрозділу університету та підтверджують ефективність реалізованих інтерфейсних рішень.

4.3 Результати тестування та аналіз ефективності системи, склад інсталяційного пакету

Результати проведеного тестування інформаційної системи підтримки повторного вивчення дисципліни підтверджують коректність реалізації основних функціональних модулів, стабільність роботи програмного забезпечення та ефективність взаємодії між інтерфейсною, прикладною та інформаційною підсистемами. У процесі перевірки оцінювались швидкодія

системи, правильність виконання бізнес-процесів, стійкість роботи локальної бази даних SQLite, а також зручність взаємодії користувачів із графічним інтерфейсом.

Під час тестування було підтверджено коректність роботи механізмів авторизації, формування заяв, оброблення статусів, створення записів повторного навчання, формування графіків занять, збереження результатів оцінювання та експорту аналітичних звітів. Система забезпечує стабільну роботу при одночасному виконанні великої кількості операцій читання та оновлення даних, а також підтримує цілісність інформації під час взаємодії між прикладними модулями.

На рисунку 4.9 представлено діаграму розгортання та склад інсталяційного пакету інформаційної системи.

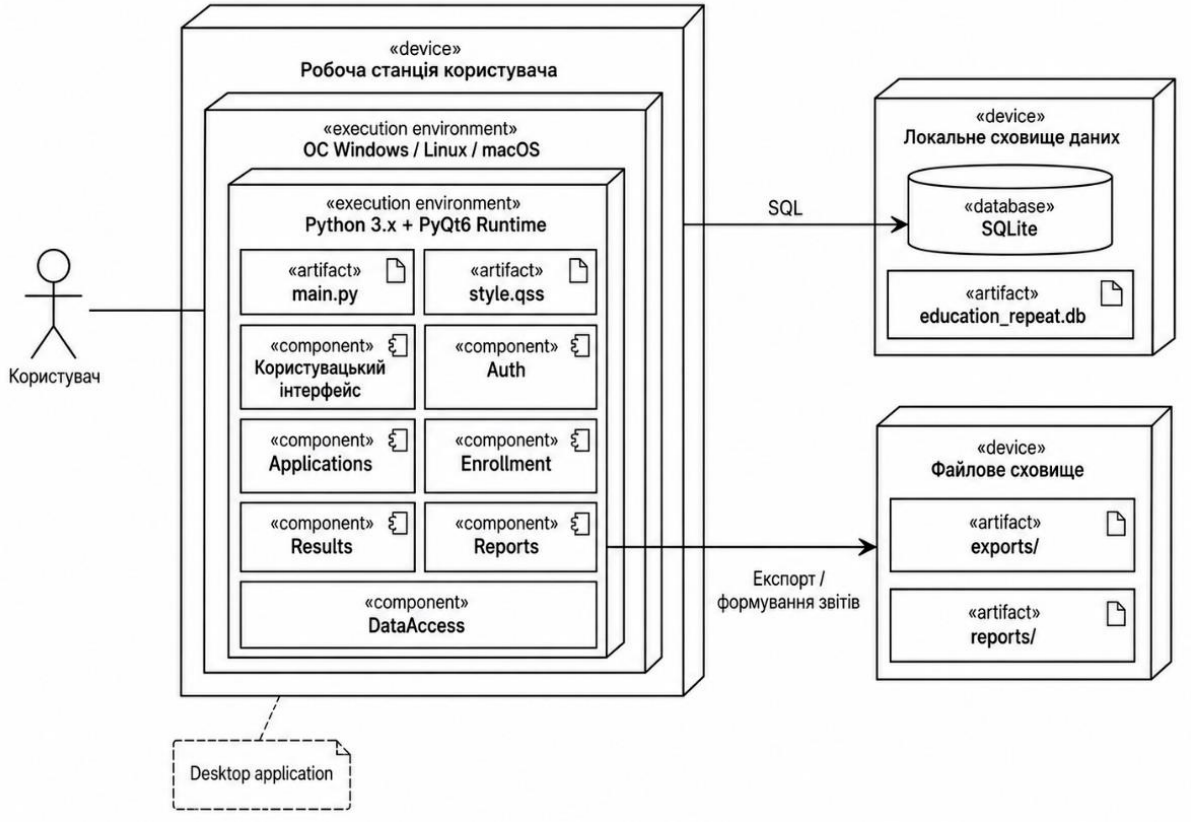

Рисунок 4.9 – Діаграма розгортання та склад інсталяційного пакету системи

Архітектура програмного забезпечення реалізована у вигляді локального desktop-застосунку, який функціонує в середовищі Python 3.x та PyQt6 Runtime.

