НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету	**Завідувач кафедри комп'ютерних наук**
_____ **Ігор БОЛБОТ**	_____ **Белла ГОЛУБ**
(підпис)	(підпис)
«___» _____ 2026 р.	«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Програмне забезпечення для автоматизованого обліку та управління навчальним процесом»

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Гарант освітньої програми к.т.н., доцент	_____ (підпис)	**Ганна ВАЙГАНГ**
Керівник бакалаврської кваліфікаційної роботи к.е.н., ст.викладач	_____ (підпис)	**Дмитро НІКОЛАЄНКО**
Виконав	_____ (підпис)	**Олександр ЗАЄЦЬ**

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук

к.т.н., доцент _____ Белла ГОЛУБ
(підпис)

«___» _____ 2025 р.

ЗАВДАННЯ
ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ

_____
Заєць Олександр Васильович

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Тема бакалаврської кваліфікаційної роботи

«Програмне забезпечення для автоматизованого обліку та управління навчальним процесом»

затверджена наказом від «19» грудня 2025 р. № 3204 «С»
Термін подання завершеної роботи на кафедру «22» травня 2026 р.
Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту): Технологічний стек розробки: мова програмування Python, вебфреймворк Django (Backend) , мови та інструменти веб-розмітки HTML5, CSS3, JavaScript (Frontend) , об'єктно-реляційна система управління базами даних PostgreSQL
Перелік питань, що підлягають дослідженню:
Аналіз діяльності закладу вищої освіти як складної предметної області, дослідження нормативних вимог та стану цифровізації адміністративних процесів.
Перелік графічних документів (за необхідності).
_____

Дата видачі завдання «19» грудня 2025 р.

Керівник бакалаврської кваліфікаційної роботи _____ (підпис) **Дмитро НІКОЛАЄНКО**

Завдання взяв до виконання _____ (підпис) **Олександр Заєць**

ЗМІСТ

ВСТУП

Актуальність теми. Сучасні тенденції цифровізації системи вищої освіти в Україні вимагають впровадження ефективних інструментів для автоматизації адміністративних процесів. Одним із ключових аспектів управління навчальним закладом є облік успішності здобувачів, що в умовах використання застарілих паперових чи неінтегрованих цифрових методів призводить до значних часових витрат, ризиків виникнення помилок та порушення цілісності даних. У зв'язку з цим, розробка програмного забезпечення для автоматизації обліку успішності є актуальним завданням, спрямованим на підвищення прозорості навчального процесу та оптимізацію роботи викладацького і адміністративного складу.

Мета роботи полягає у проектуванні та розробці інформаційної системи обліку успішності здобувачів вищої освіти, що забезпечує надійне зберігання, швидку обробку та аналітичну підтримку прийняття управлінських рішень у закладі освіти.

Для досягнення поставленої мети було вирішено такі завдання:

1. Проаналізувати стан автоматизації обліку успішності в закладах вищої освіти та визначити основні функціональні вимоги до майбутньої системи.
2. Обґрунтувати вибір архітектурного шаблону та технологічного стека для реалізації програмного продукту.
3. Здійснити проектування структури бази даних та логіки взаємодії компонентів системи.
4. Розробити програмний комплекс, що реалізує модулі автентифікації, обліку оцінок та генерації звітності.
5. Провести тестування системи для підтвердження її працездатності, стабільності та відповідності встановленим вимогам.

Об'єкт дослідження – процес обліку успішності здобувачів вищої освіти в межах інформаційно-освітнього середовища.

Предмет дослідження – методи та технології розробки програмного забезпечення для автоматизації ведення даних про успішність студентів.

Наукова новизна та практичне значення. Практична цінність роботи полягає у створенні інструменту, який дозволяє перевести процес обліку успішності у цифровий формат, забезпечуючи захист даних, зручність доступу для викладачів та можливість формування аналітичних звітів у реальному часі. Розроблена система може бути інтегрована в поточну IT-інфраструктуру вищого навчального закладу, сприяючи підвищенню ефективності управління якістю освіти.

Розділ 1. Аналіз предметної області

1.1. Аналіз діяльності закладу освіти як предметної області

Заклад вищої освіти (ЗВО) є складною соціально-технічною системою, функціонування якої спрямоване на реалізацію освітнього процесу, наукову діяльність та підготовку фахівців за визначеними спеціальностями. З позицій системного аналізу, діяльність закладу освіти можна розглядати як сукупність взаємопов'язаних підсистем, що забезпечують перетворення вхідних ресурсів – абітурієнтів, нормативних вимог та матеріальної бази – у результат у вигляді підготовлених фахівців [3, с. 45].

Основним регулятором освітньої діяльності в Україні є Закон України «Про вищу освіту», який визначає правові, організаційні та фінансові засади функціонування ЗВО [1, ст. 20]. Згідно з цим нормативним документом, навчальний процес базується на кредитно-модульній системі, що вимагає ретельного обліку навчального навантаження, академічних кредитів ECTS та систематичного контролю успішності здобувачів.

Процес управління в закладі освіти охоплює кілька ключових напрямів, які потребують автоматизації:

- Облік контингенту: формування та ведення особових справ, відстеження руху студентів (зарахування, переведення, відрахування) [4, с. 189].
- Організація освітнього процесу: формування індивідуальних навчальних планів, контроль виконання графіків навчання та розподіл дисциплін [5, с. 120].
- Моніторинг якості навчання: облік результатів поточного та семестрового контролю, формування рейтингів успішності [4, с. 191].

Водночас у багатьох закладах освіти ці процеси залишаються частково паперовими або виконуються в розрізнених програмних середовищах, що створює проблему низької оперативності обробки даних та ризик виникнення людських помилок. Для усунення цих недоліків необхідно розробити інтегровану програмну систему, яка б забезпечила централізоване зберігання

даних та автоматизацію звітності, що є критично важливим для прийняття ефективних управлінських рішень [4, с. 193].

Саме тому предметною областю розробки є автоматизація функціональних обов'язків працівників деканату та кафедр, спрямована на оптимізацію управління навчальним процесом у відповідності до сучасних вимог законодавства України [1, ст. 35; 3, с. 150].

1.2. Моделювання процесів обліку та управління навчальним процесом (діаграми UML, функціональні/контекстні діаграми)

Проектування автоматизованої системи управління навчальним процесом базується на методології об'єктно-орієнтованого аналізу, що дозволяє формалізувати бізнес-процеси та структуру даних закладу вищої освіти [2, с. 16]. Для візуалізації архітектури системи було використано уніфіковану мову моделювання UML (Unified Modeling Language), яка є стандартом де-факто у розробці програмного забезпечення [2, с. 18].

Першочерговим етапом проектування є визначення функціональних вимог до системи. На рис. 1.1 наведено діаграму варіантів використання (Use Case Diagram), яка визначає взаємодію основних акторів – Студента, Викладача та працівника Деканату – із системою.

Як відображено на діаграмі (див. рис. 1.1), система забезпечує розмежування прав доступу, що дозволяє автоматизувати такі процеси: реєстрацію користувачів, ведення успішності, планування навчального навантаження та формування звітності. Зазначена структура прецедентів відповідає завданням оптимізації управління навчальним процесом, визначеним у наукових дослідженнях з питань автоматизації ЗВО [4, с. 192].

Для проектування структури бази даних та архітектури програмного коду було побудовано діаграму класів (рис. 1.2), яка відображає об'єктно-орієнтовану модель предметної області.

Рис. 1.1 – Діаграма варіантів використання системи управління

Джерело: [1] та [5]

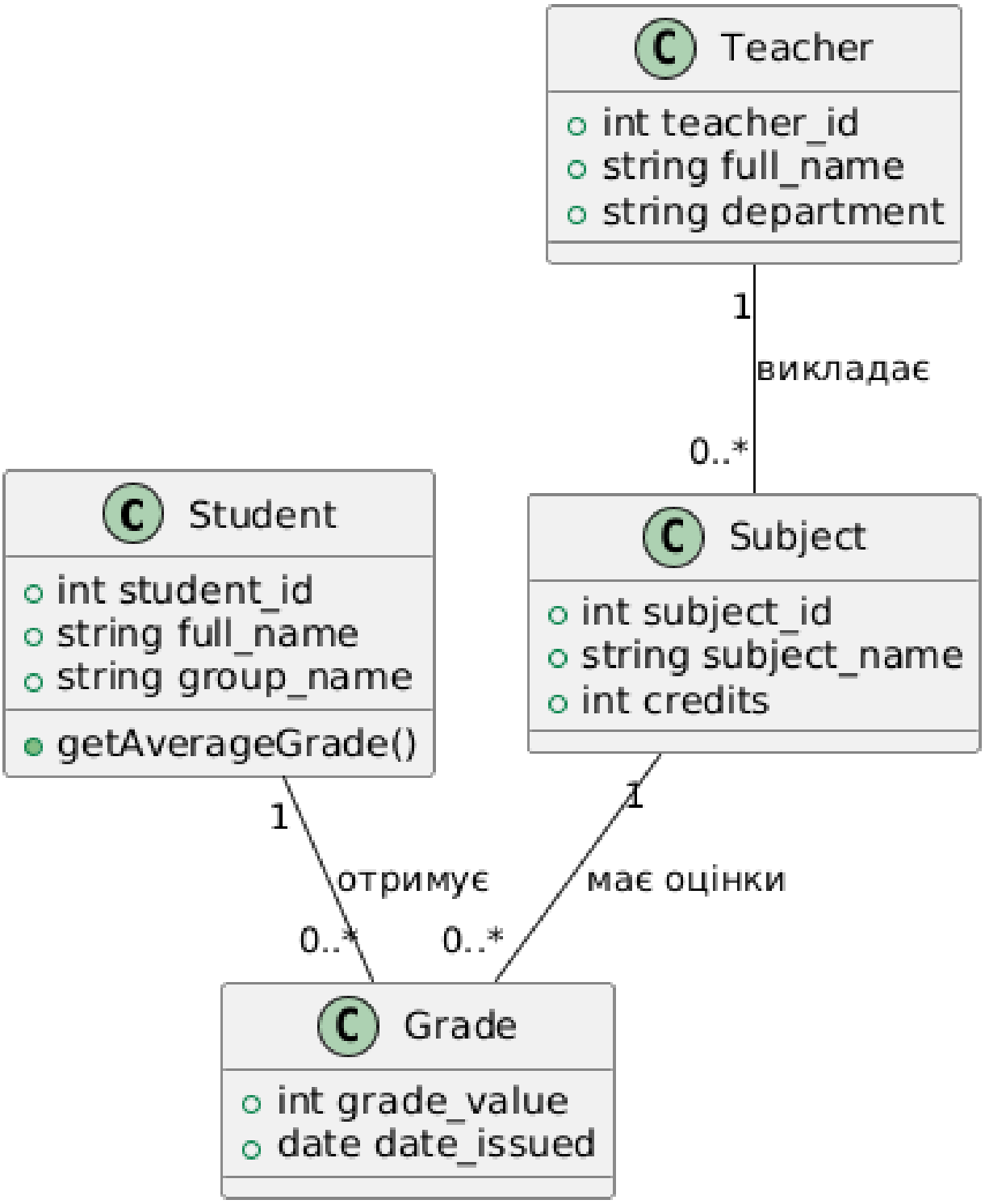


Рис. 1.2 – Діаграма класів предметної області системи

Джерело: [2]

Діаграма класів (рис. 1.2) демонструє логічні зв'язки між основними сутностями: Student, Teacher, Subject та Grade. Використання асоціацій між цими класами забезпечує цілісність даних та дозволяє реалізувати механізми

формування індивідуальних освітніх траєкторій, що є критично важливим для ефективного функціонування сучасного закладу вищої освіти [3, с. 150; 4, с. 191].

