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 системи рекомендацій навчальних матеріалів на основі аналізу прогресу студента»

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Гарант освітньої програми
к.т.н., доцент _____ **Ганна ВАЙГАНГ**
(підпис)

Керівник бакалаврської кваліфікаційної роботи
(науковий ступінь та вчене звання) _____ **Роман РУДЕНСЬКИЙ**
(підпис)

Виконав _____ **Антон ХАРЧЕНКО**
(підпис)

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

Харченку Антону Володимировичу
(прізвище, ім'я, по батькові)

Спеціальність: 121 – «Інженерія програмного забезпечення»
Освітньо-професійна програма: «Інженерія програмного забезпечення»
Тема бакалаврської кваліфікаційної роботи

«Програмне забезпечення системи рекомендацій навчальних матеріалів на основі аналізу прогресу студента».

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: отримання результатів оцінювання робіт учнів та рекомендованих навчальних матеріалів у електронному форматі.
Перелік питань, які потрібно розробити:

- Аналіз проблемної області
- Моделювання предметної області
- Проєктування програмної системи
- Впровадження та експлуатація системи

Перелік графічних документів (за потреби)

_____

Дата видачі завдання "_____"_____20___ р.

Керівник бакалаврської кваліфікаційної роботи

_____проф. д.е.н._____	_____	__Руденський Р. А.__
(науковий ступінь та вчене звання)	(підпис)	(ПІБ)

Завдання прийняв до виконання _____ ____Харченко А.В._____
(підпис) (ПІБ студента)

ЗМІСТ

ВСТУП

Сучасний етап розвитку інформаційних технологій характеризується активним впровадженням цифрових рішень у сферу освіти, що зумовлено необхідністю забезпечення доступності, гнучкості та ефективності навчального процесу. Особливої актуальності набувають системи дистанційного навчання, які дозволяють організовувати освітній процес незалежно від місця перебування користувачів, забезпечувати доступ до навчальних матеріалів у будь-який час, а також автоматизувати процеси взаємодії між викладачами та студентами. Використання електронних освітніх платформ стало невід'ємною складовою сучасного освітнього середовища, оскільки такі системи забезпечують централізоване управління навчальними курсами, контроль успішності студентів, розміщення навчальних матеріалів та підтримку комунікації між учасниками навчального процесу.

Однією з головних проблем сучасних систем електронного навчання є недостатній рівень персоналізації освітнього процесу. Більшість існуючих платформ дистанційного навчання надають однаковий набір навчальних матеріалів для всіх студентів незалежно від рівня їх знань, успішності, швидкості засвоєння інформації та індивідуальних особливостей навчання. Такий підхід знижує ефективність освітнього процесу, оскільки не враховує потреб конкретного користувача та не дозволяє адаптувати навчальний контент відповідно до його навчального прогресу.

Актуальність теми бакалаврської роботи полягає у необхідності створення програмного забезпечення, яке поєднуватиме функціональні можливості систем дистанційного навчання та механізми інтелектуального аналізу даних для підвищення ефективності навчального процесу. Реалізація

системи рекомендацій навчальних матеріалів на основі аналізу прогресу студента дозволить забезпечити адаптивний підхід до навчання, сприятиме покращенню засвоєння навчального матеріалу та підвищенню рівня успішності студентів. Крім того, використання рекомендаційних алгоритмів у сфері електронної освіти є перспективним напрямом розвитку сучасних інформаційних технологій, що активно застосовується у провідних освітніх платформах та онлайн-сервісах.

Метою бакалаврської роботи є покращення поточної сфери онлайн-освіти для учнів за допомогою розробки веб-орієнтованого програмного забезпечення системи рекомендацій навчальних матеріалів на основі аналізу прогресу студента, яке забезпечуватиме автоматизацію процесів дистанційного навчання, управління навчальними курсами, контролю успішності користувачів та формування персоналізованих рекомендацій освітнього контенту відповідно до результатів навчальної діяльності студентів.

Для досягнення поставленої мети необхідно виконати такі **завдання:**

1. здійснити аналіз предметної області дистанційного навчання та сучасних систем управління навчальним процесом;
2. дослідити особливості функціонування рекомендаційних систем та методів персоналізації навчального контенту;
3. визначити функціональні та нефункціональні вимоги до програмного забезпечення;
4. спроєктувати архітектуру веб-застосунку та структуру бази даних;
5. реалізувати серверну та клієнтську частини програмного забезпечення;
6. реалізувати механізм аналізу прогресу студентів та формування рекомендацій навчальних матеріалів;
7. провести тестування програмного забезпечення та оцінити ефективність його роботи.

Об’єктом дослідження є процес організації дистанційного навчання із використанням інформаційних технологій.

Предметом дослідження є методи та програмні засоби аналізу навчального прогресу студентів і формування рекомендацій навчальних матеріалів у системах електронного навчання.

Під час розробки програмного забезпечення використовуються сучасні методи та технології вебпрограмування, проєктування інформаційних систем та управління базами даних. Серверна частина застосунку реалізується за допомогою мови програмування PHP та фреймворку Symfony, що дозволяє створити структуровану, масштабовану та безпечну архітектуру програмного забезпечення. Для зберігання та обробки даних використовується реляційна система керування базами даних MySQL. Клієнтська частина програмного забезпечення створюється із застосуванням HTML, CSS та JavaScript для забезпечення зручного та адаптивного користувацького інтерфейсу. Для моделювання структури системи та опису взаємодії її компонентів використовуються UML-діаграми.

Результати бакалаврської роботи апробовано у тезах доповіді на V Всеукраїнській науково-практичній конференції «Сучасні інформаційні технології та програмне забезпечення», що відбулася у 2026 році.

Структура записки: бакалаврська робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. Загальний обсяг роботи становить 77 сторінок. Список використаних джерел містить 30 посилань. Робота також містить 3 додатки.

У першому розділі наведено аналіз предметної області, дослідження сучасних систем дистанційного навчання та рекомендаційних систем. Другий розділ присвячений проєктуванню програмного забезпечення, моделюванню бази даних та розробці UML-діаграм. У третьому розділі описано процес програмної реалізації вебзастосунку, створення серверної та клієнтської частин системи, а також реалізацію механізму рекомендації навчальних матеріалів. У четвертому розділі наведено результати тестування програмного

забезпечення, аналіз ефективності роботи системи та перспективи її подальшого розвитку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Сучасний розвиток інформаційних технологій суттєво вплинув на всі сфери діяльності суспільства, зокрема й на освітню галузь. Активне впровадження цифрових технологій у навчальний процес сприяло появі та стрімкому розвитку систем дистанційного навчання, які забезпечують можливість організації освітнього процесу за допомогою мережі Інтернет. Такі системи дозволяють автоматизувати процеси управління навчальними матеріалами, взаємодію між викладачами та студентами, контроль знань, оцінювання результатів навчання та моніторинг навчальної активності користувачів.

Особливої актуальності дистанційне навчання набуло в умовах глобальної цифровізації освіти, а також через необхідність забезпечення безперервного доступу до освітніх ресурсів незалежно від місця перебування студентів. Використання електронних освітніх платформ дозволяє значно розширити можливості традиційного навчання, забезпечуючи доступ до навчальних матеріалів у будь-який час, спрощуючи комунікацію між учасниками освітнього процесу та підвищуючи рівень автоматизації адміністративних і навчальних процедур.

Системи дистанційного навчання являють собою спеціалізовані програмні комплекси, призначені для організації, підтримки та контролю освітнього процесу в електронному середовищі. Основною метою таких систем є забезпечення ефективної взаємодії між викладачем та студентом, централізоване управління навчальними курсами, автоматизація процесів

оцінювання та створення умов для самостійного опрацювання навчального матеріалу [1].

Серед найбільш поширених функціональних можливостей сучасних систем дистанційного навчання можна виділити:

- створення та адміністрування навчальних курсів;
- завантаження та структурування навчальних матеріалів;
- організацію тестування та перевірки знань;
- контроль успішності студентів;
- взаємодію між учасниками навчального процесу;
- формування статистики та аналітики навчальної діяльності;
- управління ролями та правами доступу користувачів.

Однією з найвідоміших систем дистанційного навчання є Moodle, яка широко використовується у закладах освіти різного рівня. Дана платформа забезпечує можливість створення курсів, розміщення навчальних матеріалів, проходження тестів, оцінювання студентів та ведення електронного журналу успішності. Водночас більшість існуючих систем орієнтовані переважно на управління навчальним контентом і не забезпечують достатнього рівня інтелектуального аналізу навчальної діяльності студентів.

Суттєвим недоліком багатьох сучасних платформ електронного навчання є відсутність механізмів персоналізації навчального процесу. У більшості випадків усі студенти отримують однаковий набір навчальних матеріалів незалежно від рівня знань, швидкості засвоєння інформації або результатів виконання завдань. Такий підхід не дозволяє враховувати індивідуальні особливості користувачів та знижує ефективність навчального процесу.

У зв’язку з цим важливого значення набувають рекомендаційні системи, які дозволяють автоматично формувати персоналізовані рекомендації навчальних матеріалів на основі аналізу поведінки та успішності

студентів. Інтеграція рекомендаційних алгоритмів у системи дистанційного навчання дозволяє реалізувати адаптивний підхід до навчання, за якого кожен студент отримує індивідуально підібраний освітній контент відповідно до власного рівня підготовки та навчального прогресу.

Адаптивне навчання є одним із перспективних напрямів розвитку сучасних інформаційних систем у сфері освіти. Основна ідея такого підходу полягає у динамічному налаштуванні навчального процесу відповідно до характеристик конкретного користувача. Це дозволяє не лише підвищити ефективність засвоєння знань, але й покращити мотивацію студентів до навчання, оскільки навчальні матеріали стають більш релевантними та відповідають індивідуальним потребам користувача [2].

Для реалізації адаптивного навчання системи повинні здійснювати аналіз різних параметрів навчальної діяльності студентів, серед яких:

- результати виконання практичних завдань;
- рівень успішності за окремими темами;
- частота використання навчальних матеріалів;
- активність у межах навчального курсу;
- час виконання завдань;
- динаміка змін успішності;
- рівень засвоєння окремих категорій навчального матеріалу.

На основі отриманих даних система може визначати слабкі місця у знаннях студента та рекомендувати додаткові навчальні матеріали для покращення результатів навчання. Таким чином, рекомендаційна система виступає важливим інструментом інтелектуальної підтримки освітнього процесу.

Розробка програмного забезпечення системи рекомендацій навчальних матеріалів є актуальним завданням, оскільки поєднання функціоналу дистанційного навчання та механізмів персоналізації дозволяє створити сучасну освітню платформу, орієнтовану на індивідуальні потреби студентів.

Такий підхід забезпечує підвищення якості навчального процесу, ефективності засвоєння знань та рівня автоматизації управління освітньою діяльністю.

Детальний опис класів та атрибутів предметної області наведено у таблиці 1.1.

Таблиця 1.1

Опис атрибутів класів предметної області

Клас предметної області	**Атрибут**	**Опис**
Користувач	Ім'я	Повне ім'я користувача (здобувача вищої освіти або викладача).
	Роль	Визначає, чи є користувач здобувач вищої освіти або викладачем.
	Електронна пошта	Використовується для входу в систему та отримання сповіщень.
Дисципліна	Назва дисципліни	Найменування навчальної дисципліни.
	Викладач	Особа, яка веде дисципліну та керує нею.
	Здобувач вищої освіти	Список здобувачів вищої освіти, які вивчають дисципліну.
Завдання	Назва роботи	Тема або заголовок конкретного завдання.
	Категорія	Тип роботи (лабораторна, практична, модульна тощо).
	Дисципліна	До якої навчальної дисципліни належить це завдання.
	Дедлайн	Кінцевий термін здачі завдання.
Рекомендація	Навчальний матеріал	Матеріал, який рекомендується системою

1.2 Огляд існуючих рішень

Сучасний розвиток цифрових технологій та активне впровадження дистанційного навчання сприяли появі великої кількості програмних платформ, призначених для організації освітнього процесу в електронному середовищі. Такі системи забезпечують автоматизацію управління навчальними курсами, взаємодію між викладачами та студентами, контроль успішності, проведення тестування та зберігання навчальних матеріалів. Значна частина сучасних освітніх платформ реалізується у вигляді веборієнтованих систем, що дозволяє забезпечити доступ до освітнього контенту незалежно від місця перебування користувача та використовуваного пристрою.

Незважаючи на велику кількість існуючих систем дистанційного навчання, проблема персоналізації освітнього процесу залишається недостатньо вирішеною. Більшість платформ орієнтовані переважно на управління навчальними матеріалами та адміністрування освітнього процесу, тоді як функціональність інтелектуального аналізу навчальної діяльності студентів і формування персоналізованих рекомендацій реалізована частково або повністю відсутня. У зв’язку з цим доцільним є аналіз існуючих програмних рішень з метою визначення їх функціональних можливостей, переваг та недоліків.

Moodle є однією з найвідоміших та найбільш поширених систем дистанційного навчання у світі. Дана платформа належить до класу систем управління навчанням (Learning Management System) та використовується закладами освіти, навчальими центрами й організаціями для організації освітнього процесу у дистанційному форматі. Система дозволяє забезпечити централізоване управління навчальними курсами, взаємодію між викладачами та студентами, контроль успішності користувачів, а також зберігання та структурування навчальних матеріалів [3].

Основною особливістю системи є її відкритий вихідний код, що надає можливість змінювати та розширювати функціональність відповідно до потреб конкретного закладу освіти. Завдяки цьому платформа отримала широке поширення серед університетів та інших навчальних установ, оскільки дозволяє адаптувати систему до специфіки організації освітнього процесу. Крім того, система підтримує модульну архітектуру, що забезпечує можливість підключення додаткових плагінів та інтеграції із зовнішніми сервісами.

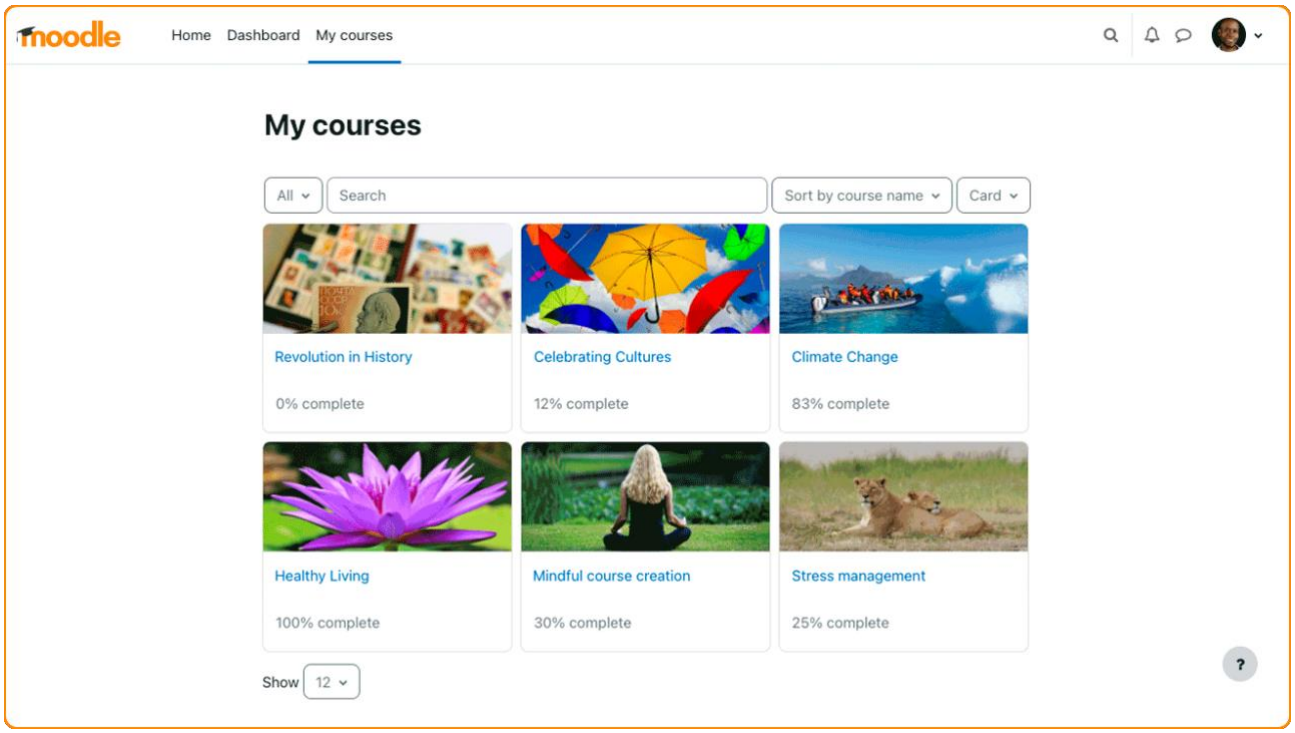

Рисунок . 1.1 - Інформаційна система «Moodle»

Система надає широкі можливості для створення та адміністрування навчальних курсів. Викладачі можуть додавати текстові матеріали, презентації, відеофайли, посилання на зовнішні ресурси, практичні завдання та тестування. У межах навчального курсу передбачено можливість встановлення дедлайнів виконання робіт, перевірки студентських завдань, виставлення оцінок та надання коментарів щодо виконаних робіт. Усі результати навчальної діяльності студентів зберігаються у системі, що дозволяє здійснювати моніторинг успішності та аналізувати навчальний прогрес користувачів.