Основні прикладні модулі працюють у межах робочої станції користувача та взаємодіють із локальною базою SQLite через компонент доступу до даних. Формування звітів та експорт документів здійснюється у файлове сховище системи.

У процесі аналізу ефективності оцінювались часові характеристики виконання основних операцій системи. Перевірка показала, що операції авторизації, відкриття модулів, формування заяв та оновлення статусів виконуються без помітних затримок, а взаємодія з локальною базою даних забезпечує достатню швидкодію для умов структурного підрозділу університету.

Основні результати оцінювання ефективності системи наведено у таблиці 4.3.

Таблиця 4.3 – Результати оцінювання ефективності системи

№	Показник	Результат
1	Час авторизації користувача	до 1 с
2	Формування заявки	до 2 с
3	Відкриття списку заяв	до 1 с
4	Формування графіка занять	до 3 с
5	Формування аналітичного звіту	до 2 с
6	Експорт CSV/HTML	до 2 с
7	Стабільність роботи SQLite	стабільна
8	Коректність збереження даних	підтверджено
9	Робота механізмів журналювання	коректна
10	Використання оперативної пам’яті	помірне

Отримані результати підтверджують ефективність обраної архітектури програмного забезпечення та доцільність використання локальної моделі зберігання даних на базі SQLite. Використання PyQt6 дозволило забезпечити достатню швидкодію графічного інтерфейсу, стабільну роботу навігаційних елементів і підтримку багатовіконної взаємодії користувача із системою.

Інсталяційний пакет системи включає основні програмні компоненти, конфігураційні файли, локальну базу даних та каталоги експорту звітної інформації. До складу інсталяційного пакету входять:

- головний файл запуску системи main.py;
- файл стилізації інтерфейсу style.qss;
- модулі авторизації, заяв, зарахування, результатів і звітності;
- компонент доступу до даних DataAccess;
- локальна база даних education_repeat.db;
- каталоги exports/ та reports/ для формування документів;
- бібліотеки середовища Python та PyQt6 Runtime.

Програмне забезпечення підтримує встановлення у середовищах Windows, Linux та macOS без необхідності використання окремого серверного обладнання. Це дозволяє спростити процес розгортання системи у структурних підрозділах університету та зменшити вимоги до технічної інфраструктури.

Проведене тестування та оцінювання ефективності підтвердили коректність реалізації програмної системи, стабільність її функціонування та придатність до використання у реальному освітньому середовищі. Реалізоване програмне забезпечення забезпечує автоматизацію процесів підтримки повторного вивчення дисциплін, підвищує ефективність оброблення заяв та покращує контроль за організацією повторного навчання студентів.

4.4 Висновки до четвертого розділу

У четвертому розділі виконано тестування та оцінювання ефективності інформаційної системи підтримки повторного вивчення дисципліни студентами структурного підрозділу університету. Проведено комплексну перевірку функціонування програмних модулів, графічного інтерфейсу користувача, механізмів роботи з базою даних SQLite та підсистем формування звітної інформації.

У процесі тестування підтверджено коректність роботи модулів авторизації, реєстрації студентів, керування заявами, організації повторного навчання, формування графіків занять, оброблення результатів оцінювання та експорту звітів. Перевірка функціональних сценаріїв показала правильність реалізації бізнес-логіки системи та узгодженість взаємодії між прикладними компонентами програмного забезпечення.

Проведене тестування інтерфейсних модулів підтвердило стабільність роботи графічного інтерфейсу, коректність навігації між функціональними розділами системи та правильність відображення інформації у таблицях, аналітичних блоках і формах введення даних. Реалізований інтерфейс забезпечує зручність взаємодії користувачів із системою та підтримує централізоване керування основними процесами повторного вивчення дисциплін.

У результаті аналізу ефективності встановлено, що програмне забезпечення забезпечує достатню швидкодію при виконанні основних операцій, стабільну роботу локальної бази даних та коректне функціонування механізмів оброблення інформації. Використання технологій Python, PyQt6 та SQLite дозволило реалізувати компактну, стабільну та достатньо продуктивну desktop-систему без необхідності використання окремої серверної інфраструктури.