Розроблені UML-моделі є концептуальною основою для подальшої реалізації програмного продукту, забезпечуючи високий рівень структурованості та масштабованості системи відповідно до принципів об'єктно-орієнтованого проектування [2, с. 160].

1.3. Огляд існуючих аналогічних систем автоматизації навчального процесу

Сучасний стан цифровізації закладів вищої освіти (ЗВО) характеризується активним впровадженням інформаційно-аналітичних інструментів, спрямованих на підвищення якості освітньої діяльності. Згідно з пріоритетами державної політики, цифрова трансформація освіти є ключовим чинником забезпечення прозорості та ефективності управління [6].

Ринок освітніх ІТ-рішень представлений різноманітними платформами, які можна класифікувати на:

1. Системи аналітичного моніторингу та саморефлексії, такі як SELFIE (Self-reflection on Effective Learning by Fostering Innovation through Educational Technologies), що впроваджується під егідою Європейської Комісії та МОН України для оцінки цифрової зрілості закладу [7, 10].
2. Платформи для масових відкритих онлайн-курсів (MOOC), зокрема Coursera та Udemy, які фокусуються на наданні доступу до освітнього контенту та розвитку навичок [8, 9].
3. Спеціалізовані системи управління навчальним процесом (SIS/LMS), орієнтовані на автоматизацію адміністративних процесів, формування звітності та облік контингенту здобувачів [4].

Для об'єктивної оцінки функціональних можливостей вказаних систем було проведено порівняльний аналіз, результати якого наведено у таблиці 1.1. Аналіз здійснено з використанням методології якісної оцінки ІС за критеріями ISO/IEC 25010 [3].

Таблиця 1.1 – Порівняльна характеристика функціональних можливостей систем управління та навчання

Критерій порівняння	SELFIE	Coursera	Udemy
Функціональна повнота	Висока (аналіз)	Середня (курси)	Середня (курси)
Інтеграція з держреєстрами	Ні	Ні	Ні

Продовження таблиці 1.1

Зручність використання	Висока	Висока	Висока
Масштабованість	Висока	Висока	Висока
Призначення системи	Моніторинг	Навчання	Навчання

Як свідчать дані таблиці 1.1, глобальні платформи (Coursera, Udemy) забезпечують високу доступність освітнього контенту, проте не орієнтовані на виконання адміністративних функцій, необхідних для внутрішнього обліку ЗВО [8, 9]. Інструмент SELFIE є ефективним для стратегічного планування, проте не забезпечує операційного обліку успішності та руху здобувачів, що є вимогою Закону України «Про вищу освіту» [1, 7, 10].

Аналіз продемонстрував, що існуючі рішення не забезпечують комплексної автоматизації адміністративних процесів з урахуванням особливостей вітчизняного законодавства та Типового положення про організацію освітнього процесу [5]. Таким чином, розробка спеціалізованої системи, що поєднує функціонал операційного обліку та можливість інтеграції з державними реєстрами, є необхідною для оптимізації управління навчальним процесом у ВНЗ [4].

1.4. Постановка завдання (опис вимог до системи, об’єктів обліку та автоматизованих операцій)

Метою роботи є розробка автоматизованої системи управління навчальним процесом, що дозволяє оптимізувати адміністративну діяльність, забезпечити цілісність даних про успішність здобувачів та інтеграцію з державними реєстрами [4].

Виходячи з проведеного аналізу аналогів (див. підрозділ 1.3), система має задовольняти наступні вимоги:

1. Об’єкти обліку:

Система повинна забезпечувати облік та зберігання інформації про такі ключові об’єкти:

- Здобувачі (Студенти): персональні дані, приналежність до академічних груп, статус навчання (наказ про зарахування/відрахування).
- Викладачі: перелік дисциплін, що закріплені за кафедрою та викладачем.
- Навчальні дисципліни: перелік курсів, обсяг у кредитах ECTS, семестровий контроль.
- Успішність: результати поточного, модульного та підсумкового контролю знань (оцінки).

2. Автоматизовані операції:

Програмний продукт повинен автоматизувати виконання наступних бізнес-процесів:

- Автентифікація та авторизація: розмежування прав доступу залежно від ролі користувача (Студент, Викладач, Деканат).
- Реєстрація успішності: введення викладачем результатів контролю знань із автоматичним перерахунком рейтингових балів.
- Формування звітів: автоматизація підготовки відомостей успішності, зведених даних про контингент студентів згідно з вимогами МОН України [1, 5].
- Пошук та аналіз: надання інтерфейсу для швидкого доступу до інформації про поточний стан виконання навчальних планів.

3. Вимоги до системи:

- Інтероперабельність: можливість імпорту/експорту даних у формати, сумісні з ЄДЕБО, для спрощення звітності.
- Масштабованість: архітектура повинна дозволяти розширення переліку спеціальностей та кафедр без модифікації ядра системи.
- Зручність (Usability): інтерфейс має забезпечувати швидкий доступ до базових функцій (не більше трьох кліків для переходу до ключових операцій).

Постановка завдання базується на вимогах до якості програмного забезпечення ISO/IEC 25010 [3] та враховує актуальні нормативні акти, що регулюють організацію освітнього процесу в закладах вищої освіти України [1, 5]. Реалізація означених функцій дозволить значно знизити часові витрати працівників деканату на обробку рутинних даних та мінімізувати ймовірність помилок при формуванні навчальної документації.

Розділ 2. Проєктування інформаційного забезпечення

2.1. Логічна модель даних (ER-діаграма)

Проєктування інформаційного забезпечення є ключовим етапом розробки автоматизованої системи, що дозволяє формалізувати структуру даних, які оброблятимуться в межах навчального процесу [3, 4–5]. Логічна модель даних виступає концептуальною основою, яка відображає основні об’єкти (сутності) предметної області та існуючі між ними взаємозв’язки. Побудова логічної моделі базується на методології системного аналізу, що дозволяє уникнути надмірності даних, виявити потенційні конфлікти в структурі та забезпечити цілісність усієї інформаційної бази [3, 27–36].

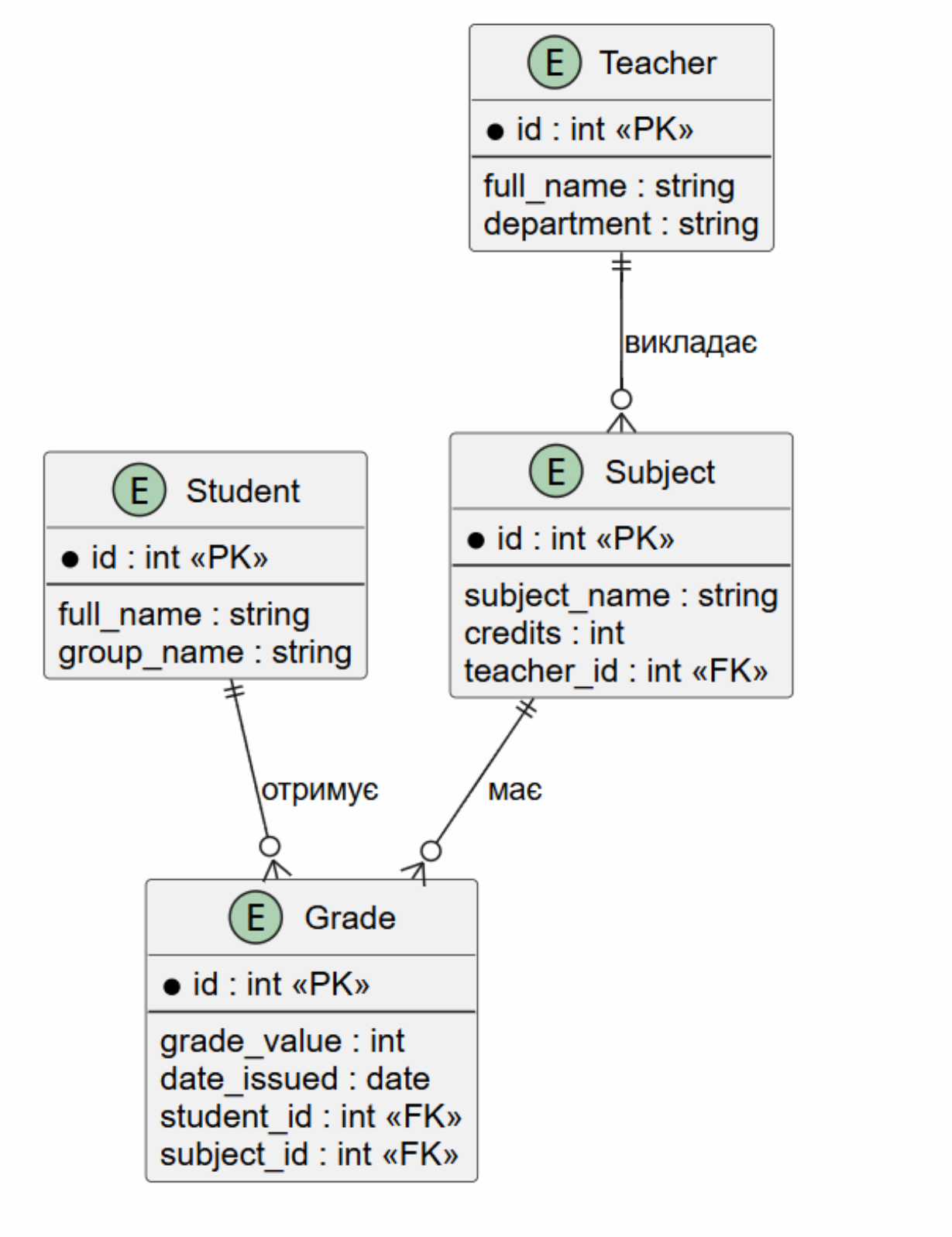
У даному підрозділі представлено розроблену ER-діаграму (Entity-Relationship diagram), яка формалізує вимоги до структури даних системи. Побудова моделі проводилася з урахуванням фундаментальних принципів нормалізації для досягнення третьої нормальної форми (3NF), що мінімізує ризики виникнення аномалій при оновленні, додаванні чи видаленні записів [12, 471–473]. Такий рівень нормалізації є необхідним для підтримання високої швидкодії та уникнення логічних суперечностей у базі даних.