Важливою перевагою платформи є підтримка різних ролей користувачів. У системі можуть бути визначені адміністратори, викладачі та студенти, причому кожна роль має власний набір функціональних можливостей та прав доступу. Такий підхід дозволяє забезпечити безпечне управління інформацією та розмежування доступу до окремих компонентів системи.

Платформа також містить інструменти для організації комунікації між учасниками освітнього процесу. Зокрема, підтримуються форуми, чати, обговорення та система повідомлень, що дозволяє забезпечити взаємодію між студентами та викладачами у межах навчальних курсів. Це сприяє підвищенню ефективності дистанційного навчання та створенню більш інтерактивного освітнього середовища [3].

Попри значну кількість функціональних можливостей, система має і певні недоліки. Одним із найбільш помітних є складність інтерфейсу користувача. Через велику кількість налаштувань та модулів новим користувачам може бути складно швидко освоїти роботу із системою. Крім того, окремі елементи інтерфейсу виглядають перевантаженими, що негативно впливає на зручність використання платформи.

Ще одним суттєвим недоліком є недостатній рівень персоналізації навчального процесу. Хоча система дозволяє збирати інформацію про успішність студентів та їхню активність, механізми автоматичного аналізу прогресу користувачів і формування рекомендацій навчальних матеріалів реалізовані обмежено. У більшості випадків усі студенти отримують однаковий набір навчального контенту незалежно від індивідуального рівня знань або результатів навчання. Для реалізації адаптивного навчання та рекомендаційних механізмів зазвичай необхідно використовувати додаткові плагіни або сторонні програмні рішення.

Таким чином, Moodle є потужною платформою дистанційного навчання, яка забезпечує ефективну організацію освітнього процесу та автоматизацію управління навчальними курсами. Водночас аналіз системи

демонструє недостатній рівень реалізації інтелектуальних механізмів персоналізації навчального контенту. Саме тому розробка програмного забезпечення системи рекомендацій навчальних матеріалів на основі аналізу прогресу студента є актуальним напрямом удосконалення сучасних освітніх інформаційних систем та дозволяє забезпечити більш адаптивний і ефективний підхід до організації навчального процесу.

Розглянемо ще одне програмне рішення, представлене на ринку електронного навчання.

eLearn є сучасною системою дистанційного навчання, призначеною для організації освітнього процесу у цифровому середовищі. Дана платформа використовується для створення навчальних курсів, розміщення освітніх матеріалів, контролю знань студентів та забезпечення взаємодії між учасниками навчального процесу. Основною метою системи є автоматизація процесів електронного навчання та забезпечення зручного доступу до навчального контенту через вебінтерфейс.

Система орієнтована на використання у закладах освіти, навчальних центрах та корпоративному секторі. Платформа дозволяє викладачам створювати структуровані навчальні курси, додавати лекційні матеріали, презентації, тестові завдання та практичні роботи. Студенти, у свою чергу, отримують можливість переглядати навчальний контент, виконувати завдання, проходити тестування та відстежувати результати власної навчальної діяльності.

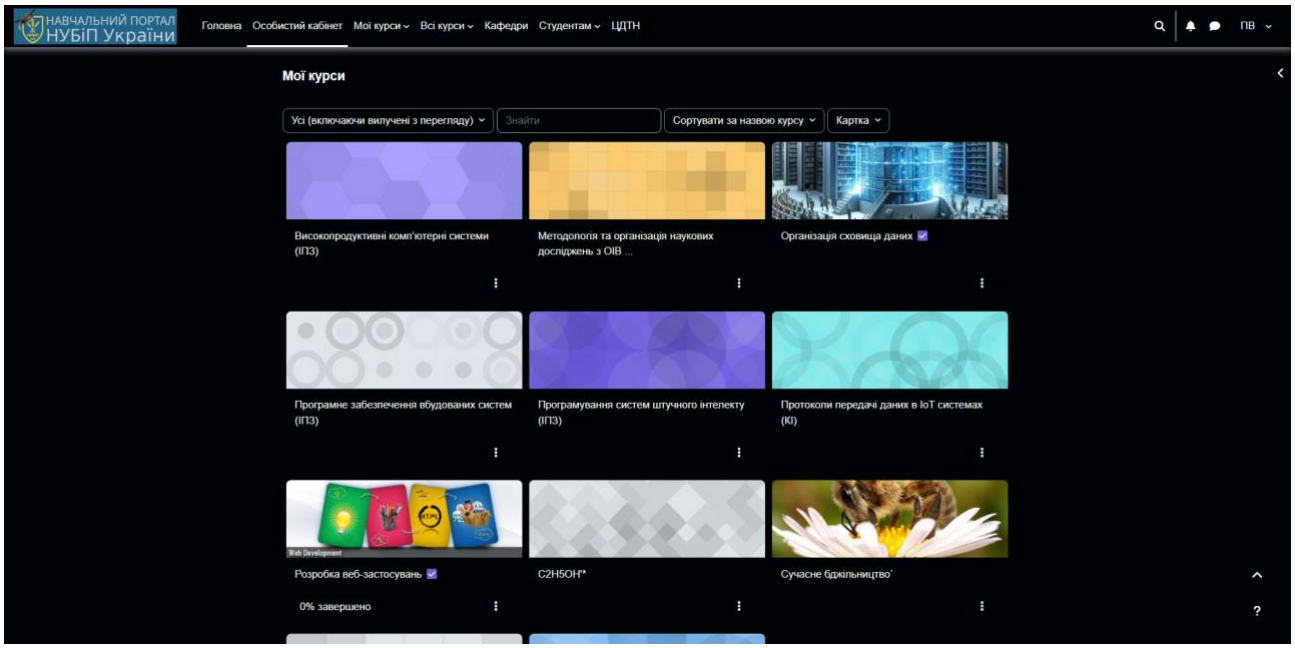

Рисунок 1.2 - Інформаційна система «eLearn»

Однією з головних особливостей системи є простий та зрозумілий інтерфейс користувача. Платформа орієнтована на забезпечення швидкого доступу до основних функцій системи, що спрощує процес роботи як для викладачів, так і для студентів. Інтерфейс побудований таким чином, щоб мінімізувати складність навігації та забезпечити комфортну взаємодію користувачів із навчальним середовищем [4].

Система підтримує функціональність управління користувачами та ролями. Адміністратори мають можливість створювати облікові записи, керувати правами доступу та контролювати роботу системи. Викладачі можуть створювати дисципліни, додавати навчальні матеріали та перевіряти роботи студентів. Студенти отримують доступ до навчальних курсів, результатів оцінювання та інформації щодо власного прогресу.

Важливою складовою системи є модуль оцінювання та моніторингу успішності. Платформа дозволяє здійснювати перевірку виконаних завдань, виставляти оцінки та формувати статистику навчальної діяльності студентів. Завдяки цьому викладачі можуть контролювати рівень засвоєння навчального матеріалу та аналізувати результати навчання.

Попри значну кількість переваг, система має і певні обмеження. Зокрема, функціональність персоналізації навчального процесу реалізована недостатньо повно. У більшості випадків система надає однаковий набір навчальних матеріалів усім користувачам без урахування їх індивідуального рівня підготовки, успішності або навчальної активності. Механізми аналізу прогресу студентів мають переважно статистичний характер та не забезпечують формування інтелектуальних рекомендацій щодо подальшого вивчення матеріалів.

Крім того, система обмежено використовує можливості рекомендаційних алгоритмів та адаптивного навчання. Це знижує рівень індивідуалізації освітнього процесу та не дозволяє повною мірою враховувати особливості навчальної діяльності кожного студента. У сучасних умовах цифровізації освіти така функціональність є особливо важливою, оскільки персоналізований підхід до навчання сприяє підвищенню ефективності засвоєння знань та мотивації студентів [4].

Таким чином, eLearn є функціональною системою дистанційного навчання, яка забезпечує автоматизацію основних процесів організації освітньої діяльності. Водночас аналіз платформи демонструє недостатній рівень реалізації інтелектуальних механізмів персоналізації навчального контенту та рекомендаційних систем, що підтверджує актуальність розробки програмного забезпечення системи рекомендацій навчальних матеріалів на основі аналізу прогресу студента.

1.3 Постановка завдання

Організація системи рекомендацій навчальних матеріалів на основі аналізу прогресу студента базується на необхідності забезпечення ефективного управління освітнім процесом, автоматизації взаємодії між викладачами та здобувачами вищої освіти, а також персоналізації навчального контенту відповідно до індивідуальних результатів навчальної діяльності

користувачів. Система функціонує як централізоване веборієнтоване середовище, у межах якого реалізуються процеси створення навчальних дисциплін, розміщення освітніх матеріалів, виконання та перевірки завдань, оцінювання студентів і формування рекомендацій щодо подальшого навчання.

Основним завданням викладача в системі є організація та супровід навчального процесу. Викладач повинен мати можливість створювати навчальні дисципліни, додавати до них здобувачів вищої освіти, розміщувати навчальні матеріали та практичні завдання, а також здійснювати контроль успішності студентів. Крім того, система повинна забезпечувати викладачу доступ до аналітичної інформації, яка характеризує рівень активності студентів та результати їх навчальної діяльності. До таких показників належать результати виконання завдань, середній рівень успішності, своєчасність подання робіт, активність використання навчальних матеріалів та динаміка зміни навчального прогресу.

Здобувач вищої освіти у межах системи отримує доступ до навчальних дисциплін, матеріалів та завдань, які були призначені викладачем. Користувач має можливість переглядати освітній контент, завантажувати виконані роботи, проходити тестування та контролювати власні результати навчання. Важливою особливістю системи є реалізація механізму персоналізованих рекомендацій, який формує індивідуальні пропозиції щодо навчальних матеріалів на основі аналізу успішності та навчальної активності студента. Такий підхід дозволяє адаптувати навчальний процес до індивідуальних потреб користувача та сприяє підвищенню ефективності засвоєння знань.

Програмний додаток повинен підтримувати централізовану базу даних, у якій зберігається інформація про користувачів, навчальні дисципліни, матеріали, завдання, результати оцінювання та сформовані рекомендації. Цілісність та узгодженість даних є критично важливими вимогами до системи, оскільки програмне забезпечення повинно забезпечувати коректне збереження результатів навчальної діяльності, уникати втрати інформації та запобігати несанкціонованому доступу до персональних даних користувачів.

Основні функціональні можливості системи повинні забезпечувати:

1. створення та адміністрування навчальних дисциплін;
2. додавання здобувачів вищої освіти до навчальних курсів;
3. розміщення навчальних матеріалів та практичних завдань;
4. завантаження та перевірку студентських робіт;
5. оцінювання результатів навчальної діяльності;
6. аналіз успішності та активності користувачів;
7. формування персоналізованих рекомендацій навчальних матеріалів;
8. моніторинг навчального прогресу студентів.

Процес роботи із завданнями організовується за визначеним життєвим циклом. Після створення завдання викладачем система зберігає інформацію про його опис, категорію, термін виконання та максимальну кількість балів. Здобувач вищої освіти отримує доступ до завдання та може завантажити виконану роботу у встановлений термін. Після подання роботи викладач перевіряє результат виконання, виставляє оцінку та залишає коментарі щодо якості роботи. Усі зміни статусів та результати оцінювання фіксуються у системі та використовуються для подальшого аналізу навчального прогресу студента.

1.4 Функціональні та нефункціональні вимоги

Функціональні вимоги визначають основні можливості програмного забезпечення та описують функції, які система повинна виконувати для забезпечення ефективної організації дистанційного навчання й формування рекомендацій навчальних матеріалів на основі аналізу прогресу студента.

Система повинна забезпечувати:

1. реєстрацію та авторизацію користувачів у системі;
2. розмежування ролей користувачів на викладачів та здобувачів вищої освіти [5];

3. створення, редагування та видалення навчальних дисциплін;
4. додавання здобувачів вищої освіти до навчальних дисциплін;
5. завантаження та зберігання навчальних матеріалів;
6. створення практичних, лабораторних та модульних завдань;
7. встановлення дедлайнів для виконання завдань;
8. завантаження студентами виконаних робіт;
9. перевірку та оцінювання студентських робіт викладачем;
10. додавання коментарів до результатів оцінювання;
11. перегляд студентами власних оцінок та результатів навчання;
12. аналіз успішності та активності користувачів;
13. формування персоналізованих рекомендацій навчальних матеріалів;
14. відображення статистики навчального прогресу;
15. пошук та фільтрацію навчальних матеріалів;
16. збереження історії виконаних завдань та результатів оцінювання;
17. захист персональних даних користувачів;
18. ведення централізованої бази даних користувачів, курсів та навчальних матеріалів.

Нефункціональні вимоги визначають характеристики якості програмного забезпечення, особливості його функціонування, продуктивності, безпеки та зручності використання.

Програмне забезпечення повинно відповідати таким нефункціональним вимогам:

1. система повинна мати зрозумілий та інтуїтивно зрозумілий інтерфейс користувача;
2. вебзастосунок повинен коректно працювати у сучасних браузерах;
3. система повинна підтримувати адаптивний інтерфейс для різних типів пристроїв;

4. час обробки основних запитів користувача не повинен перевищувати кількох секунд;
5. система повинна забезпечувати безпечне зберігання персональних даних користувачів [6];
6. доступ до функціональних можливостей повинен обмежуватися відповідно до ролі користувача;
7. програмне забезпечення повинно забезпечувати цілісність та узгодженість даних;
8. система повинна бути масштабованою та придатною до подальшого розширення функціональності;
9. архітектура програмного забезпечення повинна підтримувати модульність компонентів;
10. система повинна забезпечувати стабільну роботу при одночасній роботі декількох користувачів;

1.5 Вимоги до інтерфейсу користувача

Інтерфейс користувача є однією з найважливіших складових програмного забезпечення, оскільки саме через нього здійснюється взаємодія користувачів із функціональними можливостями системи. Від якості організації інтерфейсу значною мірою залежить ефективність роботи користувачів, швидкість виконання основних операцій та загальний рівень зручності використання програмного забезпечення. У межах розробки системи рекомендацій навчальних матеріалів на основі аналізу прогресу студента особлива увага повинна приділятися створенню інтуїтивно зрозумілого, логічно структурованого та адаптивного вебінтерфейсу.

Інтерфейс системи повинен забезпечувати просту та зручну навігацію між основними функціональними модулями програмного забезпечення. Користувачі повинні мати можливість швидко отримувати доступ до навчальних дисциплін, навчальних матеріалів, завдань, результатів

оцінювання та рекомендаційних матеріалів без необхідності виконання складних або зайвих дій. Структура інтерфейсу повинна бути побудована таким чином, щоб основні функції системи були зрозумілими навіть для користувачів, які не мають значного досвіду роботи із подібними інформаційними системами [7].

Важливою вимогою до інтерфейсу є його адаптивність. Програмне забезпечення повинно коректно відображатися та функціонувати на різних типах пристроїв, зокрема на персональних комп'ютерах, ноутбуках, планшетах та смартфонах. Адаптивний дизайн дозволяє забезпечити комфортне використання системи незалежно від роздільної здатності екрана або використовуваного браузера. Це є особливо важливим для систем дистанційного навчання, оскільки користувачі можуть взаємодіяти із платформою з різних пристроїв та в різних умовах.

Інтерфейс повинен підтримувати чітке візуальне розмежування функціональних елементів системи. Навігаційні меню, форми введення даних, кнопки управління та інформаційні блоки мають бути логічно структурованими та візуально зрозумілими для користувача. Використання єдиного стилю оформлення інтерфейсу сприяє підвищенню зручності використання системи та забезпечує цілісність користувацького досвіду.

Для викладачів інтерфейс повинен забезпечувати зручні засоби управління навчальними дисциплінами, створення завдань, додавання навчальних матеріалів та перевірки студентських робіт. Важливою вимогою є можливість швидкого перегляду результатів навчальної діяльності студентів, доступ до статистики успішності та зручний механізм оцінювання виконаних робіт. Система повинна мінімізувати кількість дій, необхідних для виконання типових операцій, що дозволить підвищити ефективність роботи викладача.

Для здобувачів вищої освіти інтерфейс повинен забезпечувати зручний доступ до навчального контенту, інформації про дедлайни, результатів оцінювання та рекомендаційних матеріалів. Особливе значення має наочне відображення прогресу навчання, що дозволяє студенту контролювати власні

результати та своєчасно реагувати на зниження успішності. Блок рекомендацій навчальних матеріалів повинен бути інтегрований у структуру інтерфейсу таким чином, щоб користувач міг швидко переглядати запропоновані системою матеріали та переходити до їх опрацювання.

Інтерфейс користувача також повинен забезпечувати коректне відображення повідомлень про помилки, результатів виконання операцій та системних сповіщень. Усі інформаційні повідомлення повинні бути зрозумілими та містити достатню кількість інформації для пояснення причин помилки або результату виконаної дії. Це дозволить зменшити кількість помилок під час роботи із системою та покращити загальний рівень взаємодії користувачів із програмним забезпеченням.