Також визначено склад інсталяційного пакету програмної системи, який включає прикладні модулі, конфігураційні файли, локальну базу даних та каталоги експорту звітної інформації. Реалізована структура програмного забезпечення забезпечує простоту розгортання, супроводу та подальшого розширення функціональності системи.

Отримані результати тестування підтверджують готовність інформаційної системи підтримки повторного вивчення дисципліни до використання у реальному освітньому середовищі та засвідчують ефективність автоматизації процесів подання заяв, організації повторного навчання, контролю результатів та формування аналітичної звітності.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне завдання розроблення інформаційної системи підтримки повторного вивчення дисципліни студентами структурного підрозділу університету. Актуальність роботи зумовлена необхідністю автоматизації процесів подання та оброблення заяв, організації повторного навчання, контролю результатів оцінювання та формування аналітичної звітності в умовах зростання обсягів освітньої інформації та підвищення вимог до ефективності управління освітнім процесом.

У першому розділі виконано системний аналіз предметної області повторного вивчення дисциплін у закладах вищої освіти. Проведено аналіз існуючих підходів і програмних рішень, визначено їх основні переваги та недоліки. Сформовано функціональні, нефункціональні та безпекові вимоги до програмної системи, а також визначено структуру вхідних і вихідних даних. Результати аналізу дозволили обґрунтувати необхідність створення спеціалізованої інформаційної системи підтримки повторного вивчення дисциплін.

У другому розділі виконано проектування інформаційного та програмного забезпечення системи. Побудовано логічну модель даних у вигляді ER-діаграми, що відображає структуру взаємозв’язків між основними сутностями предметної області. Розроблено UML-діаграму класів, діаграми кооперації, компонентів, пакетів та розгортання системи. Сформовано фізичну модель бази даних SQLite та визначено архітектуру взаємодії прикладних модулів програмного забезпечення.

У третьому розділі виконано програмну реалізацію інформаційної системи. Реалізовано desktop-застосунок із використанням Python, PyQt6 та SQLite. Розроблено модулі авторизації користувачів, керування заявами, організації повторного навчання, формування графіка занять, оцінювання результатів, аналітики та звітності. Реалізовано механізми журналювання подій,

формування аналітичних показників та експорту звітної інформації у зовнішні файли. Також виконано алгоритмізацію основних бізнес-процесів системи та описано логіку функціонування ключових програмних модулів.

У четвертому розділі проведено тестування та оцінювання ефективності розробленого програмного забезпечення. Виконано модульне, функціональне, інтеграційне та навантажувальне тестування системи. Перевірено коректність роботи інтерфейсних модулів, стабільність взаємодії з локальною базою даних SQLite, правильність оброблення заяв та формування звітної інформації. Отримані результати підтвердили працездатність системи, достатню швидкодію та готовність програмного забезпечення до використання у реальному освітньому середовищі.

У результаті виконання кваліфікаційної роботи досягнуто поставленої мети – розроблено інформаційну систему підтримки повторного вивчення дисципліни студентами структурного підрозділу університету, яка забезпечує автоматизацію процесів подання заяв, контролю статусів, організації повторного навчання, формування графіків занять, ведення результатів оцінювання та створення аналітичної звітності.

Практичне значення роботи полягає у можливості використання розробленої системи для підвищення ефективності управління процесами повторного навчання, зменшення навантаження на працівників деканату та методичних підрозділів, скорочення часу оброблення заяв і покращення контролю за освітнім процесом. Розроблена система може бути адаптована для використання в інших структурних підрозділах закладів вищої освіти та доповнена новими функціональними можливостями.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Про освіту : Закон України від 05.09.2017 № 2145-VIII // База даних «Законодавство України» / Верховна Рада України. URL: https://zakon.rada.gov.ua/go/2145-19
2. Про вищу освіту : Закон України від 01.07.2014 № 1556-VII // База даних «Законодавство України» / Верховна Рада України. URL: https://zakon.rada.gov.ua/go/1556-18
3. Положення про організацію освітнього процесу в Національному університеті біоресурсів і природокористування України. Київ : НУБіП України, 2025. URL: https://nubip.edu.ua/polozhennya-pro-orhanizatsiyu-osvitnoho-protsesu-v-natsionalnomu-universyteti-bioresursiv-i
4. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Київ : ДП «УкрНДНЦ», 2016. URL: https://kubg.edu.ua/images/stories/podii/2017/06_21_posylannia/dstu_8302.pdf
5. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. Київ : ДП «УкрНДНЦ», 2016. URL: https://op.edu.ua/document/16872
6. MoodleDocs : official documentation / Moodle. URL: https://docs.moodle.org/
7. About Moodle / MoodleDocs. URL: https://docs.moodle.org/en/About_Moodle
8. Canvas LMS API Documentation / Instructure Developer Documentation Portal. URL: https://developerdocs.instructure.com/services/canvas
9. Blackboard Product Documentation / Anthology. URL: https://help.anthology.com/blackboard-product-documentation.html
10. PeopleSoft Campus Solutions : documentation / Oracle. URL: https://docs.oracle.com/en/applications/peoplesoft/campus-solutions/index.html