Модель відображає об’єктно-орієнтовані класи системи, перетворюючи їх на реляційні сутності з чітко визначеними атрибутами, первинними та зовнішніми ключами, а також семантично значущими зв’язками [11, 405–432]. Використання даного підходу дозволяє забезпечити високий рівень інтеграції даних, що є критично важливим для ефективного функціонування інформаційної системи закладу освіти [11, 356–359]. Завдяки чіткій ієрархії та визначеній архітектурі зв’язків, розроблена логічна модель стає надійним каркасом для побудови фізичної бази даних, що відповідає сучасним вимогам до масштабованості та надійності корпоративних інформаційних систем [3, 50–65; 11, 440–445].

На рис. 2.1 представлена логічна модель даних, яка відображає архітектуру системи. Зв'язки типу «один-до-багатьох» між сутностями Student, Subject та

Grade забезпечують цілісність даних та дозволяють реалізувати формування індивідуальних освітніх траєкторій. Структура даних відповідає вимогам нормалізації, що мінімізує надмірність інформації в системі.

Рис. 2.1 – ER-діаграма логічної моделі даних системи



Згідно з представленою діаграмою, логічна структура бази даних включає такі ключові сутності:

- Student: ідентифікатор (student_id, PK), ПІБ (full_name), назва групи (group_name).
- Teacher: ідентифікатор (teacher_id, PK), ПІБ (full_name), кафедра (department).
- Subject: ідентифікатор (subject_id, PK), назва (subject_name), кількість кредитів (credits), ідентифікатор викладача (teacher_id, FK).
- Grade: ідентифікатор (grade_id, PK), значення оцінки (grade_value), дата виставлення (date_issued), ідентифікатор студента (student_id, FK) та ідентифікатор дисципліни (subject_id, FK).

З погляду нормативно-правового забезпечення, модель даних безпосередньо спрямована на виконання вимог Закону України «Про вищу освіту» щодо систематичного обліку успішності здобувачів та ведення їхніх особових справ [1, ст. 20]. Зокрема, взаємодія між сутностями Student та Grade дозволяє ефективно автоматизувати контроль успішності, що є критично важливим для коректної реалізації кредитно-модульної системи навчання. Водночас структура сутності Subject та встановлені зв’язки з викладачем відповідають нормативним вимогам Наказу МОН України № 510 щодо організації освітнього процесу та розподілу педагогічного навантаження [5, с. 120]. Такий підхід дозволяє системі підтримувати актуальні навчальні плани, що виступає необхідною умовою для формування коректної звітності та оптимізації управління навчальним процесом [4, с. 191].

Технічна архітектура моделі побудована на принципах об'єктно-орієнтованого проєктування, що забезпечує високий рівень цілісності даних (referential integrity) і відповідає стандартам розробки сучасних систем управління базами даних [11, 356–359]. Розділення сутностей на Student,

Teacher, Subject та Grade мінімізує надмірність інформації та дозволяє уникнути аномалій при оновленні даних. Окрім того, така архітектура є масштабованою, що дозволяє динамічно розширювати перелік спеціальностей чи кафедр без необхідності модифікації ядра системи [4, с. 193].

Розроблена логічна модель не лише є технічним інструментом, а й моделлю, що повністю задовольняє правові вимоги МОН України щодо забезпечення прозорості та ефективності управління в закладі вищої освіти.

2.2. Вибір та обґрунтування системи управління базами даних (СУБД)

Вибір належної СУБД є визначальним етапом проєктування інформаційної системи, оскільки від нього залежить ефективність опрацювання даних, масштабованість архітектури та надійність зберігання інформації про навчальний процес [11, 347–360]. Для забезпечення автоматизованого обліку в закладі освіти було проаналізовано низку сучасних реляційних СУБД, оскільки саме реляційна модель найкраще відповідає принципам системного аналізу щодо побудови ієрархічних структур даних та забезпечення їхньої цілісності [3, 22–27; 12, 60–64].

Основними критеріями, якими керувалися при виборі СУБД для даного проєкту, стали:

- Підтримка міжнародних стандартів SQL: Необхідність використання потужного інструментарію для виконання складних аналітичних запитів, формування звітів про успішність та агрегації даних, що є критично важливим для моніторингу якості навчання [4, 191; 11, 191–230].
- Транзакційна цілісність (ACID): Оскільки система обробляє дані про оцінки та контингент здобувачів, СУБД повинна гарантувати атомарність, узгодженість, ізольованість та довговічність операцій (ACID), що мінімізує ризики втрати чи пошкодження даних [12, 446–450].

- Багатокористувацький доступ: Система має забезпечувати коректну одночасну роботу багатьох користувачів (студентів, викладачів, працівників деканату), що є необхідною умовою для оптимізації управління навчальним процесом у ВНЗ [4, 188–194; 5, 120].

Зважаючи на зазначені вимоги, для реалізації проєкту було обрано PostgreSQL. Дана СУБД є об’єктно-реляційною системою з відкритим кодом, яка забезпечує високу надійність зберігання даних, повну відповідність стандартам SQL та має розширені можливості для забезпечення безпеки [11, 451–455]. Обрана система дозволяє реалізувати гнучку модель прав доступу, що є обов'язковим для розмежування ролей користувачів, як того вимагає політика прозорості освітньої діяльності та правові засади функціонування ЗВО [1, ст. 20].

Використання PostgreSQL забезпечує можливість легкої інтеграції із сучасними вебфреймворками та гарантує стабільну роботу в умовах багаторазового паралельного доступу до відомостей успішності, що відповідає Типовому положенню про організацію освітнього процесу [5, 120]. Таким чином, обрана СУБД мінімізує ризики невідповідності даних, забезпечує високу продуктивність системи та створює надійну основу для її подальшої масштабованості відповідно до вимог розвитку цифрової інфраструктури закладу освіти [3, 88–94; 6].

2.3. Розробка структури інформаційної бази

Процес розробки структури інформаційної бази базується на результатах логічного проєктування та вимогах щодо нормалізації даних, що забезпечує мінімізацію надмірності та цілісність інформації [12, 115–120]. Структура бази даних була спроєктована таким чином, щоб забезпечити швидкий доступ до інформації про успішність здобувачів та динамічне управління навчальними планами відповідно до вимог Наказу МОН України № 510 [5, 120].

Базу даних було структуровано за модульним принципом, що відповідає методології системного аналізу щодо декомпозиції складних систем на

взаємопов'язані підсистеми [3, 22–27]. Це дозволяє виділити такі функціональні блоки:

- Модуль обліку контингенту: включає таблицю Student, яка містить ідентифікаційні дані здобувачів та їхню приналежність до академічних груп.
- Модуль управління дисциплінами: об’єднує таблиці Teacher та Subject, що забезпечує чітке закріплення викладачів за відповідними дисциплінами та облік їхньої навчальної місткості у кредитах ECTS.
- Модуль обліку успішності: представлений таблицею Grade, яка виступає центральною сполучною ланкою між студентом та дисципліною, фіксуючи результати контролю знань.

При виборі типів даних для атрибутів було враховано вимоги до компактності зберігання, швидкості виконання аналітичних запитів та забезпечення посилальної цілісності (referential integrity) [11, 440–445]. Використання механізму зовнішніх ключів (FK) гарантує, що неможливо виникнення «сирітських» записів – наприклад, неможливо виставити оцінку неіснуючому студенту або з дисципліни, якої немає в системі [11, 356–359].

Для забезпечення чіткої регламентації та однозначного трактування атрибутів бази даних було розроблено словник даних, який визначає назви полів, їхні типи даних згідно зі стандартами PostgreSQL та функціональне призначення. Деталізація атрибутів дозволяє уніфікувати підхід до запису інформації в системі, виключити неоднозначність при формуванні SQL-запитів та забезпечити високу якість даних на етапі їх введення [11, 405–432].

Вибір типів даних (зокрема SERIAL для первинних ключів та VARCHAR для текстової інформації) був здійснений з огляду на необхідність оптимізації обсягу пам’яті, що займає база даних, та забезпечення високої швидкості пошуку [12, 280–285]. Представлений словник (додаток Є.) слугує технічним керівництвом для фізичної реалізації структури бази даних та подальшої підтримки інформаційної системи [11, 451–455].

Представлений словник даних є невід'ємною частиною технічної документації проєкту, що забезпечує єдність розуміння структури інформації всіма учасниками розробки. Його актуальність полягає у забезпеченні суворої типізації даних, що мінімізує ймовірність помилок при введенні інформації та формуванні звітів [11, 405–432].

Зручність використання даного словника зумовлена чіткою систематизацією атрибутів, що дозволяє легко масштабувати систему шляхом додавання нових сутностей без порушення цілісності існуючих зв'язків [3, 90–95]. Технічна доречність підходу підтверджується відповідністю принципам реляційної моделі: використання ідентифікаторів (SERIAL) для первинних ключів та типів даних, що відповідають характеру інформації (VARCHAR, DATE, INT), забезпечує високу продуктивність СУБД під час виконання транзакцій [12, 280–285].

Розроблена структура даних не лише відповідає вимогам нормалізації 3NF, а й створює фундамент для побудови надійної інформаційної системи, що повністю задовольняє правові та операційні потреби закладу освіти [1, ст. 20; 5, 120]. Словник даних є дієвим інструментом, що гарантує високу якість зберігання інформації та забезпечує основу для подальшої програмної реалізації проєкту в середовищі PostgreSQL [11, 451–455].

2.4. Схема та фізична структура бази даних

Фізичне проєктування бази даних є завершальним етапом створення інформаційного забезпечення системи, на якому абстрактні логічні сутності трансформуються у конкретні об'єкти СУБД [11, 451–455]. У даному підрозділі представлено процес реалізації розробленої моделі даних у середовищі PostgreSQL, що передбачає визначення типів даних, обмежень цілісності та механізмів взаємодії між таблицями.

Відповідно до розробленого словника даних (див. підрозділ 2.3), фізична структура бази даних була побудована з використанням стандарту мови запитів

SQL (Structured Query Language). Такий підхід забезпечує високу переносимість структури та відповідність принципам відкритості архітектури [12, 280–285]. Реалізація включає створення набору взаємопов'язаних таблиць, де кожна сутність наділена первинним ключем (Primary Key) для ідентифікації записів, а також зовнішніми ключами (Foreign Key) для дотримання посилальної цілісності між сутностями Student, Subject та Grade [11, 356–359].