Однією з важливих вимог до інтерфейсу є забезпечення швидкодії та плавності роботи вебзастосунку. Усі сторінки системи повинні завантажуватися швидко, а взаємодія користувача з елементами інтерфейсу повинна відбуватися без помітних затримок. Це позитивно впливає на користувацький досвід та забезпечує комфортне використання системи у процесі навчання.

Таким чином, інтерфейс користувача системи рекомендацій навчальних матеріалів повинен поєднувати простоту використання, адаптивність, функціональність та логічну структуру. Реалізація якісного користувацького інтерфейсу є важливою складовою ефективного функціонування програмного забезпечення та забезпечує комфортну взаємодію користувачів із системою дистанційного навчання.

2 МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 Загальні відомості

Моделювання предметної області є одним із найважливіших етапів розробки програмного забезпечення, оскільки саме на цьому етапі здійснюється формалізація основних процесів, об’єктів та взаємозв’язків, які існують у межах досліджуваної системи. Якісне моделювання дозволяє сформувати цілісне уявлення про структуру майбутнього програмного продукту, визначити основні функціональні компоненти системи, а також забезпечити правильну організацію даних і логіки взаємодії між окремими елементами програмного забезпечення [8].

У межах розробки системи рекомендацій навчальних матеріалів на основі аналізу прогресу студента моделювання предметної області спрямоване на опис процесів дистанційного навчання, взаємодії між викладачами та здобувачами вищої освіти, організації навчальних дисциплін, контролю успішності та формування персоналізованих рекомендацій навчального контенту. Основною метою моделювання є створення логічної структури системи, яка дозволить реалізувати ефективне управління навчальним процесом та забезпечити підтримку адаптивного навчання.

Предметна область охоплює процеси створення та адміністрування навчальних дисциплін, розміщення освітніх матеріалів, створення практичних і лабораторних завдань, завантаження виконаних робіт студентами, перевірки та оцінювання результатів навчальної діяльності, а також аналізу навчального прогресу користувачів. Особливістю предметної області є наявність механізму персоналізації навчального процесу, який базується на аналізі успішності та активності студентів для формування рекомендацій щодо подальшого вивчення навчальних матеріалів.

Основними об’єктами предметної області є користувачі системи, навчальні дисципліни, завдання, навчальні матеріали, результати оцінювання,

подані роботи та рекомендації навчального контенту. Кожен із цих об’єктів характеризується певним набором властивостей та взаємозв’язків із іншими компонентами системи. Наприклад, викладач може створювати дисципліни та додавати навчальні матеріали, тоді як здобувач вищої освіти має можливість переглядати матеріали, виконувати завдання та отримувати рекомендації відповідно до власного прогресу навчання.

Важливим аспектом моделювання предметної області є визначення ролей користувачів та особливостей їх взаємодії із системою. У межах програмного забезпечення передбачено використання декількох ролей, серед яких основними є викладач та здобувач вищої освіти. Кожна роль має власний набір функціональних можливостей та рівень доступу до інформації. Такий підхід дозволяє забезпечити безпечне управління даними та логічне розмежування функціональності системи [8].

Під час моделювання також визначаються основні бізнес-процеси, які повинні підтримуватися програмним забезпеченням. До них належать процеси створення дисциплін, призначення завдань, подання робіт, оцінювання результатів навчання, аналізу успішності та формування рекомендацій. Формалізація цих процесів дозволяє забезпечити структуровану організацію системи та спрощує подальше проєктування архітектури програмного забезпечення.

Результатом моделювання предметної області є створення концептуальної моделі системи, яка відображає структуру даних, взаємозв’язки між об’єктами та логіку функціонування програмного забезпечення. Отримані моделі використовуються як основа для подальшого проєктування бази даних, побудови UML-діаграм та реалізації програмних компонентів вебзастосунку.

2.2 Об’єктне та функціональне моделювання

2.2.1 Діаграма прецедентів. Use-case діаграма є одним із основних інструментів моделювання функціональних можливостей програмного забезпечення та використовується для відображення взаємодії користувачів із системою. Даний тип UML-діаграм дозволяє наочно представити основні сценарії використання програмного забезпечення, визначити ролі користувачів та описати функції, які система повинна забезпечувати для кожної категорії користувачів. Використання use-case діаграми є важливим етапом проєктування інформаційної системи, оскільки вона дозволяє формалізувати функціональні вимоги та визначити межі взаємодії між користувачами та програмним забезпеченням.

У межах розробки системи рекомендацій навчальних матеріалів на основі аналізу прогресу студента use-case діаграма використовується для опису основних процесів дистанційного навчання, управління навчальними дисциплінами, роботи із завданнями та формування рекомендацій освітнього контенту. Діаграма дозволяє відобразити функціональність системи з точки зору кінцевих користувачів та забезпечує розуміння логіки взаємодії між окремими компонентами програмного забезпечення.

Основними акторами системи є викладач та здобувач вищої освіти. Кожен із користувачів взаємодіє із системою відповідно до власної ролі та набору функціональних можливостей. Викладач виконує функції створення та адміністрування навчальних дисциплін, додавання навчальних матеріалів, створення завдань, перевірки виконаних робіт та оцінювання результатів навчання студентів. Крім того, викладач має можливість переглядати статистику успішності користувачів та аналізувати результати навчальної діяльності.

Здобувач вищої освіти взаємодіє із системою шляхом перегляду навчальних дисциплін, отримання доступу до навчальних матеріалів, виконання та завантаження завдань, перегляду оцінок та контролю власного

прогресу навчання. Однією з ключових функцій системи для студента є отримання персоналізованих рекомендацій навчальних матеріалів, які формуються на основі аналізу результатів навчальної діяльності користувача.

Use-case діаграма також дозволяє відобразити взаємозв’язки між окремими сценаріями використання системи.

Розроблена діаграма прецедентів використання представлена на рис. 2.1.

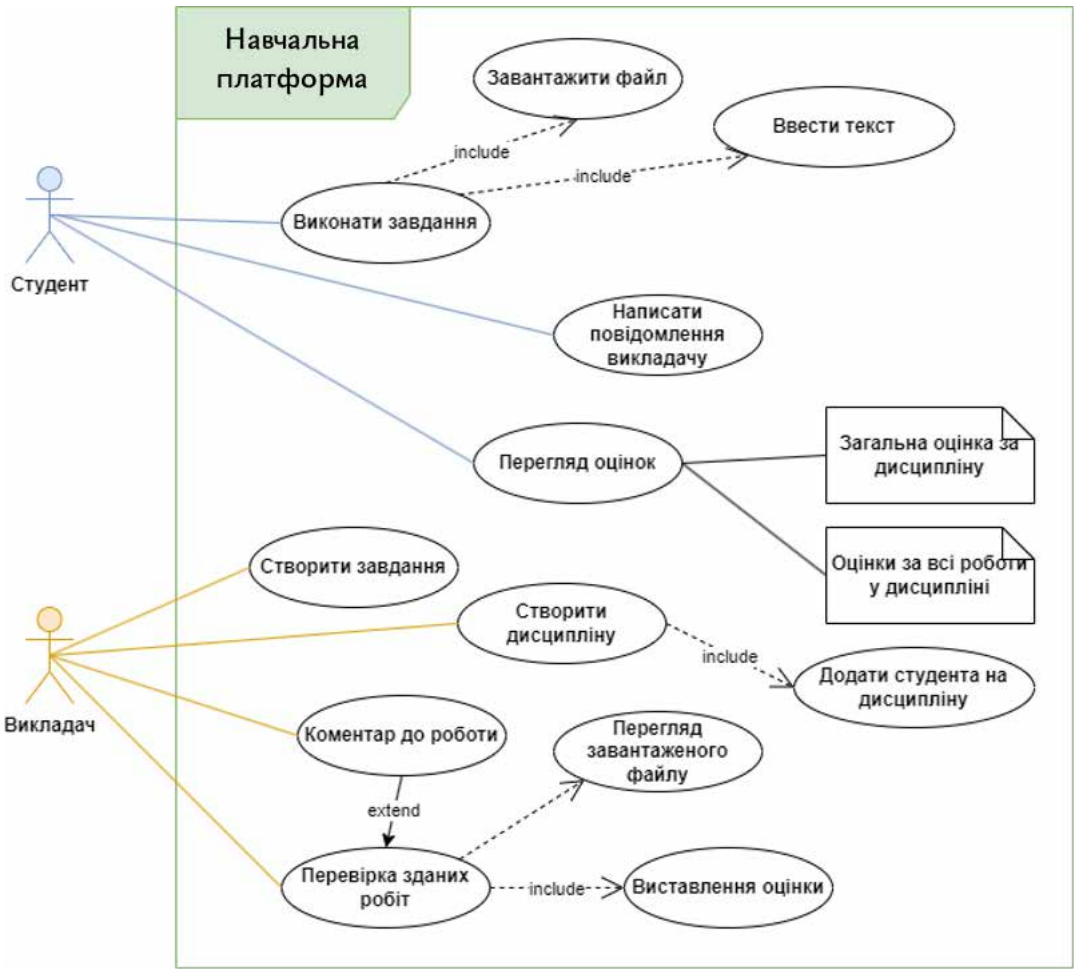

Рис. 2.1 Діаграма прецедентів

Створена діаграма прецедентів містить акторів:

- “Студент”;
- “Викладач”.

Актор «Студент» включає такі прецеденти:

- виконати завдання;
- завантажити файл;
- ввести текст;
- написати повідомлення викладачу;
- перегляд оцінок.

Актор «Викладач» включає такі прецеденти:

- створити завдання;
- створити дисципліну;
- додати здобувача вищої освіти до дисципліни;
- перевірка зданих робіт;
- виставлення оцінки;
- коментар до роботи;
- перегляд завантаженого файлу.

Прецеденти певним чином залежать одне від одного.

Прецедент “Виконати завдання” доповнюється іншими такими прецедентами як “Завантажити файл” та “Ввести текст”.

Прецедент “Перегляд оцінок” включає в себе такі анотації як “Загальна оцінка за дисципліну”, “Оцінки за всі роботи у дисципліни”.

Розглянемо детальніше вищеописані прецеденти.

Назва випадку використання: створити завдання

Актор: викладач

Передумови: викладач повинен бути авторизований у системі. Викладач повинен мати доступ хоча б до однієї навчальної дисципліни, у межах якої буде створюватися завдання.

Основний потік:

1. викладач переходить до розділу керування навчальними дисциплінами;
2. викладач обирає навчальну дисципліну, для якої необхідно створити нове завдання;
3. система відображає сторінку дисципліни із переліком наявних завдань та навчальних матеріалів;
4. викладач натискає кнопку «Створити завдання»;
5. система відкриває форму створення нового завдання;
6. викладач вводить назву завдання, опис, категорію роботи, дату завершення та максимальну кількість балів;
7. викладач за потреби додає додаткові файли або навчальні матеріали до завдання;
8. система перевіряє коректність введених даних;
9. викладач підтверджує створення завдання натисканням кнопки «Зберегти»;
10. система зберігає інформацію про завдання у базі даних;
11. система відображає повідомлення про успішне створення завдання;
12. здобувачі вищої освіти, які зареєстровані на дисципліну, отримують сповіщення про нове завдання.

Альтернативні потоки:

1. якщо викладач не заповнив обов’язкові поля форми, система відображає повідомлення про помилку та вимагає введення відсутніх даних. Після виправлення помилок викладач повторно надсилає форму;
2. якщо дата завершення завдання є меншою за поточну дату, система повідомляє про некоректність введених даних та не дозволяє створити завдання;
3. якщо викладач вирішує скасувати створення завдання, він натискає кнопку «Скасувати», після чого система повертає

користувача до сторінки дисципліни без збереження введеної інформації;

4. якщо під час збереження завдання виникає помилка з’єднання із сервером або базою даних, система відображає повідомлення про технічну помилку та пропонує повторити спробу пізніше.

У даному сценарії описано процес створення нового завдання у системі дистанційного навчання. Сценарій демонструє взаємодію між викладачем та програмним забезпеченням під час додавання нового навчального завдання до дисципліни, а також відображає механізми

перевірки введених даних, збереження інформації та сповіщення здобувачів вищої освіти про нове завдання.

Розглянемо інший сценарій.

Назва випадку використання: отримання рекомендацій навчальних матеріалів

Актор: здобувач вищої освіти

Передумови: здобувач вищої освіти повинен бути авторизований у системі. У системі повинні бути наявні результати навчальної діяльності користувача, необхідні для аналізу прогресу та формування рекомендацій.

Основний потік:

1. здобувач вищої освіти входить до системи дистанційного навчання;
2. користувач переходить до розділу перегляду власного навчального прогресу;
3. система аналізує результати оцінювання, активність користувача та історію виконання завдань;
4. система визначає теми або категорії навчальних матеріалів, які потребують додаткового опрацювання;
5. система формує персоналізований список рекомендованих навчальних матеріалів;
6. здобувач вищої освіти відкриває розділ рекомендацій;
7. система відображає перелік рекомендованих матеріалів із коротким описом;
8. користувач обирає необхідний навчальний матеріал;
9. система відкриває сторінку перегляду вибраного навчального матеріалу;
10. система фіксує факт перегляду рекомендованого матеріалу для подальшого аналізу активності користувача.

Альтернативні потоки:

1. якщо у системі недостатньо даних для аналізу прогресу користувача, система відображає повідомлення про неможливість формування рекомендацій на поточний момент;
2. якщо для певної дисципліни відсутні додаткові навчальні матеріали, система повідомляє користувача про відсутність доступних рекомендацій;
3. якщо під час формування рекомендацій виникає помилка обробки даних, система відображає повідомлення про технічну помилку та пропонує повторити спробу пізніше;
4. якщо користувач не обирає жодного рекомендованого матеріалу, система зберігає сформований список рекомендацій для подальшого перегляду.

У даному сценарії описано процес отримання персоналізованих рекомендацій навчальних матеріалів у системі дистанційного навчання. Сценарій демонструє механізм аналізу навчального прогресу здобувача вищої освіти, формування рекомендацій на основі результатів навчальної діяльності та взаємодію користувача із рекомендованим освітнім контентом.

2.2.2 Діаграма послідовності. Sequence-діаграма є одним із типів UML-діаграм, який використовується для моделювання динамічної поведінки програмного забезпечення та відображення послідовності взаємодії між об’єктами системи у процесі виконання певного сценарію. Основним призначенням sequence-діаграми є демонстрація порядку обміну повідомленнями між компонентами системи в часовій послідовності. Такий підхід дозволяє детально описати логіку функціонування програмного забезпечення та визначити механізми взаємодії між користувачами, серверною частиною, базою даних і окремими програмними модулями [10].

У межах розробки системи рекомендацій навчальних матеріалів на основі аналізу прогресу студента sequence-діаграми використовуються для моделювання процесів взаємодії користувачів із системою, обробки запитів,

збереження інформації та формування рекомендацій навчального контенту. Використання даного типу діаграм дозволяє наочно представити внутрішню логіку роботи програмного забезпечення та визначити порядок виконання окремих операцій у межах конкретного сценарію використання.

Основними елементами sequence-діаграми є актори, об’єкти системи, повідомлення та часові лінії. Актори представляють користувачів або зовнішні системи, які взаємодіють із програмним забезпеченням. Об’єкти системи відображають програмні компоненти, що беруть участь у виконанні певного процесу, наприклад вебінтерфейс, серверну логіку, модуль рекомендацій або базу даних. Повідомлення між об’єктами демонструють виклики функцій, передачу даних та результати виконання операцій.

У системі дистанційного навчання sequence-діаграми можуть використовуватися для опису таких процесів, як авторизація користувача, створення навчальної дисципліни, додавання завдання, завантаження студентської роботи, оцінювання результатів навчання та формування рекомендацій навчальних матеріалів. Наприклад, під час сценарію створення завдання sequence-діаграма відображає послідовність дій

викладача, передачу даних через вебінтерфейс, обробку інформації серверною частиною та збереження результату у базі даних.

Діаграма послідовності, представлена на Рис. 2.2, містить наступні об’єкти:

- “Студент”;
- “Викладач”;
- “Завдання”;
- “Оцінки”.