11. PeopleSoft Campus Solutions 9.2 PeopleBooks / Oracle. URL: https://docs.oracle.com/cd/F85283_01/psft/index.html
12. Campus Solutions / Oracle. URL: https://docs.oracle.com/cd/E52319_01/infoportal/cs.html
13. eLearning : Odoo 19.0 documentation / Odoo S.A. URL: https://www.odoo.com/documentation/19.0/applications/websites/elearning.html
14. Python 3 Documentation / Python Software Foundation. URL: https://docs.python.org/3/
15. The Python Standard Library / Python Software Foundation. URL: https://docs.python.org/3/library/index.html
16. sqlite3 — DB-API 2.0 interface for SQLite databases / Python Software Foundation. URL: https://docs.python.org/3/library/sqlite3.html
17. hashlib — Secure hashes and message digests / Python Software Foundation. URL: https://docs.python.org/3/library/hashlib.html
18. csv — CSV File Reading and Writing / Python Software Foundation. URL: https://docs.python.org/3/library/csv.html
19. venv — Creation of virtual environments / Python Software Foundation. URL: https://docs.python.org/3/library/venv.html
20. PyQt Components / Riverbank Computing. URL: https://riverbankcomputing.com/software/pyqt/intro
21. Qt for Python Documentation / The Qt Company. URL: https://doc.qt.io/qtforpython-6/
22. Qt Style Sheets Reference / The Qt Company. URL: https://doc.qt.io/qt-6/stylesheet.html
23. SQLite Documentation / SQLite Consortium. URL: https://sqlite.org/docs.html
24. SQLite Foreign Key Support / SQLite Consortium. URL: https://sqlite.org/foreignkeys.html
25. Datatypes In SQLite / SQLite Consortium. URL: https://sqlite.org/datatype3.html

26. Unified Modeling Language Specification / Object Management Group. URL: https://www.omg.org/spec/UML/
27. Unified Modeling Language / Object Management Group. URL: https://www.omg.org/uml/
28. Authentication Cheat Sheet / OWASP Cheat Sheet Series. URL: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html
29. Password Storage Cheat Sheet / OWASP Cheat Sheet Series. URL: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html
30. Authorization Cheat Sheet / OWASP Cheat Sheet Series. URL: https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html
31. PyInstaller Manual : PyInstaller documentation. URL: https://www.pyinstaller.org/
32. Chacon S., Straub B. Pro Git. 2nd ed. New York : Apress, 2014. URL: https://git-scm.com/book/en/v2
33. Silberschatz A., Korth H. F., Sudarshan S. Database System Concepts. 7th ed. New York : McGraw-Hill Education, 2019. URL: https://www.db-book.com/
34. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner's Approach. 9th ed. New York : McGraw-Hill Education, 2020. URL: https://www.mheducation.com/highered/product/software-engineering-practitioners-approach-pressman.html

ДОДАТОК А

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, Кириченко Владислав Сергійович, здобувач(ка) НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота на тему «Інформаційна система підтримки надання освітніх послуг з повторного вивчення дисциплін студентами структурного підрозділі університету» є результатом моєї особистої праці.

2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.

3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне): [✓] інструменти генеративного ШІ не використовувалися.

[✓] інструменти ШІ (ChatGPT) були використані для:

- [ ] перекладу іншомовних джерел;
- [✓] стилістичного редагування та перевірки граматики; ▪
- [✓] допомоги у написанні/відлагодженні програмного коду;
- [ ] інше: _____.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«22» травня 2026 р. _____ **Владислав Кириченко**