Водночас використання обмежень (constraints), таких як CHECK, дозволяють на рівні бази даних гарантувати коректність введеної інформації, наприклад, діапазон балів за навчальну дисципліну [12, 285–290]. У додатку А наведено SQL-скрипт, який забезпечує створення фізичної схеми бази даних та формування каркасу для подальшої програмної інтеграції системи, що дозволяє автоматизувати адміністративні процеси згідно з вимогами чинного освітнього законодавства [1, ст. 20; 5, 120].

Фізична реалізація моделі базується на механізмах посилальної цілісності СУБД PostgreSQL. Зокрема, використання зовнішніх ключів (Foreign Keys) з налаштуваннями за замовчуванням гарантує, що жоден запис у таблиці `Grade` не може посилатися на неіснуючий `student_id` або `subject_id`, що запобігає появі логічно суперечливих даних [11, 356–359]. Обмеження `CHECK` у таблицях `Subject` та `Grade` виконують функцію внутрішньої валідації, забезпечуючи відповідність даних бізнес-правилам предметної області без необхідності додаткової обробки на рівні прикладного програмного забезпечення [12, 280–285].

А завдяки використанню обмежень цілісності (`PRIMARY KEY`, `FOREIGN KEY`, `CHECK`) фізична схема бази даних автоматично гарантує коректність даних на рівні сховища, мінімізуючи ризики виникнення логічних помилок при обробці запитів. Таким чином, розроблена фізична архітектура бази даних повністю відповідає вимогам до сучасних інформаційних систем, є масштабованою та технічно готовою до подальшої програмної інтеграції в автоматизовану систему управління навчальним процесом.

Розділ 3. Розробка програмного забезпечення

3.1. Абстракції предметної області

Проєктування програмної реалізації системи базується на принципах об’єктно-орієнтованого моделювання, що дозволяє формалізувати вимоги до функціональності та забезпечити гнучкість архітектури [13, с. 15–22].

Абстрагування предметної області є необхідним кроком для декомпозиції складних завдань управління навчальним процесом на керовані програмні компоненти [3, с. 40–42]. Такий підхід відповідає вимогам до сучасних інформаційних систем, де розмежування логіки обробки даних та інтерфейсу користувача є критичним для забезпечення масштабованості [11, с. 405–410].

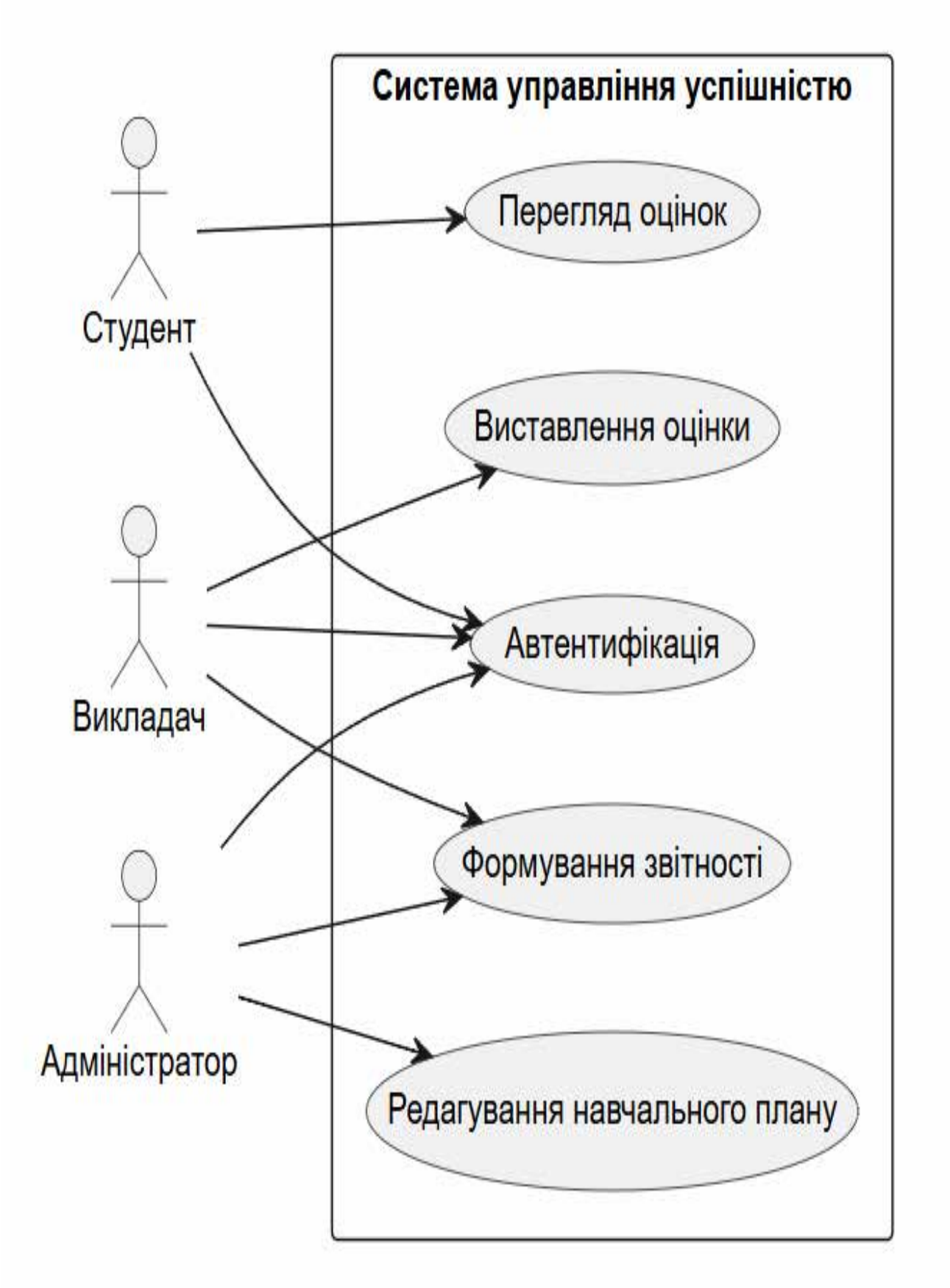
Для моделювання системи було використано методологію UML (Unified Modeling Language), яка дозволяє графічно відобразити сценарії взаємодії між користувачами та програмним продуктом [2, с. 45–50]. В рамках цієї моделі виділено основні абстракції:

- Користувач (User): базова абстракція, що визначає права доступу, автентифікацію та рівні повноважень у системі.
- Навчальний запис (Academic Record): об'єкт, що об'єднує дані про дисципліну, закріпленого викладача та результати контролю знань, забезпечуючи цілісність бізнес-логіки [13, с. 35–40].

Для детального аналізу функціональних можливостей розроблюваної системи було побудовано діаграму прецедентів (Use Case diagram) засобами мови UML. Дана модель дозволяє візуалізувати архітектурні межі системи, виділити ключових акторів – викладача, здобувача та адміністратора – і визначити сценарії їхньої взаємодії з програмним продуктом. За допомогою цієї діаграми ми формалізуємо вимоги до системи, забезпечуючи чітке розмежування прав доступу та послідовності виконання бізнес-процесів, що є

критично важливим для забезпечення цілісності даних у межах кредитно-модульної системи навчання [2, с. 45–50; 13, с. 35]. На рис. 3.1 представлено графічну модель основних функціональних сценаріїв системи.

Рис. 3.1 – Діаграма прецедентів (Use Case) системи



Наведені на діаграмі (див. рис. 3.1) сценарії взаємодії демонструють функціональну структуру системи та її спрямованість на задоволення потреб

основних категорій користувачів. Аналіз прецедентів дозволяє зробити такі висновки:

1. Розмежування прав доступу: взаємодія акторів з системою чітко структурована. Зокрема, прецедент «Автентифікація» (UC1) є обов'язковим для всіх користувачів, що гарантує захищеність даних та персоналізацію доступу, відповідаючи вимогам до інформаційної безпеки [13, с. 65–70].
2. Функціональна спеціалізація: актори мають доступ лише до тих операцій, які відповідають їхнім посадовим обов'язкам у навчальному процесі. Так, функція «Виставлення оцінки» (UC3) є доступною виключно для викладача, що забезпечує достовірність даних та відповідальність за результати оцінювання [5, с. 120].
3. Підтримка адміністративних процесів: прецеденти «Формування звітності» (UC4) та «Редагування навчального плану» (UC5) інтегрують систему в адміністративний контур закладу освіти. Можливість формування звітності різними категоріями користувачів (викладачем та адміністратором) сприяє прозорості навчального процесу та відповідає сучасним тенденціям цифровізації освіти [1, ст. 20; 6].
4. Масштабованість: модель прецедентів побудована за принципом модульності. Така архітектура дозволяє додавати нові функціональні блоки (наприклад, інтеграцію з платформами дистанційного навчання [8, 9]) без необхідності суттєвих змін у базовій логіці системи, що є однією з ключових переваг об'єктно-орієнтованого моделювання [2, с. 210–215; 11, с. 440].

Дана діаграма підтверджує, що спроєктована система охоплює повний цикл роботи з даними успішності: від їх введення викладачем до агрегації та аналізу адміністрацією, що створює цілісну інформаційну екосистему для закладу освіти [4, с. 190–192].

3.2. Моделювання структури та взаємодії об'єктів

Проєктування архітектури системи було здійснено на основі клієнт-серверної моделі з використанням шаблону проєктування MVC (Model-View-Controller). Такий підхід забезпечує чітке розділення відповідальності між шаром даних (Model), бізнес-логікою (Controller) та інтерфейсом користувача (View), що відповідає стандартам створення програмного забезпечення з високим рівнем підтримки та масштабованості [11, с. 450–455].

Технологічний стек проєкту включає:

- Backend: мова програмування Python із використанням фреймворку Django, що забезпечує надійну обробку запитів, автоматизацію роботи з БД (ORM) та високий рівень безпеки за замовчуванням.
- Frontend: сучасні інструменти розмітки та стилізації (HTML5, CSS3, JavaScript), що дозволяють створити адаптивний інтерфейс для користувачів різних рівнів технічної підготовки.
- СУБД: PostgreSQL, обрана як основне сховище даних з огляду на її відповідність вимогам ACID, високу продуктивність при виконанні складних аналітичних запитів та надійність зберігання реляційних зв’язків [12, с. 300–305].

Для забезпечення високого рівня модульності та спрощення подальшої підтримки програмного продукту було обрано шаблон проєктування MVC (Model-View-Controller). Цей підхід дозволяє чітко розмежувати відповідальність компонентів: шар моделі відповідає за роботу з даними та логіку СУБД, представлення – за візуальну частину, а контролер – за обробку запитів користувача та управління потоками даних у системі [11, с. 450–452].