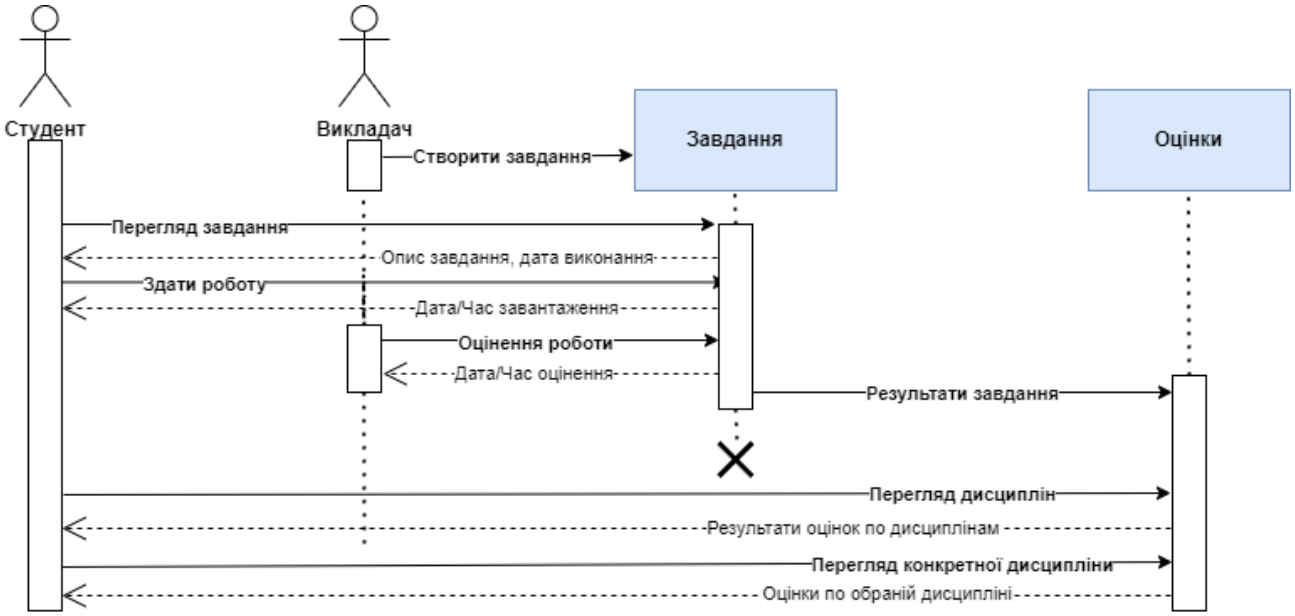

Рисунок 2.2 Діаграма послідовності

Об’єкт «Студент» надсилає запит до об’єкта “Завдання” для перегляду завдання та отримує у відповідь його опис і дату виконання. Після цього здобувач вищої освіти надсилає запит на подання виконаної роботи та отримує у відповідь дату і час завантаження.

Далі об’єкт «Викладач» надсилає запит до об’єкта “Завдання” для оцінювання роботи та отримує у відповідь дату і час оцінки. Після завершення цих дій об’єкт «Завдання» передає результати роботи до об’єкта

“Оцінки”. Об’єкт «Студент» надсилає запит до “Оцінки” на перегляд дисциплін та отримує результати оцінок по всіх дисциплінах.

Потім студент надсилає запит на перегляд конкретної дисципліни і отримує у відповідь оцінки по обраній дисципліні.

2.2.3 Діаграма активності. Activity-діаграма є одним із типів UML-діаграм, який використовується для моделювання бізнес-процесів, алгоритмів роботи системи та послідовності виконання окремих дій у межах програмного забезпечення. Основним призначенням activity-діаграми є графічне відображення логіки виконання процесів, умов переходу між окремими етапами та взаємозв’язків між діями користувачів і системи. Даний тип діаграм дозволяє детально описати порядок виконання операцій та забезпечує наочне представлення процесів функціонування програмного забезпечення [11].

У межах розробки системи рекомендацій навчальних матеріалів на основі аналізу прогресу студента activity-діаграми використовуються для моделювання основних процесів дистанційного навчання та взаємодії користувачів із системою. Використання activity-діаграм дозволяє формалізувати алгоритми роботи окремих функціональних модулів, визначити умови виконання певних дій та описати логіку переходу між етапами виконання процесів.

Основними елементами activity-діаграми є дії, переходи, умови, початкові та кінцеві стани процесу. Дії відображають окремі операції або функції, які виконуються користувачем чи системою. Переходи демонструють послідовність виконання дій, а умовні оператори дозволяють відобразити альтернативні варіанти розвитку процесу залежно від певних умов. Початковий стан визначає момент запуску процесу, тоді як кінцевий стан позначає завершення виконання сценарію.

У системі дистанційного навчання activity-діаграми можуть використовуватися для моделювання процесів авторизації користувача, створення навчальної дисципліни, додавання завдань, перевірки

студентських робіт, оцінювання результатів навчання та формування рекомендацій навчальних матеріалів. Наприклад, activity-діаграма процесу створення завдання дозволяє відобразити послідовність дій викладача під час введення даних, перевірки коректності інформації та збереження нового завдання у системі.

Особливе значення activity-діаграми мають для моделювання процесу формування персоналізованих рекомендацій навчального контенту. У межах такого сценарію діаграма може відображати етапи отримання інформації про результати навчання студента, аналіз оцінок та активності користувача, визначення слабких тем і формування списку рекомендованих матеріалів. Це дозволяє детально описати алгоритм роботи рекомендаційного модуля та визначити послідовність обробки даних у системі.

Використання activity-діаграм сприяє кращому розумінню логіки функціонування програмного забезпечення та дозволяє виявляти потенційні помилки або неефективні етапи процесів ще на етапі проєктування системи. Крім того, activity-діаграми є корисними під час документування програмного забезпечення, оскільки забезпечують наочне представлення

алгоритмів роботи системи для розробників, аналітиків та інших учасників процесу розробки.

Розроблена діаграма активності представлена на рис. 2.3

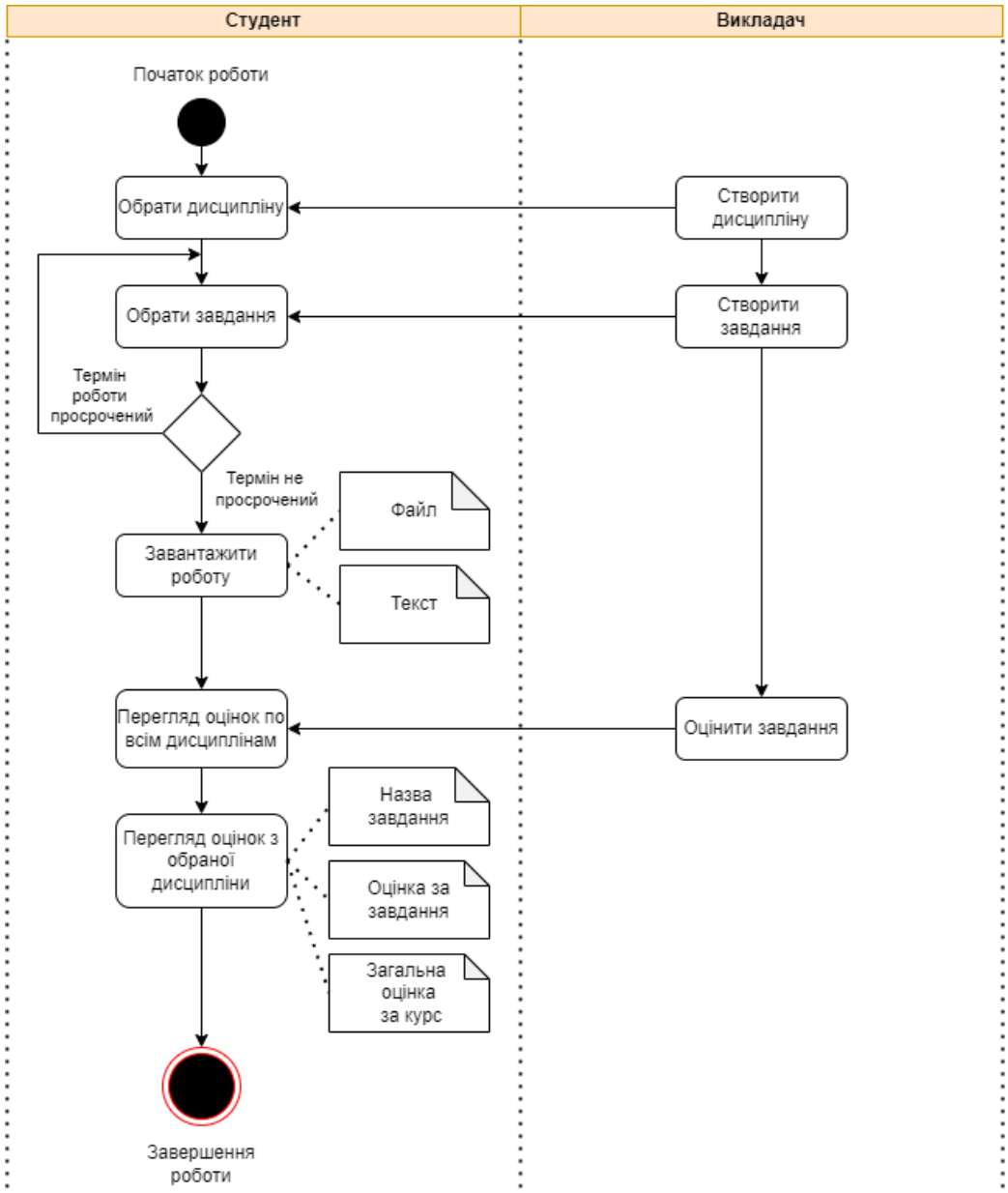

Рис. 2.3 Діаграма активності

2.3 Абстракції предметної області

Абстракції предметної області є важливим елементом процесу моделювання програмного забезпечення, оскільки вони дозволяють формалізувати основні сутності системи, визначити їх характеристики та

взаємозв’язки між ними. Використання абстракцій забезпечує спрощене представлення складних процесів предметної області та створює основу для подальшого проєктування архітектури програмного забезпечення, структури бази даних і функціональних компонентів системи [12].

У межах розробки системи рекомендацій навчальних матеріалів на основі аналізу прогресу студента предметна область охоплює процеси дистанційного навчання, взаємодії між учасниками освітнього процесу, управління навчальним контентом, контролю успішності та персоналізації освітнього середовища. Для формалізації цих процесів необхідно виділити ключові абстракції, які описують основні об’єкти системи та їх функціональне призначення.

Однією з основних абстракцій предметної області є користувач. Дана сутність представляє учасника освітнього процесу, який взаємодіє із системою. У межах програмного забезпечення користувачі поділяються на викладачів та здобувачів вищої освіти. Викладач відповідає за створення та адміністрування навчальних дисциплін, додавання матеріалів, створення завдань та оцінювання результатів навчання. Здобувач вищої освіти отримує доступ до навчального контенту, виконує завдання, переглядає оцінки та отримує персоналізовані рекомендації навчальних матеріалів.

Наступною важливою абстракцією є навчальна дисципліна. Дана сутність використовується для організації освітнього процесу та об’єднання навчальних матеріалів, завдань і учасників у межах окремого курсу. Навчальна дисципліна містить інформацію про назву курсу, опис, викладача та список здобувачів вищої освіти, які беруть участь у навчальному процесі.

Важливе місце у структурі предметної області займає абстракція завдання. Завдання є одним із ключових елементів контролю знань студентів та використовується для оцінювання результатів навчальної діяльності. У системі можуть існувати різні типи завдань, зокрема лабораторні роботи, практичні завдання, модульні контрольні роботи або тестування. Кожне

завдання характеризується назвою, описом, категорією, терміном виконання та максимальною кількістю балів.

Для забезпечення доступу до освітнього контенту використовується абстракція навчального матеріалу. Дана сутність представляє інформаційний ресурс, який може бути використаний студентом у процесі навчання. Навчальні матеріали можуть бути представлені у вигляді текстових документів, презентацій, відеофайлів, посилань на зовнішні ресурси або інших форматів електронного контенту. Матеріали пов’язуються із конкретними дисциплінами та використовуються як основне джерело навчальної інформації.

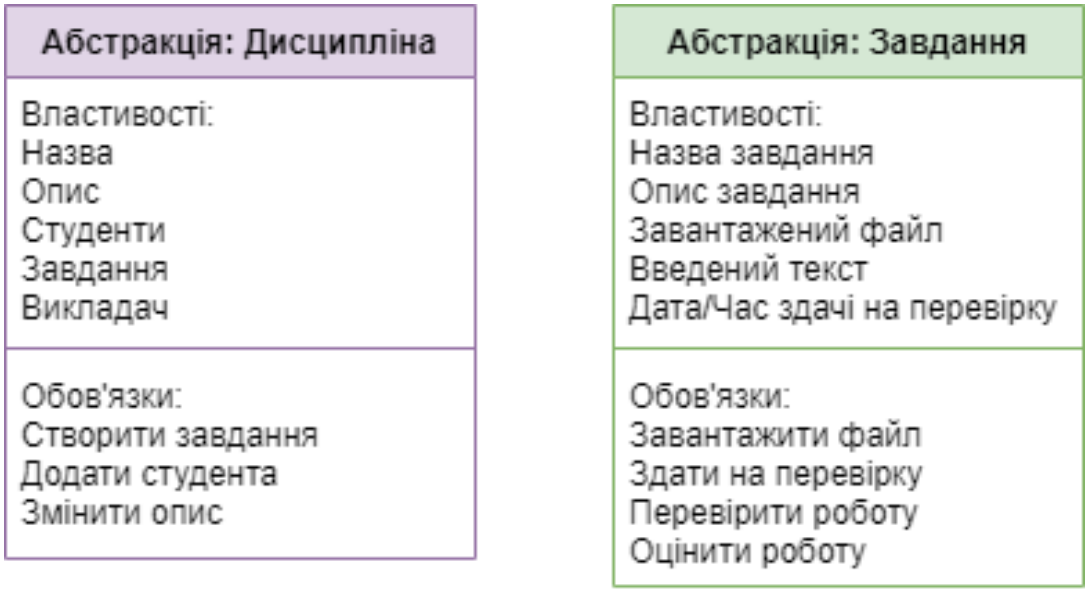

Рис. 2.4 Абстракції предметної області

Ці абстракції предметної області представляють основні сутності процесу оренди та їхні характеристики.

2.4 Діаграма класів

Діаграма класів є одним із основних інструментів об’єктноорієнтованого моделювання програмного забезпечення та використовується для відображення структури системи, її основних класів, атрибутів, методів та взаємозв’язків між окремими компонентами. Даний тип

UML-діаграм дозволяє формалізувати архітектуру програмного забезпечення, визначити структуру даних та забезпечити основу для подальшої реалізації програмної системи [15].

У межах розробки системи рекомендацій навчальних матеріалів на основі аналізу прогресу студента діаграма класів використовується для моделювання основних сутностей предметної області та логічних зв’язків між ними. Побудова діаграми класів дозволяє визначити структуру програмного забезпечення, організувати взаємодію між об’єктами системи та забезпечити правильне проєктування бази даних і серверної логіки вебзастосунку.

Основою діаграми класів є клас користувача, який представляє учасника освітнього процесу. Даний клас містить інформацію про ім’я користувача, електронну пошту, пароль, роль та інші персональні дані. У межах системи користувач може виконувати роль викладача або здобувача вищої освіти. Від ролі користувача залежить набір функціональних можливостей, доступних у системі.

Одним із центральних класів предметної області є клас навчальної дисципліни. Даний клас використовується для організації освітнього процесу та містить інформацію про назву дисципліни, опис, викладача та список учасників курсу. Клас дисципліни має зв’язки із класами завдань та навчальних матеріалів, оскільки кожна дисципліна може містити декілька освітніх ресурсів та практичних робіт.

Клас завдання використовується для представлення навчальних робіт, які виконуються студентами у процесі навчання. Даний клас містить атрибути, що описують назву завдання, категорію роботи, опис, дедлайн та максимальну кількість балів. Завдання пов’язується із конкретною дисципліною та може мати декілька подань робіт від різних студентів.

Для організації освітнього контенту використовується клас навчального матеріалу. Даний клас містить інформацію про назву матеріалу, його тип, опис та файл або посилання на ресурс. Навчальні матеріали

пов’язуються із відповідними дисциплінами та використовуються студентами у процесі навчання.

Важливим елементом діаграми класів є клас подання роботи. Даний клас використовується для збереження інформації про виконані студентами завдання. Він містить посилання на студента, завдання, файл роботи, дату подання та статус перевірки. Подання роботи дозволяє реалізувати механізм дистанційної перевірки результатів навчальної діяльності.

Клас оцінки використовується для представлення результатів оцінювання студентських робіт. Даний клас містить інформацію про отриманий бал, коментар викладача та дату оцінювання. Оцінка пов’язується із конкретним студентом та відповідним завданням, результати якого були перевірені викладачем.

Особливе місце у структурі програмного забезпечення займає клас рекомендації. Даний клас використовується для реалізації механізму персоналізованих рекомендацій навчальних матеріалів. Клас містить інформацію про користувача, рекомендований навчальний матеріал, рівень релевантності рекомендації та причину її формування. Рекомендації створюються на основі аналізу результатів навчальної діяльності студента та використовуються для підтримки адаптивного навчання.

Між класами системи існують різні типи зв’язків, серед яких асоціації, агрегації та залежності. Наприклад, один викладач може створювати декілька дисциплін, одна дисципліна може містити багато завдань та навчальних матеріалів, а один студент може подавати декілька виконаних робіт. Такі зв’язки дозволяють забезпечити цілісність структури програмного забезпечення та логічну організацію взаємодії між компонентами системи.