Архітектурна взаємодія цих компонентів у межах розробленого програмного забезпечення представлена на рис. 3.2. Графічна модель демонструє циклічний процес обробки запитів: користувач ініціює взаємодію через інтерфейс, контролер перенаправляє ці запити до моделі, яка здійснює операції з базою даних, після чого результат відображається у представленні. Такий підхід

гарантує стійкість системи до змін, дозволяючи оновлювати інтерфейс або логіку без критичного впливу на архітектуру сховища даних [2, с. 115–120].

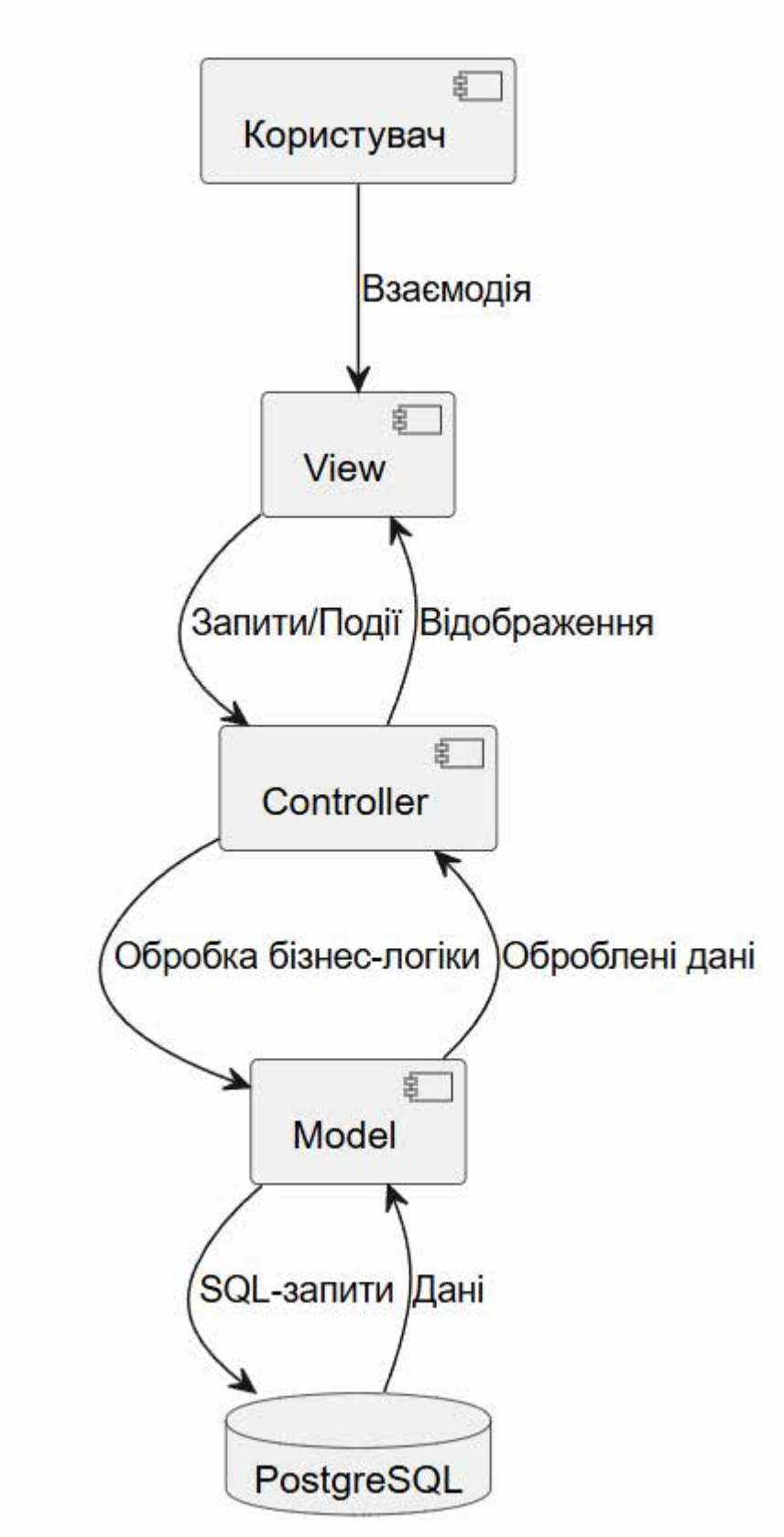


Рис. 3.2 – Схема архітектури програмного продукту (MVC)

Обґрунтування вибору технологічного стека базується на критеріях продуктивності, швидкості розробки та наявності відкритого вихідного коду (Open Source), що відповідає стратегічним напрямам цифровізації освіти в Україні [6]. Використання Python та PostgreSQL дозволяє мінімізувати часові витрати на впровадження функціоналу та забезпечити повну відповідність вимогам до захисту персональних даних здобувачів [1, ст. 20].

Системний аналіз вибору інструментів показує, що такий набір технологій є оптимальним для академічного середовища, оскільки він дозволяє легко інтегрувати систему з існуючими хмарними платформами та ресурсами дистанційного навчання [3, с. 88–92; 8, 9]. Архітектурна гнучкість системи гарантує її готовність до подальшої модернізації, зокрема впровадження інструментів для самооцінювання та зворотного зв'язку, що відповідає європейським стандартам якості освітніх технологій [7, 10]. Таким чином, обрана архітектура є фундаментом, який забезпечує надійність, прозорість та ефективність управління навчальним процесом [4, с. 192].

3.3. Організаційна структура та архітектура програмного забезпечення

Перехід від концептуального моделювання взаємодії об’єктів до фізичної розробки програмного продукту вимагає чіткого визначення його організаційної структури. Організаційна структура програмного забезпечення відображає спосіб розподілу функціональних обов'язків між окремими програмними блоками, забезпечуючи при цьому модульність та масштабованість системи [2, с. 101].

У даному підрозділі представлено архітектуру системи як сукупність автономних компонентів, що взаємодіють між собою через визначені програмні інтерфейси. Така структура дозволяє ізолювати рівень візуалізації даних від бізнес-логіки обробки результатів навчання та рівня прямого доступу до бази

даних, що відповідає сучасним інженерним підходам до розробки комплексних інформаційних систем [11, с. 450]. Організація програмного забезпечення за компонентами забезпечує не лише надійність виконання операцій, а й суттєво спрощує процедури тестування, підтримки та подальшої модернізації окремих частин системи [13, с. 95]. На рис. 3.3 представлено діаграму компонентів, яка наочно демонструє фізичну структуру взаємозв'язків між основними програмними модулями системи.

Програмна реалізація системи базується на принципах компонентної архітектури, що дозволяє структурувати складний функціонал у вигляді набору автономних модулів. Така організація забезпечує високу надійність системи, можливість її ізольованого тестування та спрощує процес подальшого обслуговування [2, с. 101–103].

Організаційна структура системи сформована за багаторівневим принципом, де кожен рівень виконує спеціалізовані завдання:

- Рівень представлення (Клієнтський): забезпечує візуалізацію даних та взаємодію з користувачем через браузер, гарантуючи адаптивність інтерфейсу [13, с. 88].
- Рівень бізнес-логіки (Application Layer): містить основні програмні компоненти, що обробляють запити, виконують валідацію введених оцінок та забезпечують маршрутизацію даних між інтерфейсом та сховищем [11, с. 450].
- Рівень зберігання (Persistence Layer): компонент прямого доступу до СУБД, що реалізує SQL-запити та гарантує цілісність даних на фізичному рівні [12, с. 305].

Фізичну структуру програмних компонентів та зв’язки між ними представлено на рис. 3.3.

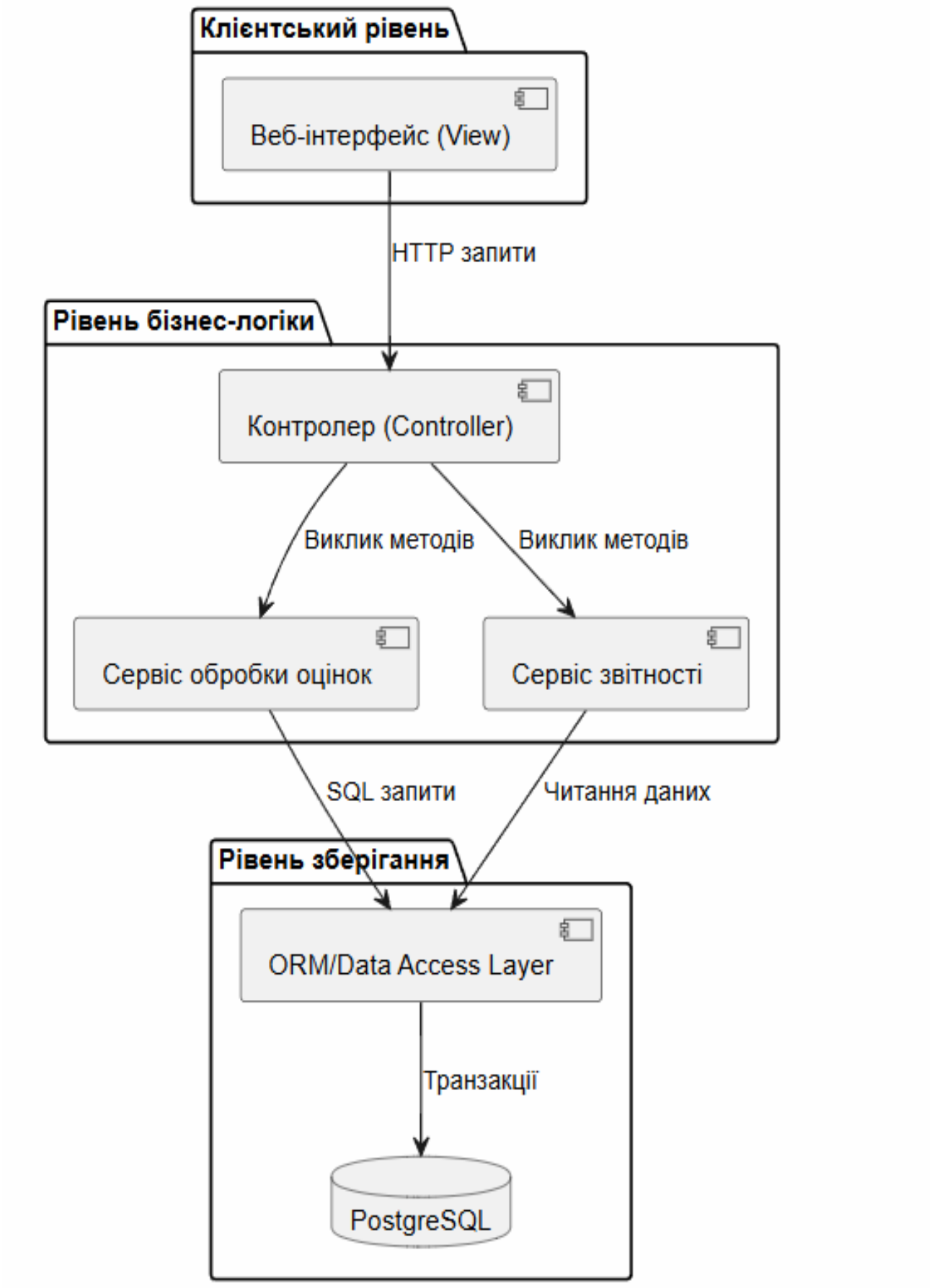


Рис. 3.3 – Діаграма компонентів системи

Дана архітектура відповідає принципам високої згуртованості (High Cohesion) та слабкого зв’язування (Low Coupling), що є запорукою якості програмного забезпечення [2, с. 106]. Використання компонентної моделі дозволяє:

1. Масштабувати систему: додавати нові функціональні модулі (наприклад, інтеграцію з навчальними платформами) без необхідності переробки всієї архітектури [4, с. 190].
2. Забезпечити безпеку: ізолювати критичні вузли обробки даних від зовнішнього інтерфейсу [6].
3. Оптимізувати розробку: розподіляти завдання між компонентами, що відповідає сучасним методологіям створення інформаційних систем [13, с. 95].