Діаграма класів також використовується як основа для проєктування структури бази даних. Кожен клас предметної області може відповідати окремій таблиці бази даних, а зв’язки між класами реалізуються за допомогою первинних та зовнішніх ключів. Це дозволяє забезпечити узгоджене зберігання інформації та підтримку цілісності даних у системі.

Для цього проєкту було створено спеціальну діаграму класів з використанням об’єктно-орієнтованого підходу (рис. 2.5).

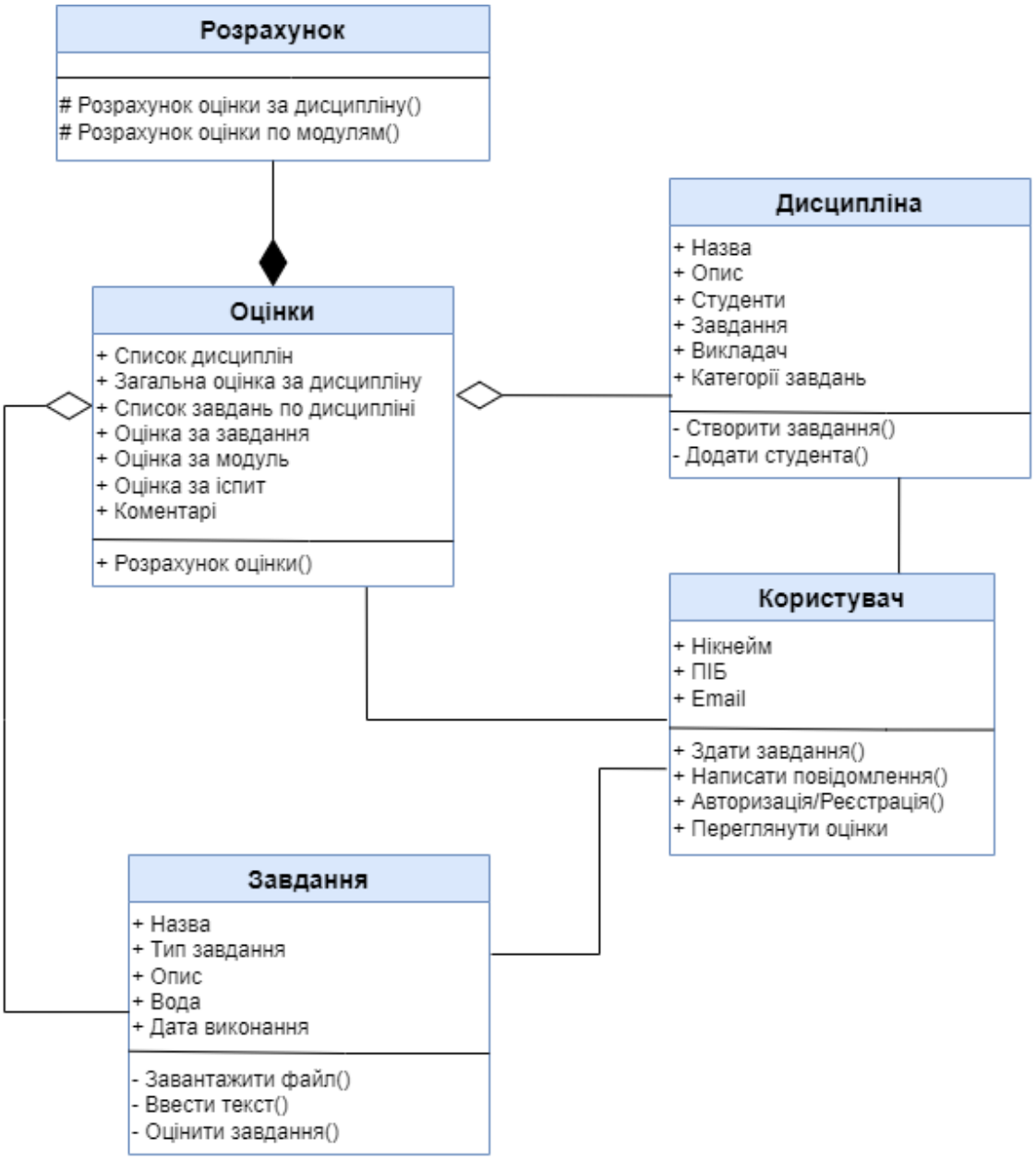

Рис. 2.5 Діаграма класів

3 ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 Логічна модель даних

Логічна модель даних є важливим етапом проєктування програмної системи, оскільки саме вона визначає структуру зберігання інформації, взаємозв’язки між окремими сутностями та принципи організації даних у межах програмного забезпечення. Побудова логічної моделі дозволяє формалізувати структуру бази даних, забезпечити цілісність інформації та створити основу для подальшої реалізації фізичної моделі даних у системі керування базами даних.

У межах розробки системи рекомендацій навчальних матеріалів на основі аналізу прогресу студента логічна модель даних повинна забезпечувати зберігання інформації про користувачів, навчальні дисципліни, завдання, навчальні матеріали, результати оцінювання, подані роботи та сформовані рекомендації. Крім того, структура даних повинна підтримувати механізми аналізу успішності студентів та формування персоналізованих рекомендацій навчального контенту.

Основною сутністю логічної моделі є користувач. Дана сутність містить інформацію про зареєстрованих користувачів системи, зокрема ім’я, електронну пошту, пароль, роль та інші персональні дані. Користувачі можуть мати роль викладача або здобувача вищої освіти. Для забезпечення безпеки системи інформація про паролі повинна зберігатися у зашифрованому вигляді.

Однією з центральних сутностей моделі є навчальна дисципліна. Дана сутність використовується для організації освітнього процесу та містить інформацію про назву дисципліни, опис, дату створення та викладача, який відповідає за проведення курсу. Оскільки одна дисципліна може містити багато студентів, а один студент може брати участь у декількох дисциплінах одночасно, між сутностями користувача та дисципліни реалізовується зв’язок типу «багато до багатьох».

Для реалізації такого зв’язку використовується проміжна сутність участі у дисципліні. Дана сутність містить інформацію про користувача та дисципліну, до якої він належить. Такий підхід дозволяє забезпечити гнучке управління учасниками навчальних курсів та спрощує організацію доступу до освітнього контенту.

Важливим елементом логічної моделі є сутність завдання. Вона містить інформацію про назву роботи, опис, категорію завдання, дату завершення, максимальну кількість балів та дисципліну, до якої належить завдання. Кожне завдання створюється викладачем і використовується для контролю знань студентів.

Для зберігання освітнього контенту використовується сутність навчального матеріалу. Дана сутність містить назву матеріалу, тип ресурсу, опис, файл або посилання на матеріал та дисципліну, до якої належить ресурс. Навчальні матеріали можуть бути представлені у вигляді текстових документів, презентацій, відеофайлів або зовнішніх посилань.

Сутність подання роботи використовується для організації процесу дистанційної перевірки завдань. Вона містить інформацію про файл виконаної роботи, дату подання, статус перевірки, користувача та завдання, до якого належить подання. Завдяки цій сутності система забезпечує централізоване зберігання результатів виконання студентських робіт.

Для реалізації механізму оцінювання використовується сутність оцінки. Вона містить інформацію про кількість отриманих балів, коментар викладача, дату оцінювання та зв’язок із відповідним поданням роботи. Дані оцінювання використовуються не лише для відображення результатів навчальної діяльності, а й для аналізу успішності студентів та формування рекомендацій.

Однією з ключових сутностей системи є рекомендація навчального матеріалу. Дана сутність використовується для реалізації механізму персоналізації навчального процесу. Вона містить інформацію про студента, рекомендований навчальний матеріал, рівень релевантності рекомендації,

причину її формування та дату створення рекомендації. Формування рекомендацій здійснюється на основі аналізу результатів навчальної діяльності користувача, його активності та успішності виконання завдань.

У логічній моделі даних важливу роль відіграють зв’язки між сутностями. Наприклад, одна дисципліна може містити багато завдань та навчальних матеріалів, тоді як одне завдання може мати багато подань робіт від різних студентів. Оцінки пов’язуються із конкретними поданнями робіт, а рекомендації — із користувачами та навчальними матеріалами. Така структура дозволяє забезпечити цілісність даних та ефективну організацію інформації у межах програмної системи.

Під час проєктування логічної моделі даних також враховуються вимоги до нормалізації структури бази даних. Використання нормалізації дозволяє уникнути дублювання інформації, зменшити ризик виникнення помилок під час оновлення даних та підвищити ефективність роботи системи керування базами даних. Крім того, логічна модель повинна забезпечувати можливість масштабування системи та подальшого розширення функціональності програмного забезпечення.

Логічна модель системи представлена на рисунку 3.1.

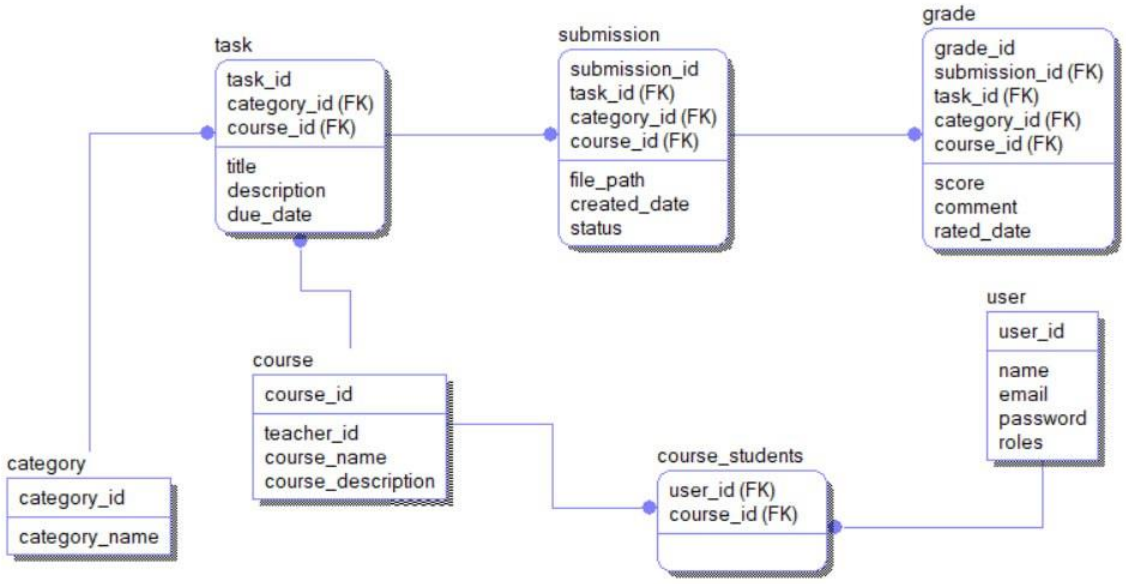

Рис. 3.1 ER-діаграма “estudy_db”

Логічна модель складається з таких сутностей:

User:

- id: int - Унікальний ідентифікатор користувача;
- name: string - Ім'я користувача;
- email: string - Email користувача;
- password: string - Хешований пароль користувача;
- roles: array - Ролі користувача (наприклад, ROLE_ADMIN, ROLE_STUDENT).

Course:

- id: int - Унікальний ідентифікатор дисципліни;
- courseName: string - Назва дисципліни;
- teacher: User – викладач, який веде дисципліну.

CourseStudents:

- id: int - Унікальний ідентифікатор запису про здобувача вищої освіти в дисципліні;
- course_id: Course – Дисципліна;
- student_id: User – Здобувач вищої освіти.

Task:

- id: int - Унікальний ідентифікатор завдання;
- title: string - Назва завдання;
- category: string - Категорія завдання;
- dueDate: DateTime - Дата, до якої необхідно здати завдання;
- course: Course.

Submission:

- id: int - Унікальний ідентифікатор виправки завдання;
- filePath: string - завантажений файл завдання;
- student: User – Здобувач вищої освіти;
- task: Task – Завдання;
- status: int - Статус відповіді.

Grade:

- id: int - Унікальний ідентифікатор оцінки;
- score: float - Оцінка за виконання завдання;
- comment: string – коментар;
- submission: Submission - Зв'язок з відправкою завдання.

У сутності Course є зв'язок з User через атрибут teacher, що вказує на викладача, який веде цю дисципліну. Сутність CourseStudents пов'язує студента освіти та дисципліни: кожен здобувач вищої освіти (посилання на User) може бути доданий у кілька дисциплін, а кожна дисципліна може містити безліч студентів.

Кожна Course може мати кілька Task, що означає завдання, що стосуються цієї дисципліни. Кожне завдання може надіслано кількома студентами, що здійснюється через сутність Submission. Сутність Submission пов'язує студентів (посилання на User) та завдання (посилання на Task) та зберігає інформацію про статус відправки та файл, завантажений здобувач вищої освітиом.

Сутність Grade пов'язує оцінки з відправками завдань, вказуючи на конкретну Submission та надаючи оцінку та коментар викладача. Це створює повний ланцюжок від викладача до студента через завдання та оцінки.

3.2 Вибір системи управління базою даних та її реалізація

Система управління базами даних є одним із ключових компонентів сучасного програмного забезпечення, оскільки саме вона забезпечує зберігання, обробку, оновлення та керування інформацією у межах інформаційної системи. Використання систем управління базами даних дозволяє організувати централізоване зберігання великих обсягів структурованих даних, забезпечити швидкий доступ до інформації та підтримувати цілісність і безпеку даних під час роботи програмного забезпечення [16].

Система управління базами даних, або СУБД, являє собою спеціалізоване програмне забезпечення, призначене для створення, адміністрування та підтримки баз даних. Основним призначенням СУБД є забезпечення ефективної взаємодії між програмним застосунком та даними, які зберігаються у базі даних. За допомогою СУБД реалізується виконання операцій створення, пошуку, редагування та видалення інформації, а також контроль доступу користувачів до даних і забезпечення їх захисту.

У сучасних інформаційних системах СУБД виконує надзвичайно важливу роль, оскільки дозволяє автоматизувати процеси управління даними та забезпечити стабільну роботу програмного забезпечення навіть при значному навантаженні. Крім того, використання систем управління базами даних дозволяє уникнути дублювання інформації, забезпечити узгодженість даних та реалізувати механізми резервного копіювання і відновлення інформації у разі виникнення помилок або технічних збоїв [17].

Для розробки системи рекомендацій навчальних матеріалів на основі аналізу прогресу студента було обрано реляційну систему управління базами даних MySQL. Дана СУБД є однією з найпоширеніших систем управління базами даних у сфері веброзробки та активно використовується під час створення веборієнтованих інформаційних систем різного рівня складності. Популярність MySQL обумовлена високою продуктивністю, надійністю, відкритістю вихідного коду та широкими можливостями інтеграції із сучасними технологіями веброзробки.

Вибір MySQL для реалізації програмної системи зумовлений рядом переваг, які є важливими для функціонування системи дистанційного навчання. Однією з основних переваг є підтримка реляційної моделі даних, що дозволяє організувати структуроване зберігання інформації про користувачів, навчальні дисципліни, завдання, результати оцінювання та рекомендації навчальних матеріалів. Використання реляційної моделі забезпечує можливість ефективного встановлення зв’язків між сутностями системи та підтримки цілісності даних.

Важливою перевагою MySQL є висока швидкість обробки запитів та стабільність роботи навіть при одночасній роботі великої кількості користувачів. Для системи дистанційного навчання це є особливо важливим, оскільки програмне забезпечення повинно забезпечувати швидкий доступ до навчальних матеріалів, результатів оцінювання та рекомендаційного контенту без значних затримок.

Крім того, MySQL підтримує механізми захисту даних та розмежування прав доступу користувачів. Це дозволяє забезпечити безпечне зберігання персональної інформації студентів та викладачів, а також обмежити доступ до окремих функціональних можливостей системи відповідно до ролі користувача. Підтримка транзакцій та механізмів резервного копіювання також підвищує надійність роботи програмного забезпечення та зменшує ризик втрати даних [18].

У межах реалізації програмної системи база даних використовується для централізованого зберігання всієї інформації, необхідної для функціонування вебзастосунку. У базі даних зберігаються облікові записи користувачів, інформація про навчальні дисципліни, матеріали, завдання, результати оцінювання, подані роботи та сформовані рекомендації навчальних матеріалів. Взаємодія між серверною частиною програмного забезпечення та базою даних здійснюється за допомогою SQL-запитів та ORM-механізмів фреймворку Symfony.

Під час реалізації структури бази даних особлива увага приділяється нормалізації таблиць та забезпеченню цілісності інформації. Для встановлення зв’язків між сутностями використовуються первинні та зовнішні ключі, що дозволяє підтримувати узгодженість даних та уникати дублювання інформації. Такий підхід сприяє підвищенню ефективності роботи системи та спрощує подальше супроводження програмного забезпечення.

Система управління базами даних є важливою складовою системи рекомендацій навчальних матеріалів на основі аналізу прогресу студента. Використання СУБД MySQL дозволяє забезпечити ефективне зберігання та

обробку даних, підтримку цілісності інформації, безпечний доступ користувачів до системи та стабільне функціонування вебзастосунку в умовах реального використання.

Таким чином було створено базу даних в середовищі MySQL Workbench (Рис. 3.2).

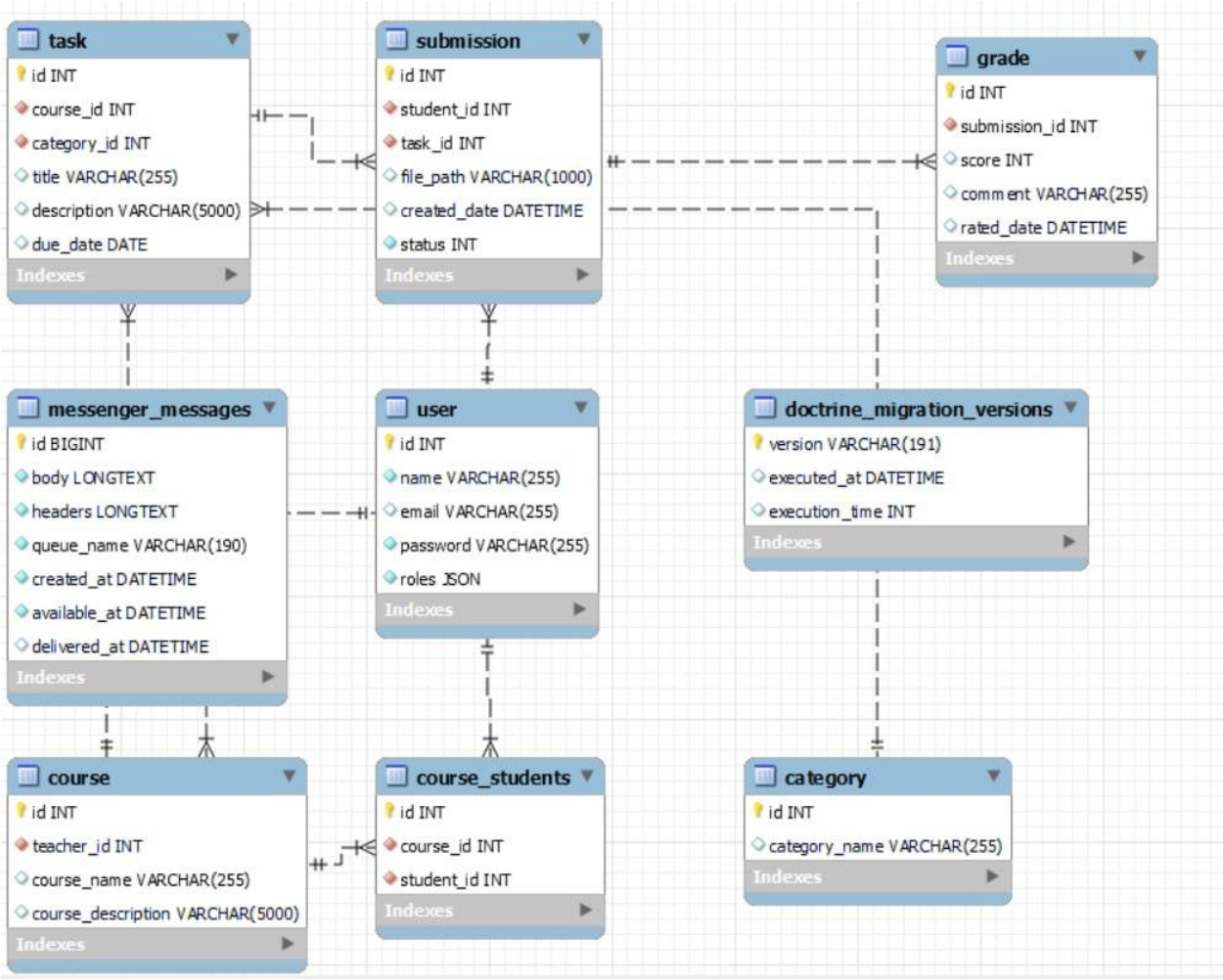

Рис. 3.2 База даних

3.3 Архітектура програмного забезпечення

Архітектура програмного забезпечення визначає загальну структуру системи, принципи взаємодії між її компонентами та способи організації функціональних модулів. Вона є ключовим етапом проєктування інформаційної системи, оскільки безпосередньо впливає на масштабованість,

продуктивність, зручність супроводу та можливість подальшого розвитку програмного продукту.

У сучасній програмній інженерії існує декілька базових архітектурних підходів, які застосовуються залежно від вимог до системи та її складності. Однією з найпростіших є монолітна архітектура, за якої всі компоненти програмного забезпечення об’єднані в єдиний програмний модуль. У такій архітектурі інтерфейс користувача, бізнес-логіка та доступ до даних реалізуються в межах одного застосунку. Перевагами монолітного підходу є простота розробки та розгортання, однак недоліками є складність масштабування та обмежена гнучкість при подальшому розвитку системи.

Іншим поширеним підходом є багаторівнева (layered) архітектура, яка передбачає розділення системи на декілька логічних рівнів. Найчастіше виділяють рівень представлення (інтерфейс користувача), рівень бізнес-логіки та рівень доступу до даних. Такий підхід дозволяє чітко розділити відповідальність між компонентами системи, спрощує тестування та супровід програмного забезпечення, а також підвищує його структурованість.

Також широко використовується клієнт-серверна архітектура, яка передбачає поділ системи на дві основні частини: клієнтську та серверну. Клієнтська частина відповідає за взаємодію з користувачем, тоді як серверна частина обробляє запити, виконує бізнес-логіку та взаємодіє з базою даних. Дана архітектура є базовою для більшості вебзастосунків і забезпечує ефективну організацію взаємодії між користувачем і системою через мережу.

Більш сучасним підходом є мікросервісна архітектура, яка передбачає розбиття системи на набір незалежних сервісів, кожен з яких відповідає за окрему функціональність. Мікросервіси взаємодіють між собою через API, що дозволяє підвищити масштабованість системи та спростити незалежне оновлення окремих компонентів. Однак такий підхід є складнішим у реалізації та вимагає додаткових інструментів для оркестрації та моніторингу [20].

Для розроблюваної системи рекомендацій навчальних матеріалів на основі аналізу прогресу студента найбільш доцільним є використання

багаторівневої клієнт-серверної архітектури. Такий підхід дозволяє ефективно розділити відповідальність між компонентами системи, забезпечити гнучкість у розвитку програмного забезпечення та спростити його супровід.

У запропонованій архітектурі виділяються три основні рівні. Перший рівень — рівень представлення, який реалізується у вигляді вебінтерфейсу та відповідає за взаємодію з користувачем. Другий рівень — рівень бізнес-логіки, який реалізується за допомогою серверної частини на PHP із використанням фреймворку Symfony. Він забезпечує обробку запитів, реалізацію навчальної логіки та формування рекомендацій. Третій рівень — рівень доступу до даних, який реалізується через систему управління базами даних MySQL і забезпечує зберігання та обробку інформації.

Такий архітектурний підхід дозволяє забезпечити масштабованість системи, оскільки кожен рівень може розвиватися незалежно. Крім того, розділення логіки сприяє підвищенню безпеки, оскільки доступ до бази даних здійснюється виключно через серверний рівень. Це також спрощує тестування та подальшу модернізацію програмного забезпечення.

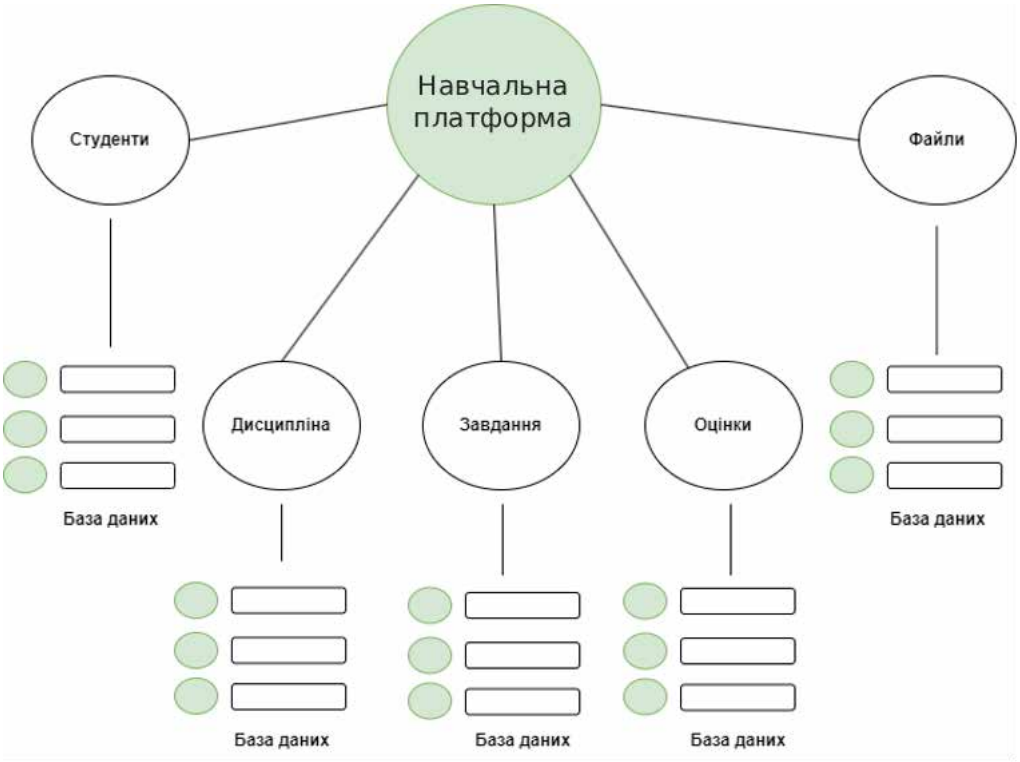

Рис. 3.3 Багатошарова архітектура

З рисунка видно, що наша система побудована на 3-рівневій багатошаровій архітектурі, яка складається з:

1. презентаційного рівня (інтерфейс користувача) – це екран комп'ютера або мобільного пристрою, через який користувач взаємодіє із системою;
2. бізнес-рівня (логіка програми) – основний функціонал системи, з яким працює користувач. На цьому рівні він може створювати дисципліни та завдання, здавати роботи на перевірку, оцінювати роботи та переглядати оцінки;
3. рівня бази даних – тут зберігається вся інформація системи, включаючи дані користувачів, завдання, оцінки та інші сутності.

Інформація у системі поступово переходить від одного рівня до наступного, забезпечуючи логічний і структурований потік даних.

3.4 Організаційна структура програмного забезпечення

3.4.1 Діаграма пакетів. Діаграма пакетів є одним із типів UML-діаграм, що використовується для відображення логічної структури програмної системи на високому рівні абстракції. Вона дозволяє згрупувати класи, модулі та компоненти системи у пакети та показати взаємозв'язки між ними. Основною метою діаграми пакетів є спрощення сприйняття складної архітектури програмного забезпечення шляхом її декомпозиції на логічні частини.

У межах розробки системи рекомендацій навчальних матеріалів на основі аналізу прогресу студента діаграма пакетів використовується для структурування серверної та клієнтської частин веб-застосунку, а також для відображення взаємодії між основними функціональними модулями. Такий підхід дозволяє чітко визначити межі відповідальності кожного пакету та забезпечити модульність архітектури програмного забезпечення.

Основними пакетами системи є модуль користувачів, навчальних дисциплін, завдань, навчальних матеріалів, оцінювання та рекомендацій. Кожен із цих пакетів виконує окрему функціональну роль у межах системи та містить набір класів і компонентів, які реалізують відповідну бізнес-логіку. Наприклад, пакет користувачів відповідає за реєстрацію, авторизацію та управління профілями користувачів, тоді як пакет дисциплін забезпечує створення та адміністрування навчальних курсів.

Пакет завдань містить функціональність, пов’язану зі створенням навчальних завдань, їх редагуванням, зберіганням та перевіркою виконаних робіт. Пакет навчальних матеріалів забезпечує роботу з освітнім контентом, включаючи завантаження, зберігання та доступ до навчальних ресурсів. Окремий пакет оцінювання відповідає за обробку результатів виконання завдань, збереження оцінок та коментарів викладача.

Особливе місце займає пакет рекомендацій, який реалізує механізм аналізу навчального прогресу студентів та формування персоналізованих навчальних матеріалів. Даний пакет взаємодіє з іншими модулями системи, оскільки використовує дані про оцінки, активність користувачів та виконані завдання для побудови рекомендаційної моделі.

Діаграма пакетів також відображає взаємозв’язки між модулями системи. Наприклад, пакет оцінювання залежить від пакета завдань та користувачів, оскільки оцінка формується на основі виконаних робіт студентів. У свою чергу, пакет рекомендацій використовує дані з пакетів оцінювання та навчальних матеріалів для формування індивідуальних рекомендацій.

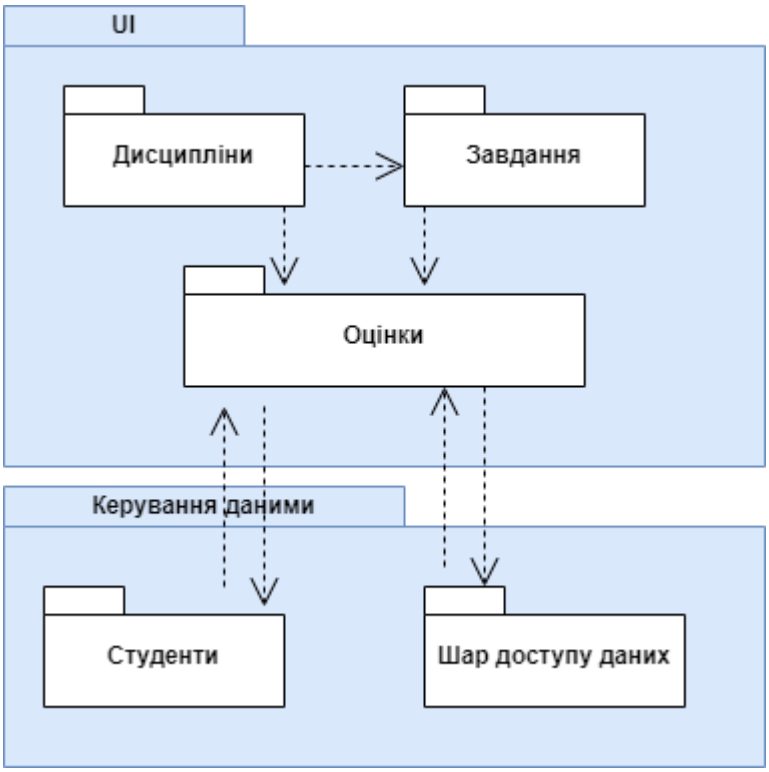

Рис. 3.4 Діаграма пакетів

Для нашої інформаційної системи пакетна організація допомагає структурувати програмне забезпечення відповідно до функціональних модулів, таких як користувачі, дисципліни, завдання та оцінки. Такий підхід забезпечує чіткий поділ обов'язків між компонентами, підвищує модульність і полегшує майбутнє розширення системи. Крім того, наявність зрозумілої пакетної структури полегшує командну роботу, оскільки кожен розробник може працювати над окремим пакетом, не порушуючи цілісність інших модулів.

Таким чином, діаграми пакетів не лише покращують розуміння архітектури системи, але й сприяють ефективній розробці, підтримці та масштабуванню програмного забезпечення.

3.5 Вибір інструментарію для створення програмного забезпечення

Вибір технологічного стеку є одним із ключових етапів проєктування програмної системи, оскільки від нього залежить продуктивність,

масштабованість, безпека та зручність подальшого супроводу розроблюваного вебзастосунку. У межах розробки системи рекомендацій навчальних матеріалів на основі аналізу прогресу студента було обрано набір сучасних технологій, які забезпечують ефективну реалізацію як серверної, так і клієнтської частин системи, а також підтримку стабільної роботи програмного забезпечення в умовах реального використання.

Для реалізації серверної частини програмного забезпечення використовується мова програмування PHP. Дана мова є однією з найпоширеніших у сфері веброзробки та широко застосовується для створення динамічних вебзастосунків. PHP забезпечує високу продуктивність, простоту інтеграції з базами даних та підтримку великої кількості бібліотек і фреймворків, що робить її доцільною для реалізації освітньої платформи.

У якості основного фреймворку для розробки серверної частини використовується Symfony. Даний фреймворк забезпечує структурований підхід до розробки програмного забезпечення, підтримує принципи MVC-архітектури та дозволяє створювати масштабовані й легко підтримувані вебзастосунки. Використання Symfony також спрощує реалізацію бізнес-логіки системи, роботи з базою даних, маршрутизації запитів та управління користувачами.

Для зберігання та обробки даних використовується система управління базами даних MySQL. Вона забезпечує надійне реляційне зберігання інформації, підтримку складних SQL-запитів та високу швидкість обробки даних. MySQL є оптимальним вибором для системи дистанційного навчання, оскільки дозволяє ефективно організувати зберігання інформації про користувачів, навчальні дисципліни, завдання, оцінки та рекомендації [21].

Для контейнеризації та спрощення процесу розгортання програмного забезпечення використовується Docker. Дана технологія дозволяє створювати ізольовані середовища для роботи застосунку, що забезпечує однакову поведінку системи на різних етапах розробки та експлуатації. Використання

Docker спрощує налаштування серверного середовища, бази даних та залежностей проєкту, а також підвищує стабільність роботи системи.