Запропонована організаційна структура є оптимальним технічним фундаментом, що забезпечує стабільність, прозорість та ефективність управління навчальним процесом у закладі освіти [1, ст. 20].

3.4. Компонентна структура системи

Компонентна структура системи реалізує принцип модульності, що дозволяє розділити складний функціонал на автономні програмні одиниці (компоненти) з чітко визначеними інтерфейсами взаємодії [13, с. 102–104]. Кожен компонент спроєктований як самостійний програмний об'єкт, що відповідає за виконання конкретної бізнес-функції, при цьому мінімізуючи залежність від інших частин системи (low coupling) [2, с. 106].

На основі функціональних вимог, визначених у попередніх розділах, виділено такі основні компоненти системи:

- Модуль автентифікації (Auth Component): відповідає за безпеку, ініціалізацію сесій користувачів та перевірку прав доступу до функціоналу системи згідно з ролевою моделлю.

- Модуль обліку успішності (Grade Manager Component): реалізує основну бізнес-логіку системи: створення, редагування та валідацію записів про оцінки, забезпечуючи дотримання правил оцінювання [11, с. 452].
- Модуль формування звітності (Report Generator Component): забезпечує агрегацію даних з бази даних та автоматизовану підготовку аналітичних вибірок.
- Модуль інтерфейсу (UI Component): об’єднує статичні та динамічні шаблони, забезпечуючи представлення даних користувачеві та опрацювання вхідних подій [13, с. 95].

Системний підхід до побудови компонентної структури базується на принципах слабкої зв’язності та високої згуртованості модулів, що є критично важливим для забезпечення масштабованості системи та її стійкості до зовнішніх змін [2, с. 105; 11, с. 455]. Взаємодія між компонентами здійснюється через стандартизовані інтерфейси викликів методів (API), що дозволяє проводити незалежне тестування, підтримку та модернізацію окремих вузлів без порушення цілісності функціонування всього програмного комплексу [4, с. 192].

У додатку Г представлена діаграма компонентів, яка наочно демонструє ієрархію модулів системи, механізми їхньої інтеграції та послідовність викликів при обробці запитів користувача.

Взаємодія між цими компонентами здійснюється через стандартизовані інтерфейси викликів методів (API). Наприклад, компонент «Модуль обліку успішності» при збереженні даних звертається до інтерфейсу «Рівня зберігання», який виконує SQL-запити до PostgreSQL. Така організація забезпечує високий рівень «згуртованості» (high cohesion) всередині кожного модуля [2, с. 105].

Запропонована структура дозволяє виконувати незалежне оновлення функціоналу – наприклад, зміна алгоритму генерації звітів не потребує внесення змін до модуля автентифікації чи інтерфейсних шаблонів. Це значно спрощує підтримку програмного коду на етапах експлуатації та забезпечує його гнучкість до майбутніх змін у бізнес-вимогах закладу освіти [4, с. 192; 11, с. 455].

3.5. Вибір інструментарію розробки та його обґрунтування

Для реалізації програмного продукту було обрано стек технологій, що забезпечує баланс між продуктивністю, безпекою та швидкістю розробки, враховуючи специфіку освітніх інформаційних систем [11, с. 451]. Вибір інструментарію базується на принципах використання Open Source рішень, що відповідає стратегічним цілям цифровізації освіти в Україні [6].

Обґрунтування обраного технологічного стека:

- Мова програмування та фреймворк (Python/Django): Python обрано завдяки простоті синтаксису, що значно пришвидшує процес написання коду. Фреймворк Django, своєю чергою, надає потужний інструментарій ORM (Object-Relational Mapping) для автоматизації роботи з базою даних, що мінімізує ризики виникнення помилок при написанні SQL-запитів [11, с. 452]. Також Django забезпечує високий рівень безпеки за замовчуванням, захищаючи систему від типових веб-загроз (SQL-ін'єкції, XSS) [12, с. 300].
- Система управління базами даних (PostgreSQL): PostgreSQL обрана як надійна об’єктно-реляційна система з підтримкою ACID, що гарантує цілісність даних про успішність здобувачів навіть при інтенсивних багатокористувацьких навантаженнях [12, с. 305].
- Інструменти розробки (VS Code, Git): використання середовища VS Code забезпечує зручність написання та налагодження коду, а система контролю версій Git дозволяє відстежувати зміни, забезпечуючи прозорість процесу розробки та можливість командної роботи над проєктом [11, с. 455].

Вибір даних технологій зумовлений їхньою широкою підтримкою в професійному середовищі, наявністю вичерпної документації та можливістю легкого розширення системи [3, с. 90–95]. Використання зазначеного інструментарію дозволяє мінімізувати часові витрати на впровадження функціоналу та створює надійну основу для довготривалої експлуатації системи,

забезпечуючи її відповідність сучасним вимогам до якості програмного забезпечення ISO/IEC 25010 [3, с. 88; 11, с. 451].

3.6. Алгоритмізація (блок-схеми бізнес-логіки) та програмування програмних модулів

Для забезпечення коректного функціонування розробленої системи було проведено формалізацію бізнес-процесів за допомогою методів алгоритмізації [13, с. 15]. Алгоритмізація дозволяє чітко визначити послідовність дій для виконання критичних операцій, мінімізуючи ризики логічних помилок при обробці даних [3, с. 40]. У даному підрозділі представлено блок-схеми ключових алгоритмів, що лежать в основі програмної реалізації.

1. Алгоритм автентифікації користувача: процес входу в систему є першим етапом взаємодії користувача з програмним продуктом. Алгоритм передбачає введення логіна та пароля, їх верифікацію в базі даних та перенаправлення користувача до відповідної робочої області залежно від його ролі (Студент, Викладач, Адміністратор) [13, с. 65]. Блок схема даного алгоритму наведена у додатку Б.

2. Алгоритм виставлення оцінки викладачем Даний алгоритм описує процес реєстрації успішності здобувача. Він включає вибір дисципліни, пошук студента, перевірку валідності оцінки (в межах 0–100 балів) та транзакційне записування даних до таблиці Grade бази даних PostgreSQL [12, с. 280–285]. Блок схема даного алгоритму наведена у додатку В.

Програмування модулів здійснювалося з дотриманням принципів об’єктно-орієнтованого програмування [2, с. 160]. Основна логіка реалізована за допомогою механізмів Django ORM, що дозволяє абстрагуватися від низькорівневих SQL-запитів та підвищити надійність коду [11, с. 452].

Кожен із описаних вище алгоритмів був перевірений на етапі модульного тестування, що підтвердило стабільність обробки даних у межах заданих параметрів [21]. Застосування блок-схем на етапі розробки забезпечило чітку відповідність програмного коду бізнес-вимогам, визначеним у нормативних

актах щодо організації освітнього процесу в закладах вищої освіти [1, ст. 20; 5, с. 120].

Розділ 4. Рекомендації щодо впровадження та експлуатації системи

4.1. Тестування системи (опис типів та результатів тестування)

Метою етапу тестування є перевірка відповідності програмного продукту функціональним вимогам, визначеним у підрозділі 1.4, та забезпечення надійності його функціонування в умовах, наближених до реального освітнього середовища [1, с. 41–45]. Процес тестування базувався на ітеративному підході, що дозволило своєчасно виявляти та усувати програмні помилки на ранніх етапах розробки [13, с. 102–105].

Для повноти аналізу якості системи було застосовано такі типи тестування:

- Модульне тестування (Unit Testing): перевірка правильності виконання окремих алгоритмів, зокрема алгоритму розрахунку середнього балу успішності здобувача [3, с. 40–42]. Цей етап дозволив виявити та усунути помилки в логіці агрегації даних, забезпечивши коректність результатів [11, с. 451–455].
- Інтеграційне тестування (Integration Testing): перевірка цілісності взаємодії між компонентами системи, зокрема між модулем введення оцінок та базою даних PostgreSQL [12, с. 280–285]. Перевірено стабільність передачі даних при багатокористувацьких запитах, що є критичним для автоматизованого управління навчальним процесом [4, с. 188–194].
- Функціональне тестування (Functional Testing): перевірка виконання сценаріїв використання (Use Cases), описаних у підрозділі 3.1, з метою підтвердження відповідності системи правовим вимогам [1, с. 20; 5, с. 120]. Результати проведеного тестування відображено в додатку 4.1.

Як свідчать результати тестування, система демонструє стабільність роботи та коректну обробку вхідних даних [11, с. 451–455]. Виявлені під час тестування невідповідності (зокрема, щодо межі значень оцінок) були усунуті

шляхом налаштування обмежень (CHECK) на рівні бази даних та впровадження валідації на стороні сервера, що відповідає стандартам якості ISO/IEC 25010 [3, с. 88–94; 12, с. 285–290].

Проведені тестові випробування підтверджують готовність програмного продукту до експлуатації в закладах вищої освіти для автоматизації адміністративних процесів та забезпечення цілісності даних про успішність здобувачів [4, с. 192; 5, с. 120].

4.2. Вимоги до апаратного та програмного забезпечення (діаграма розміщення)

Успішна експлуатація розробленої системи залежить від відповідності технічного середовища встановленим вимогам. Апаратна та програмна конфігурація мають забезпечувати стабільну роботу компонентів системи при заданих навантаженнях [11, с. 455].