Клієнтська частина вебзастосунку реалізується з використанням HTML, CSS та JavaScript. HTML використовується для структурування вебсторінок, CSS — для оформлення та візуального стилю інтерфейсу, а JavaScript — для забезпечення інтерактивності та динамічної взаємодії користувача з системою. Такий набір технологій дозволяє створити функціональний та адаптивний вебінтерфейс.

Для покращення дизайну та забезпечення адаптивності інтерфейсу використовується CSS-фреймворк Bootstrap. Він надає набір готових компонентів, стилів і сіткову систему, що дозволяє швидко створювати сучасний та зручний користувацький інтерфейс. Використання Bootstrap забезпечує коректне відображення системи на різних пристроях, включаючи персональні комп’ютери, планшети та мобільні телефони.

Таким чином, обраний інструментарій — PHP, Symfony, MySQL, Docker, HTML, CSS, JavaScript та Bootstrap — забезпечує комплексну реалізацію програмної системи, охоплюючи як серверну, так і клієнтську частини. Поєднання цих технологій дозволяє створити надійний, масштабований та зручний у використанні вебзастосунок для організації дистанційного навчання та формування персоналізованих рекомендацій навчальних матеріалів.

4 ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЯ СИСТЕМИ

4.1 Вимоги до апаратного та програмного забезпечення

Для забезпечення стабільної та ефективної роботи програмного забезпечення системи рекомендацій навчальних матеріалів на основі аналізу прогресу студента необхідно визначити мінімальні та рекомендовані вимоги до апаратного та програмного середовища. Дотримання цих вимог безпосередньо впливає на продуктивність системи, швидкість обробки запитів, а також загальний користувацький досвід під час роботи з вебзастосунком.

З точки зору апаратного забезпечення, для серверної частини системи визначаються наступні мінімальні вимоги:

- процесор із тактовою частотою не нижче 2.0 GHz;
- оперативна пам’ять обсягом від 4 GB;
- дисковий простір не менше 20 GB для зберігання програмного забезпечення, бази даних та користувацьких файлів;
- стабільне підключення до мережі Інтернет для забезпечення доступу користувачів до системи.

Для забезпечення більш стабільної роботи системи в умовах високого навантаження рекомендуються такі параметри серверного обладнання:

- багатоядерний процесор середнього або високого класу;
- оперативна пам’ять обсягом 8 GB і більше;
- використання SSD-накопичувача для підвищення швидкості обробки даних;
- резервне копіювання даних на окремий носій або хмарне сховище.

Клієнтська частина системи не потребує значних апаратних ресурсів, однак для комфортної роботи користувача рекомендується наявність:

- персонального комп’ютера, ноутбука, планшета або смартфона;
- стабільного підключення до Інтернету;
- сучасного веббраузера останніх версій.

Щодо програмного забезпечення серверної частини, мінімальні вимоги включають:

- операційну систему сімейства Linux або Windows Server;
- інтерпретатор PHP версії не нижче 8.0;
- встановлену систему управління базами даних MySQL;
- налаштоване середовище для виконання вебзастосунку.

Додатково рекомендується використання контейнеризації за допомогою Docker для спрощення розгортання та забезпечення стабільності роботи системи. Це дозволяє створювати ізольовані середовища виконання та забезпечує однакову поведінку програмного забезпечення на різних етапах розробки та експлуатації.

Клієнтська частина системи повинна підтримувати сучасні вебтехнології, зокрема HTML5, CSS3 та JavaScript. Для коректної роботи інтерфейсу користувача рекомендується використання сучасних браузерів, таких як Google Chrome, Mozilla Firefox, Microsoft Edge або Safari останніх версій.

Також було зроблено діаграму розгортання для візуального огляду деплою системи (Рис. 4.1).

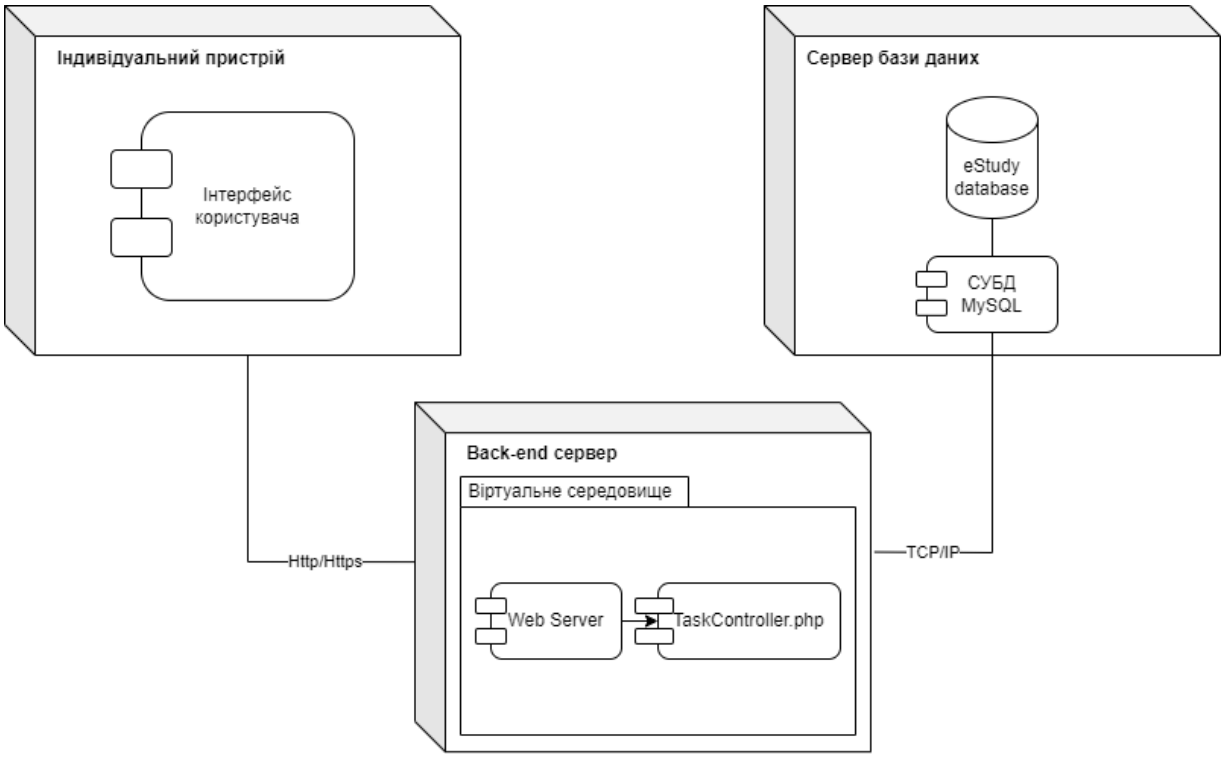

Рис. 4.1 Діаграма розгортання

4.2 Тестування системи

Після запуску системи відкривається головна сторінка програмного забезпечення (Рисунок 4.2–4.3). На головній сторінці відображається перелік дисциплін, до яких користувач має доступ відповідно до своєї ролі в системі. Крім того, у нижній частині сторінки розміщено календар, у якому відображаються завдання із зазначеними термінами виконання. Користувач може переглянути дати дедлайнів та своєчасно ознайомитися з роботами, які необхідно виконати до відповідного дня.

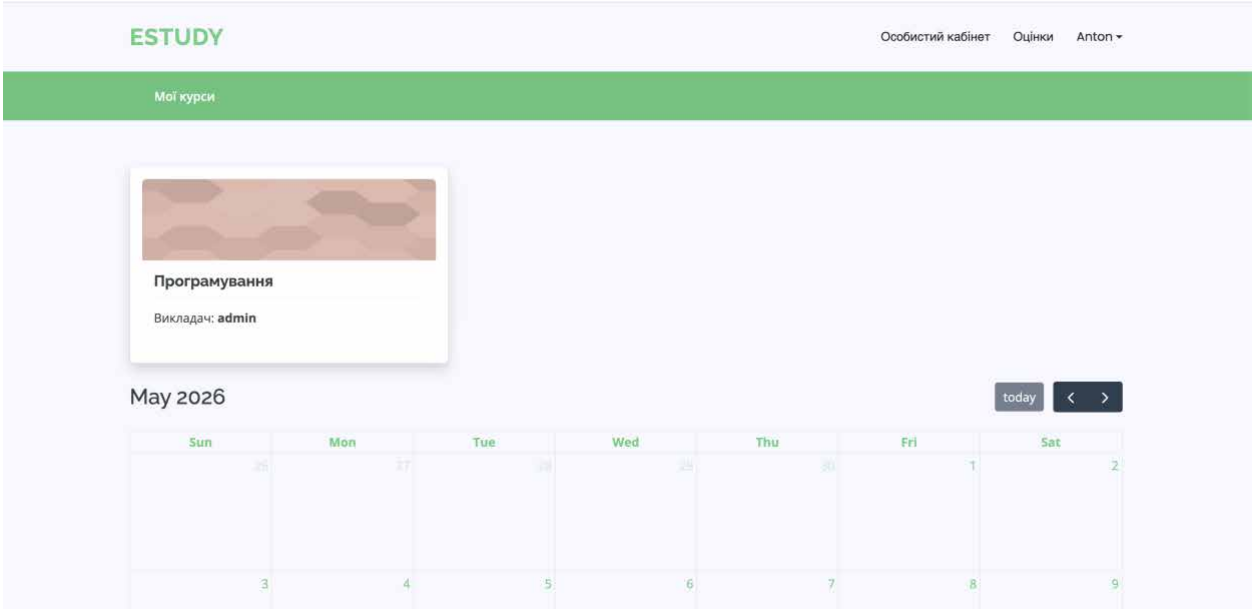

Рисунок 4.2 Головна сторінка

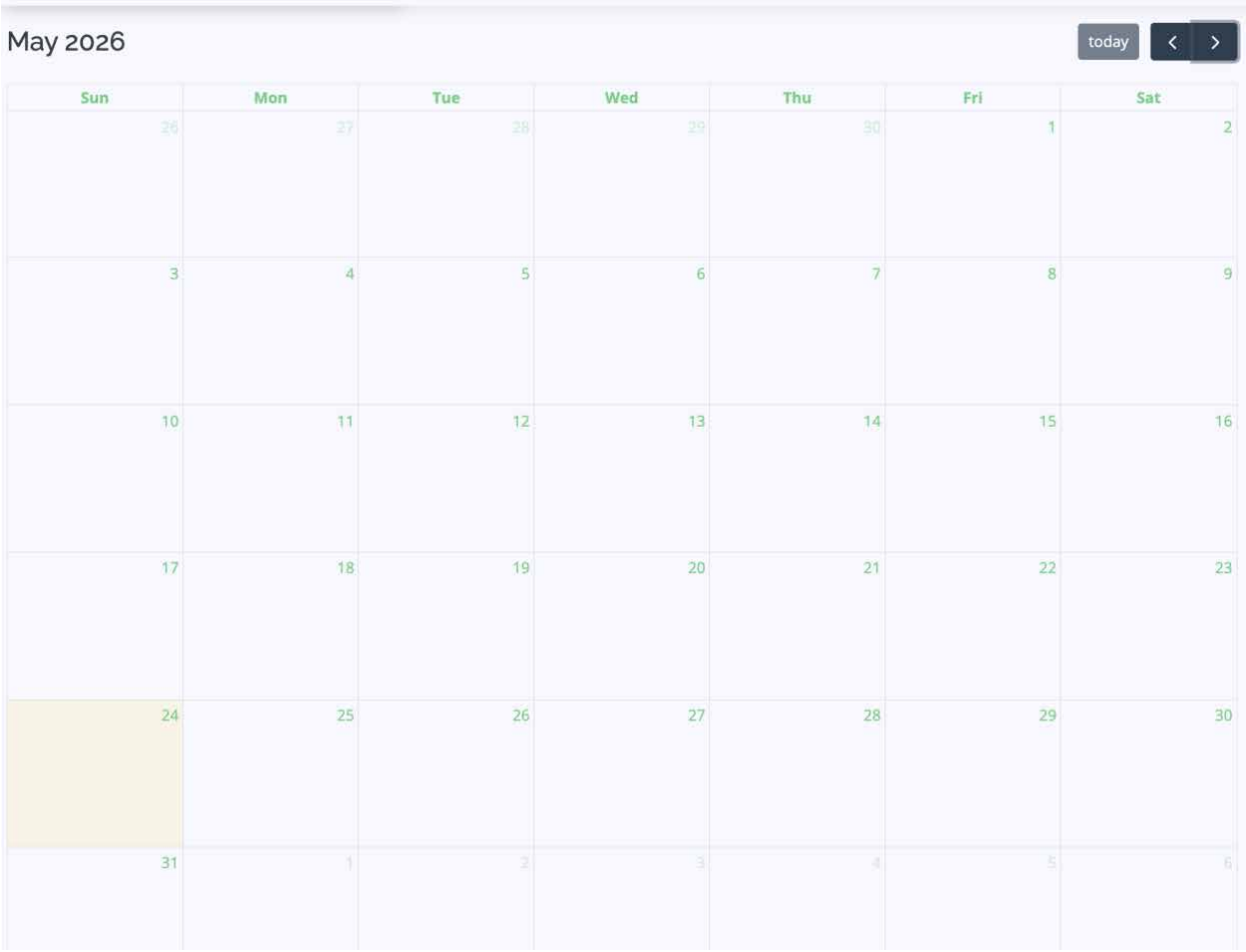

Рисунок 4.3 Календар

На Рисунок 4.4 зображено форму авторизації в системі, яку користувачу необхідно заповнити під час першого входу до системи. У формі потрібно ввести облікові дані, після чого відбувається перевірка коректності

введеної інформації та надання доступу до функціональних можливостей програмного забезпечення відповідно до ролі користувача.

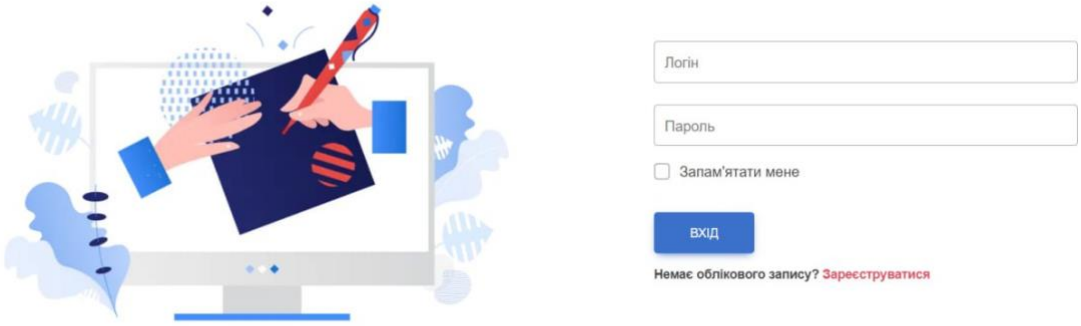

Рисунок 4.4 Авторизація

Увійшовши до системи під користувачем з правами викладача, користувач отримує можливість створювати нові дисципліни, а також у межах обраної дисципліни формувати завдання для студентів. Процес створення завдання представлено на Рисунок 4.5, де показано форму введення необхідних даних та параметрів завдання.

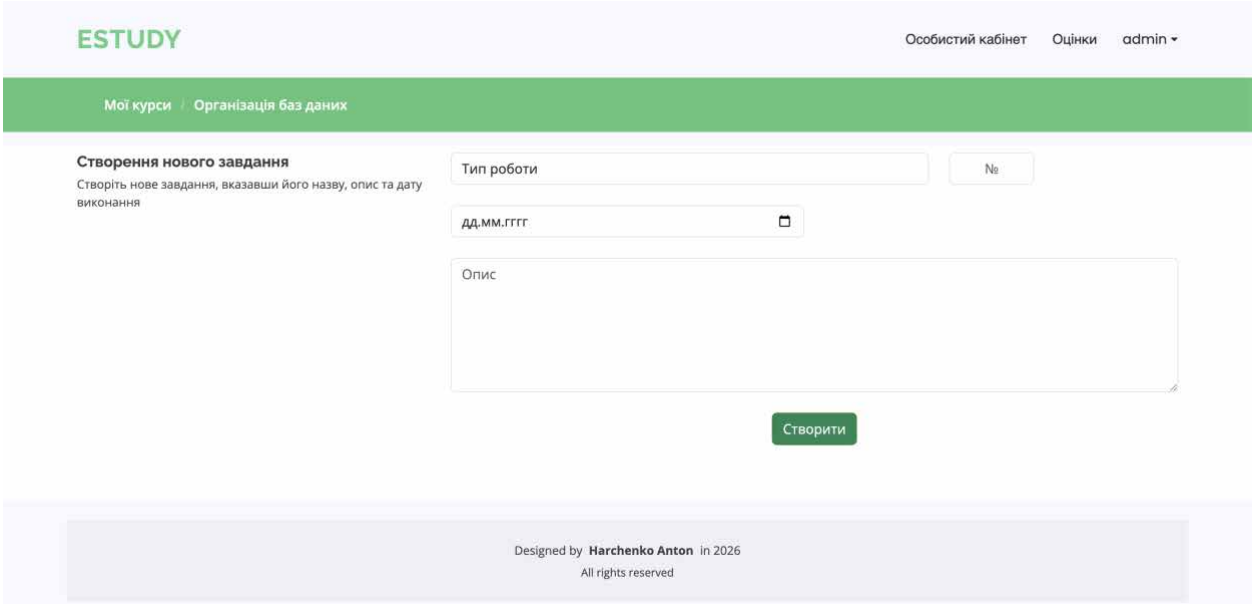

Рисунок 4.5 Форма створення завдання

Під час створення завдання необхідно обрати тип роботи (лабораторна робота, практична робота, самостійна робота, модульний тест або іспит),

вказати назву, додати опис та встановити кінцеву дату здачі. Після збереження завдання воно відображається у відповідній дисципліні.