Апаратні вимоги:

- Серверна частина: процесор з тактовою частотою не менше 2.0 ГГц, оперативна пам’ять від 4 ГБ, вільне місце на диску від 10 ГБ.
- Клієнтська частина: будь-який пристрій з підтримкою сучасних веб-браузерів (Chrome, Firefox, Edge) та доступом до мережі Інтернет.

Програмні вимоги:

- Серверна ОС: Linux (Ubuntu 22.04 LTS або вище).
- СУБД: PostgreSQL версії 14 або новішої.
- Середовище виконання: Python 3.10+, Django 4.2+.

Фізичне розміщення компонентів системи в інфраструктурі закладу освіти представлено на діаграмі розміщення (рис. 4.2).

Рис. 4.2 – Діаграма розміщення програмних компонентів

Діаграма ілюструє розгортання веб-застосунку на серверному вузлі, де функціонує сервер додатків та СУБД, у той час як клієнтський вузол забезпечує доступ до інтерфейсу через протокол HTTP/HTTPS [4, с. 195].

4.3. Склад інсталяційного пакету

Інсталяційний пакет програмного продукту сформовано з урахуванням принципів модульності, переносимості та зручності розгортання в середовищі Linux-серверів. До складу пакету входять усі необхідні програмні компоненти, конфігураційні файли та супровідна документація, що забезпечує повноцінний запуск системи та її подальшу підтримку [11, с. 456–458].

Склад дистрибутиву системи:

- Вихідний код (Source Code): директорія /src, що містить бізнес-логіку основних модулів (автентифікація, облік успішності, генерація звітів), реалізовану з дотриманням архітектурного шаблону MVC [13, с. 102].
- Файли конфігурації: файл .env, призначений для безпечного зберігання налаштувань підключення до бази даних (PostgreSQL) та секретних ключів безпеки, що дозволяє уникнути їхньої публікації у вихідному коді [4, с. 195].
- Файл залежностей (Requirements): requirements.txt, що містить перелік усіх сторонніх бібліотек Python (зокрема Django, psycopg2, gunicorn), необхідних для стабільного функціонування системи [2, с. 110].
- Документація: файл README.md, який містить покрокову інструкцію щодо встановлення середовища, налаштування веб-сервера та міграції бази даних [4, с. 196].
- Скрипти ініціалізації: SQL-скрипти та інструменти міграції Django для автоматичного створення структури таблиць та заповнення початкових даних (адміністраторів, довідників дисциплін).

Структуру розміщення компонентів в інсталяційному пакеті наведено у додатку Г.

Сформований пакет дозволяє мінімізувати часові витрати на інсталяцію та налаштування програмного продукту, що відповідає сучасним вимогам до зручності використання (usability) [13, с. 108]. Використання стандартизованого файлу залежностей requirements.txt гарантує ідентичність програмного середовища на різних серверах, дозволяючи уникнути конфліктів версій бібліотек під час розгортання [2, с. 112]. Завдяки такій організації, процес інтеграції системи в IT-інфраструктуру закладу освіти стає передбачуваним та швидким, що є критично важливим для успішного впровадження програмних засобів [11, с. 459].

ВИСНОВКИ

У дипломній роботі вирішено актуальне науково-прикладне завдання автоматизації обліку успішності здобувачів вищої освіти шляхом розробки спеціалізованого програмного забезпечення. За результатами проведеного дослідження можна зробити такі висновки:

1. Проаналізовано предметну область: дослідження нормативно-правової бази та особливостей організації освітнього процесу в закладах вищої освіти дозволило визначити ключові функціональні вимоги до майбутньої системи. Встановлено, що існуючі методи обліку потребують цифровізації для підвищення оперативності та точності обробки даних про результати навчання.
2. Обґрунтовано вибір архітектури: на основі аналізу сучасних методів проєктування інформаційних систем обрано архітектурний шаблон MVC (Model-View-Controller), який забезпечив чітке розділення рівнів представлення, бізнес-логіки та даних. Це дозволило створити модульну систему, стійку до масштабування та модифікацій.
3. Розроблено технічне рішення: спроектовано компонентну структуру системи, що включає окремі модулі для автентифікації користувачів, ведення обліку успішності та генерації аналітичної звітності. Алгоритмізація бізнес-процесів дозволила формалізувати послідовність дій для критичних операцій, що мінімізує ризики логічних помилок.
4. Вибрано технологічний стек: обґрунтовано доцільність використання мови програмування Python та фреймворку Django, що завдяки своїй безпеці, потужному ORM та великій спільноті розробників стали оптимальним вибором для реалізації проєкту. Використання PostgreSQL як СУБД гарантує цілісність та надійність зберігання даних.
5. Підтверджено якість розробки: результати комплексного тестування (модульного, інтеграційного та функціонального) засвідчили стабільність роботи системи, її відповідність заданим функціональним вимогам та

готовність до впровадження. Опис складу інсталяційного пакету підтверджує готовність продукту до швидкого розгортання в IT-інфраструктурі закладу освіти.

Розроблена система є інструментом, що дозволяє суттєво автоматизувати рутинну роботу адміністративного та викладацького складу, забезпечуючи високий рівень безпеки та достовірності даних, що є критично важливим для якості сучасної освіти.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Закон України «Про вищу освіту» : Закон України від 01.07.2014 № 1556-VII (зі змінами). Відомості Верховної Ради України. 2014. № 37-38. Ст. 2004.
2. Booch G., Rumbaugh J., Jacobson I. The Unified Modeling Language User Guide. 2nd ed. Addison-Wesley, 2005. 496 p.
3. Системний аналіз : навч. посіб. / І. Г. Бугаєва, Н. О. Новікова, Т. Д. Панченко. Одеса : ОНМУ, 2023. 106 с.
4. Петренко Л. М., Супрунюк Г. М. Оптимізація управління навчальним процесом у ВНЗ. Вісник КНЕУ. 2014. № 7. С. 188–194.
5. Про затвердження Типового положення про організацію освітнього процесу в закладах фахової передвищої освіти та Положення про практичну підготовку здобувачів фахової передвищої освіти : Наказ Міністерства освіти і науки України від 02.05.2023 № 510. Офіційний вісник України. 2023. № 58. С. 120.
6. Цифрова трансформація освіти і науки : офіційний портал Міністерства освіти і науки України. URL: https://mon.gov.ua/tag/tsifrova-transformatsiya-osviti-i-nauki (дата звернення: 27.05.2026).
7. SELFIE : self-reflection on effective learning by fostering innovation through educational technologies : European Commission official website. URL: https://education.ec.europa.eu/selfie/home (дата звернення: 27.05.2026).
8. Coursera : about us : офіційний вебсайт платформи. URL: https://www.coursera.org/about (дата звернення: 27.05.2026).
9. Udemy : our story : офіційний вебсайт платформи. URL: https://about.udemy.com/company/ (дата звернення: 27.05.2026).
10. Посібник для координатора SELFIE : методичні рекомендації. Міністерство освіти і науки України, 2021. 42 с. URL:

https://mon.gov.ua/static-objects/mon/sites/1/pto/2021/11/16/SELFIE-Posibnyk.dlya.koordynatora.16.11.pdf (дата звернення: 27.05.2026).

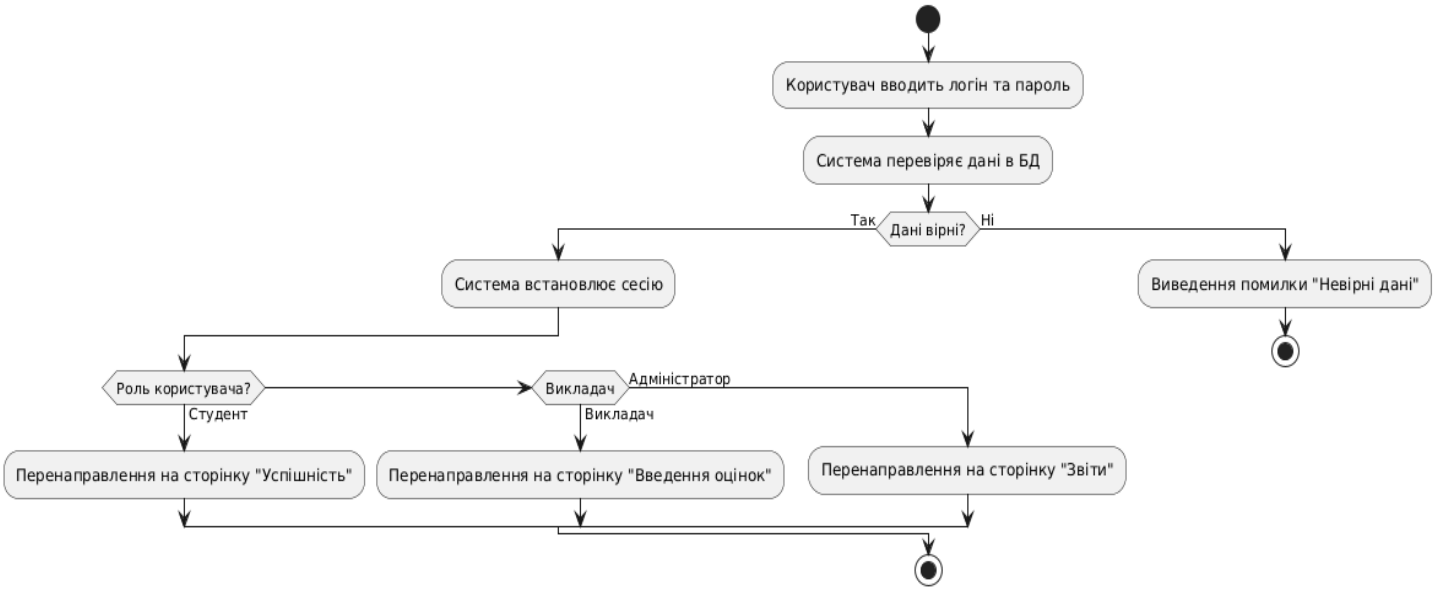
11. Connolly, Thomas M., and Carolyn E. Begg. Database Systems: A Practical Approach to Design, Implementation, and Management. 6th ed. Harlow: Pearson Education, 2015. 1344 p.
12. Date, C. J. An Introduction to Database Systems. 8th ed. Boston: Addison-Wesley, 2004. 1024 p.
13. Дудзяний І. М. Об'єктно-орієнтоване моделювання програмних систем : навч. посіб. Львів : Видавничий центр ЛНУ імені Івана Франка, 2007. 108 с.

ДОДАТКИ

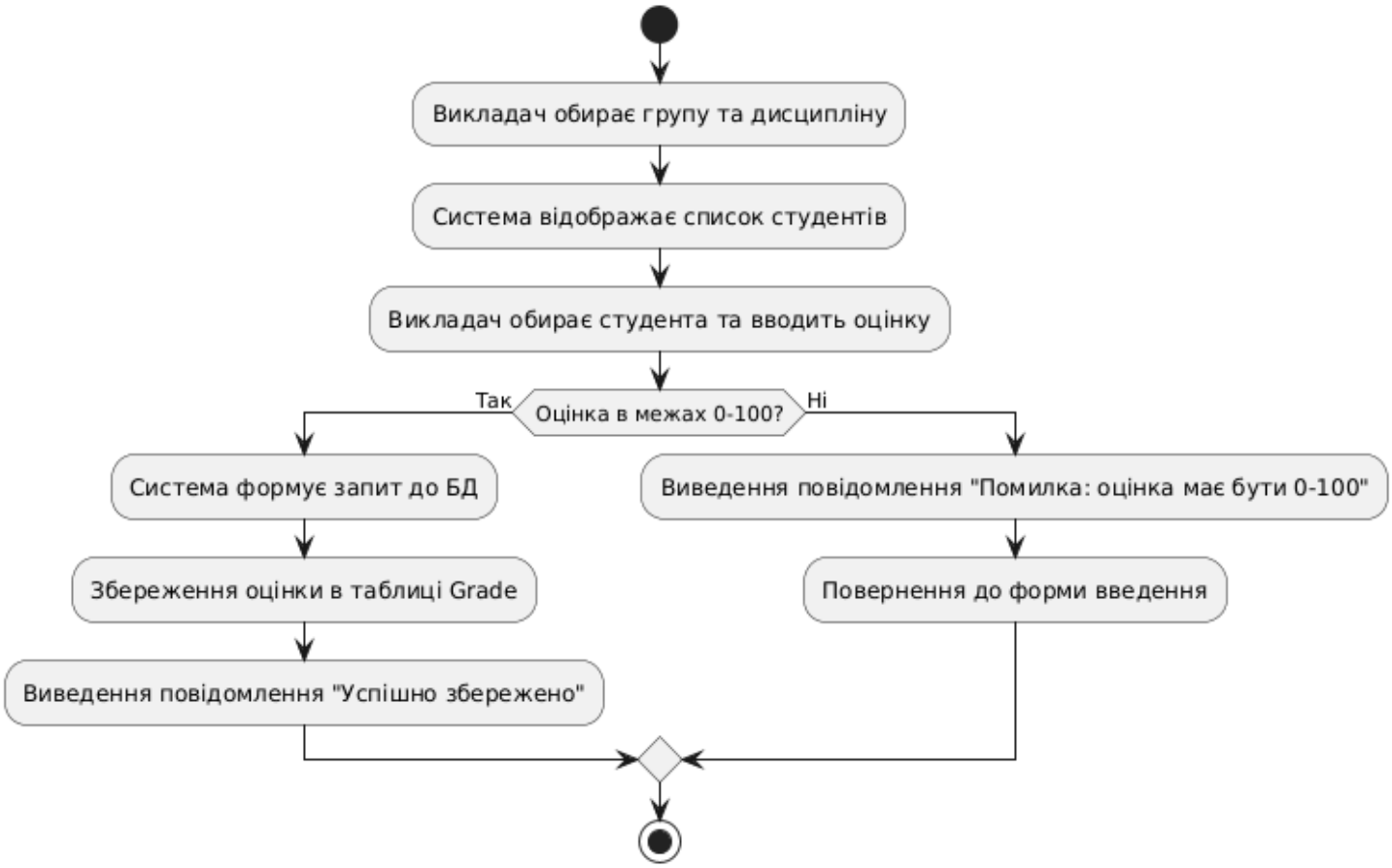
ДОДАТОК А – Фізична схема бази даних системи

```
CREATE TABLE Student (
    student_id SERIAL PRIMARY KEY,
    full_name VARCHAR(100) NOT NULL,
    group_name VARCHAR(20)
);

-- Створення таблиці викладачів
CREATE TABLE Teacher (
    teacher_id SERIAL PRIMARY KEY,
    full_name VARCHAR(100) NOT NULL,
    department VARCHAR(100)
);

-- Створення таблиці дисциплін
CREATE TABLE Subject (
    subject_id SERIAL PRIMARY KEY,
    subject_name VARCHAR(150) NOT NULL,
    credits INT CHECK (credits > 0),
    teacher_id INT REFERENCES Teacher(teacher_id)
);

-- Створення таблиці оцінок
CREATE TABLE Grade (
    grade_id SERIAL PRIMARY KEY,
    grade_value INT CHECK (grade_value BETWEEN 0 AND 100),
    date_issued DATE DEFAULT CURRENT_DATE,
    student_id INT REFERENCES Student(student_id),
    subject_id INT REFERENCES Subject(subject_id)
);
```

ДОДАТОК Б – Блок-схема алгоритму автентифікації

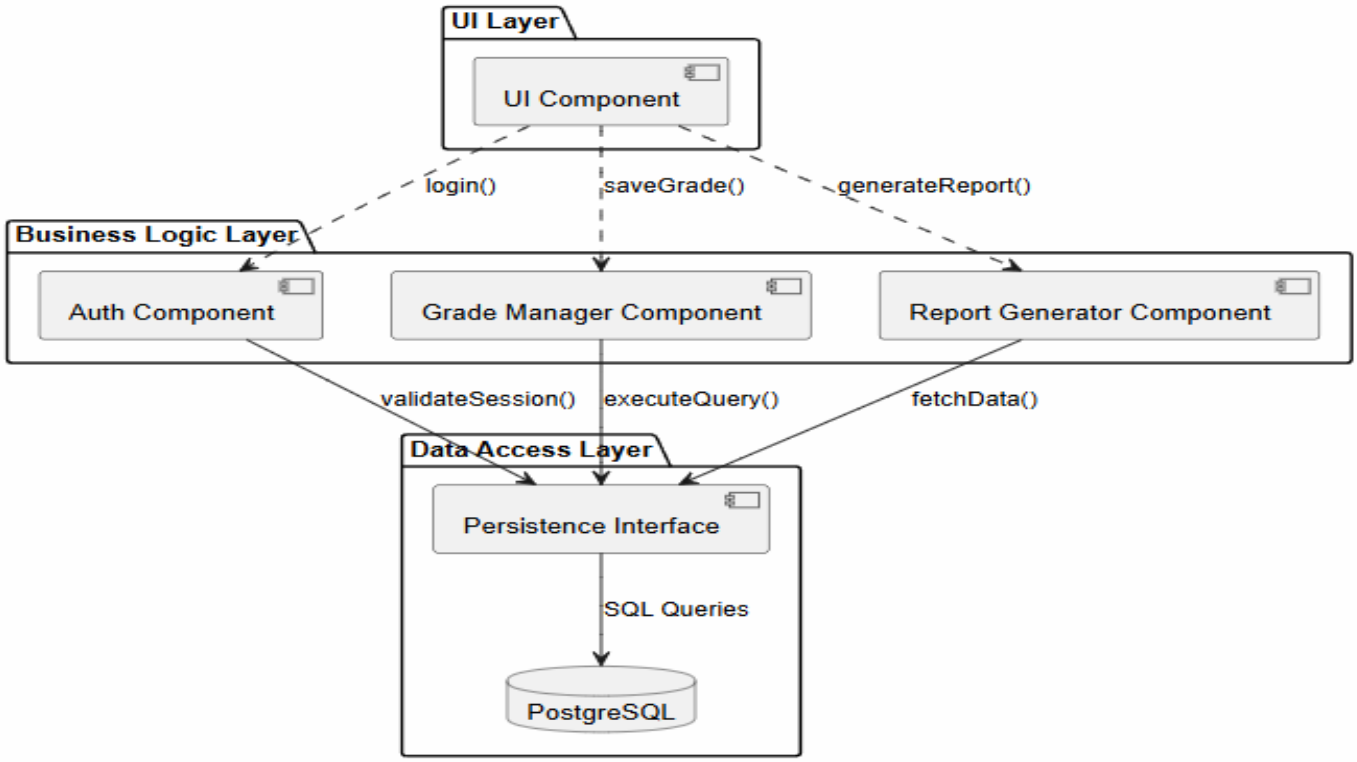


ДОДАТОК В – Блок-схема алгоритму реєстрації результатів успішності



ДОДАТОК Г – Схема структури інсталяційного пакету системи

Додаток Д – Деталізована компонентна структура та взаємодія модулів системи



ДОДАТОК Е– Звіт про результати тестування функціональних модулів

Модуль системи	Сценарій тесту	Очікуваний результат	Результат	Статус
Автентифікація	Вхід з невірним паролем	Виведення повідомлення про помилку	Помилку відображено	Успішно
Облік успішності	Введення оцінки > 100	Повідомлення про помилку валідації	Помилку відображено	Успішно
Звітність	Генерація звіту порожньої групи	Попередження про відсутність даних	Попередження відображено	Успішно
Навігація	Перехід між формами	Коректне завантаження інтерфейсу	Форми завантажено	Успішно

ДОДАТОК Є – Словник даних інформаційної бази

Сутність	Атрибут	Тип даних	Опис
Student	student_id	SERIAL (PK)	Унікальний ідентифікатор студента
	full_name	VARCHAR(100)	Прізвище, ім'я та по батькові здобувача
	group_name	VARCHAR(20)	Назва академічної групи
Teacher	teacher_id	SERIAL (PK)	Унікальний ідентифікатор викладача
	full_name	VARCHAR(100)	Прізвище, ім'я та по батькові викладача
	department	VARCHAR(100)	Назва кафедри
Subject	subject_id	SERIAL (PK)	Унікальний ідентифікатор дисципліни
	subject_name	VARCHAR(150)	Назва навчальної дисципліни
	credits	INT	Кількість кредитів ECTS
	teacher_id	INT (FK)	Ідентифікатор викладача, що веде курс
Grade	grade_id	SERIAL (PK)	Унікальний ідентифікатор запису оцінки
	grade_value	INT	Значення оцінки за шкалою
	date_issued	DATE	Дата виставлення оцінки

Продовження ДОДАТКУ Є

	student_id	INT (FK)	Ідентифікатор студента
	subject_id	INT (FK)	Ідентифікатор дисципліни

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

ДОДАТОК Г

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, Заєць Олександр Васильович, здобувач(ка) НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота (проєкт) на тему «Програмне забезпечення для автоматизованого обліку та управління навчальним процесом у закладі освіти» 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 ШІ: ChatGPT, Gemini,були використані для:
        - [ ] перекладу іншомовних джерел;
        - [ ] стилістичного редагування та перевірки граматики;

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

«___» _____ 2026 р. _____ **Олександр Заєць**

(підпис)