Перейшовши до дисципліни, користувач бачить її опис, а також перелік доданих робіт із зазначеними термінами виконання. Сторінку дисципліни показано на рисунку 4.6.

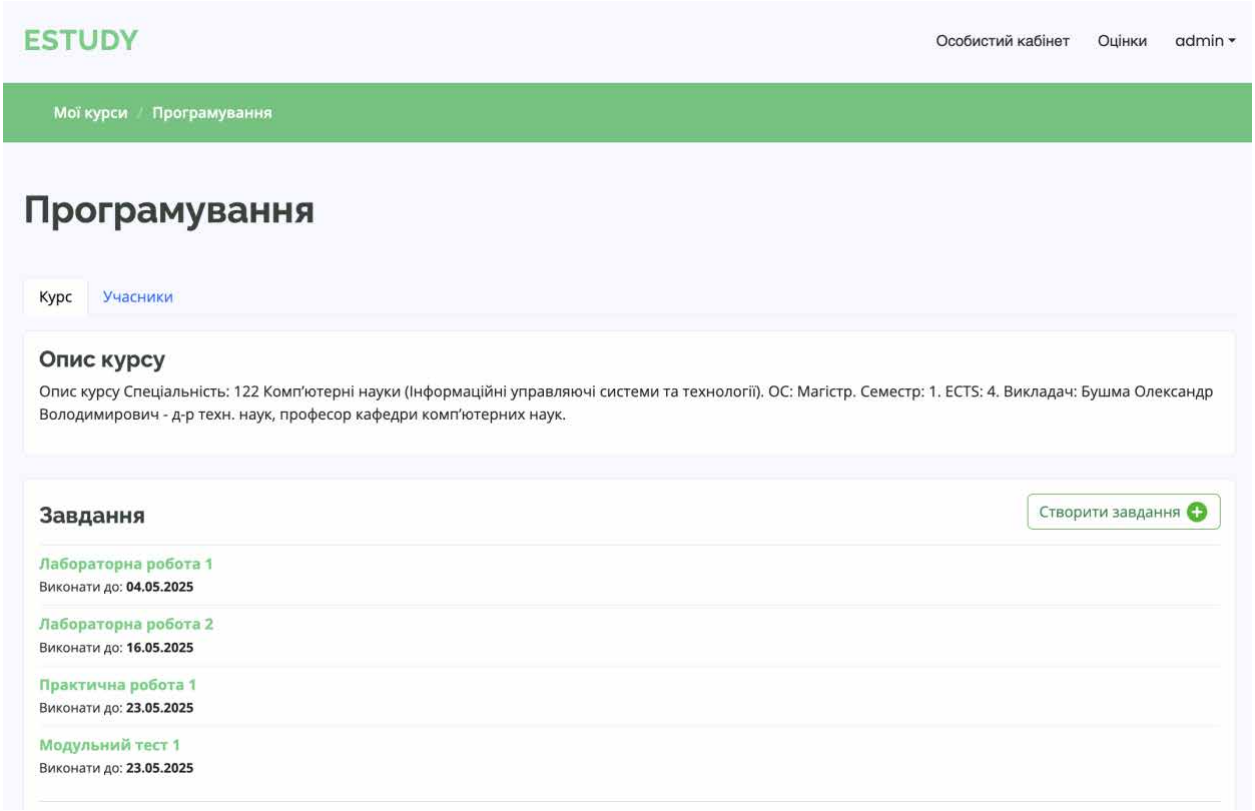

Рисунок 4.6 Сторінка дисципліни

Обравши потрібне завдання, користувач переходить на сторінку цього завдання (Рисунок 4.7). На сторінці відображається опис завдання, встановлений термін здачі, поточний статус виконання роботи, а також доступна кнопка для завантаження файлу з виконаним завданням.

Рисунок 4.7 Сторінка завдання

Після завантаження файлу та відправлення роботи на перевірку статус роботи змінюється (Рисунок 4.8). Після цього у викладача дана робота з’являється у списку робіт, що очікують перевірки (Рисунок 4.9).

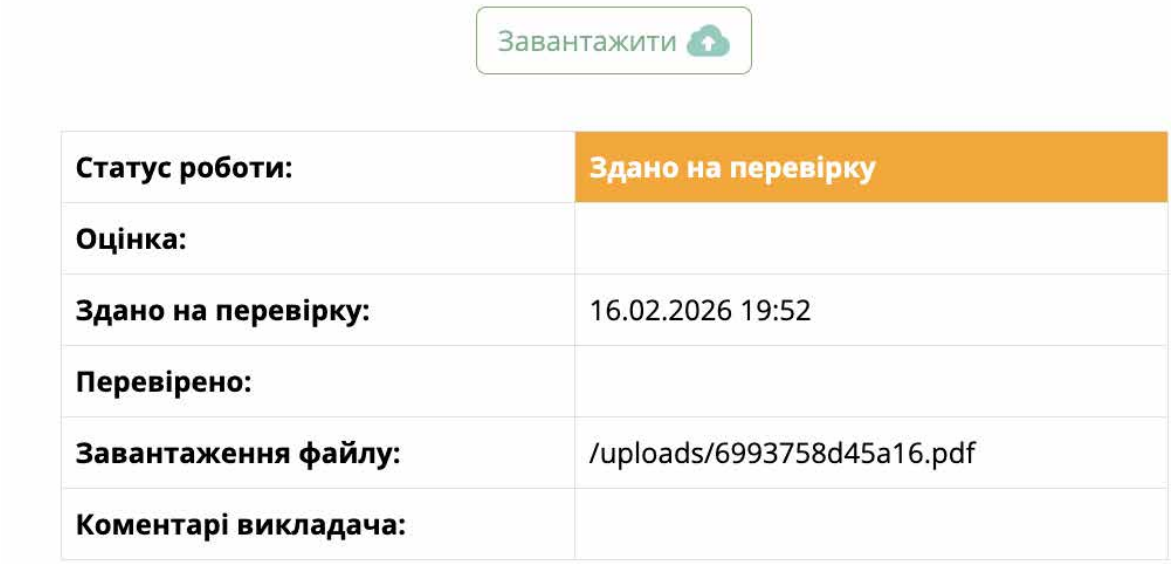

Рисунок 4.8 Зміна статусу роботи

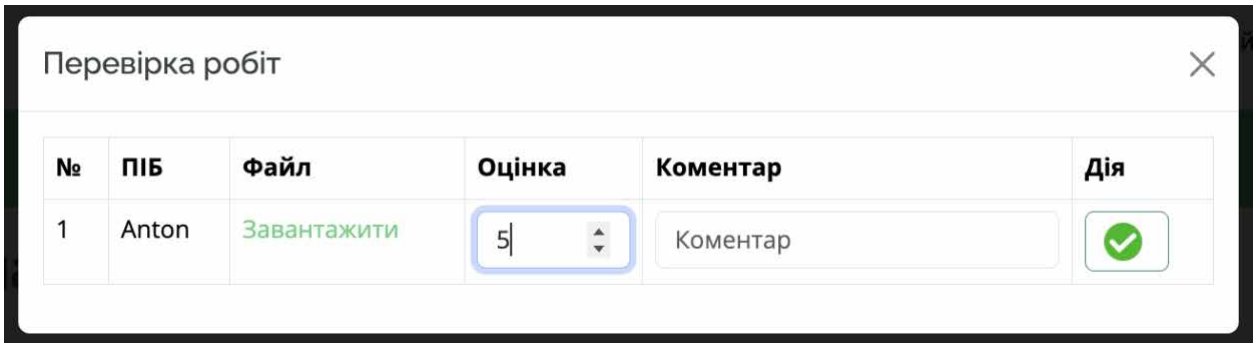

Рисунок 4.9 Перевірка робіт у викладача

Далі можна перейти на сторінку з оцінками (Рисунок 4.10), де відображаються загальні оцінки за кожну дисципліну, до якої був доданий користувач.

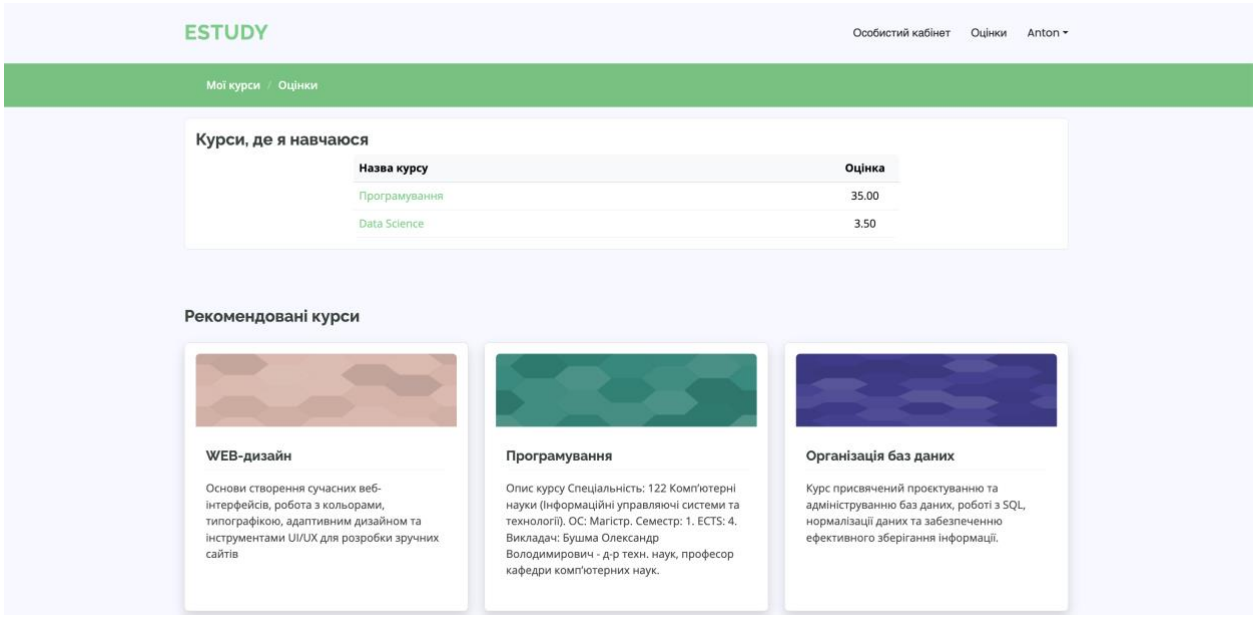

Рисунок 4.10 Сторінка “Оцінки”

Вибравши певну дисципліну, користувач переходить на сторінку, де відображаються оцінки за всі роботи цієї дисципліни. На цій сторінці показано оцінку за кожне завдання, коментар викладача та загальний бал за курс, який підсумовується внизу (Рисунок 4.11). Також тут реалізовано показ рекомендацій інших навчальних матеріалів на основі аналізу результатів студента. Йому пропонуються 3 інші курси.

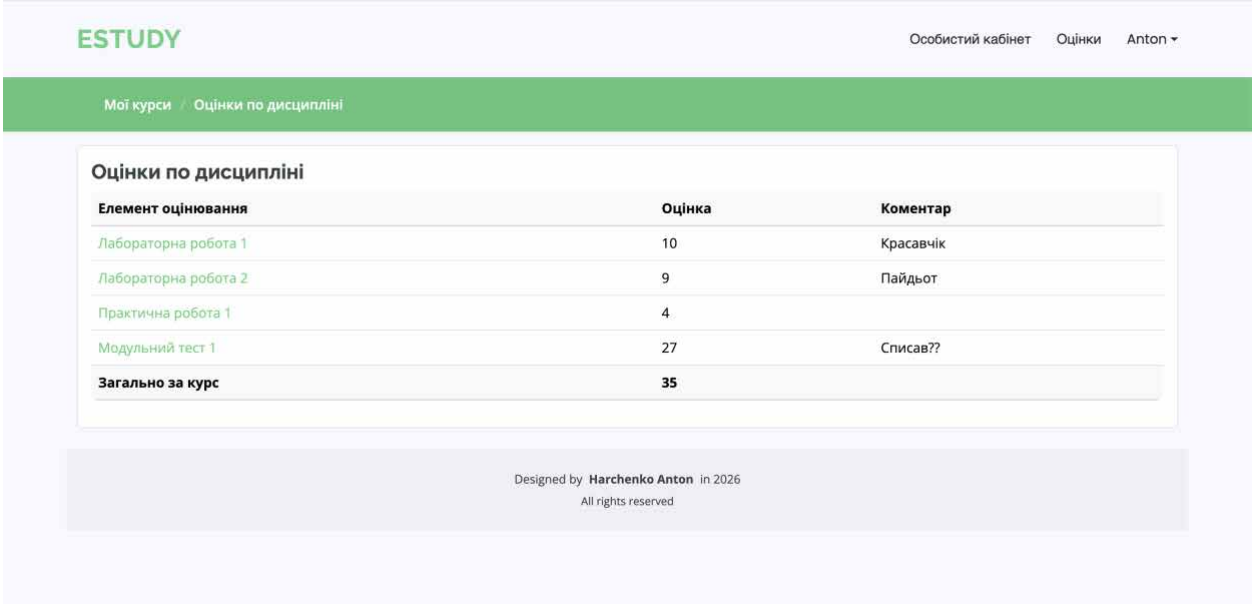

Рисунок 4.11 Оцінки по обраній дисципліни

4.3 Склад інсталяційного пакету

Інсталяційний пакет розробленої веб-системи рекомендацій навчальних матеріалів на основі аналізу прогресу студента містить повний набір програмних компонентів, конфігураційних файлів та службових ресурсів, необхідних для коректного розгортання, налаштування та подальшої експлуатації системи в локальному або серверному середовищі. Загальний обсяг інсталяційного пакету залежить від конфігурації середовища та використання додаткових бібліотек, однак базова версія системи є оптимізованою для веб-розгортання та не потребує надмірних ресурсів.

Програмна система реалізована з використанням мови програмування PHP, фреймворку Symfony та системи управління базами даних MySQL. Для забезпечення стабільного та уніфікованого середовища виконання використовується технологія контейнеризації Docker, що дозволяє автоматизувати процес розгортання та мінімізувати залежність від конкретної операційної системи.

Вихідний код програмної системи

Вихідний код є основною частиною інсталяційного пакету та містить реалізацію всієї бізнес-логіки системи. Він організований відповідно до архітектури Symfony та включає:

- контролери, що відповідають за обробку HTTP-запитів;
- сервіси бізнес-логіки, включаючи модуль обробки навчальних процесів;
- сутності ORM для взаємодії з базою даних;
- модулі авторизації та автентифікації користувачів;
- модуль оцінювання та обробки результатів навчання;
- модуль формування рекомендацій навчальних матеріалів;
- конфігураційні файли маршрутизації та налаштувань системи.

Загальна структура проєкту містить десятки програмних файлів, які забезпечують повноцінне функціонування вебзастосунку та реалізацію всіх передбачених функціональних можливостей.

Docker-конфігурація

Для автоматизації процесу розгортання системи використовуються конфігураційні файли Docker, які забезпечують створення ізольованого середовища виконання. До їх складу входять:

- Dockerfile для налаштування серверного середовища;
- docker-compose.yml для оркестрації контейнерів.

Docker-конфігурація забезпечує запуск таких компонентів системи:

- вебсервер (Nginx або Apache);
- PHP-FPM для виконання серверної логіки;
- MySQL для зберігання даних;
- додаткові сервіси кешування (за потреби).

Використання контейнеризації дозволяє забезпечити однакову роботу системи на різних середовищах та спрощує процес її розгортання.

Міграції бази даних

У складі інсталяційного пакету містяться міграції бази даних, які забезпечують автоматичне створення структури таблиць та зв’язків між ними. До них належать:

- таблиці користувачів системи;
- таблиці навчальних дисциплін;
- таблиці завдань;
- таблиці навчальних матеріалів;
- таблиці поданих робіт;
- таблиці оцінок;
- таблиці рекомендацій навчальних матеріалів.

Міграції дозволяють автоматизувати процес ініціалізації бази даних та забезпечують узгодженість структури інформаційної системи.

Конфігураційні файли середовища

Інсталяційний пакет містить файл конфігурації середовища (.env), який визначає основні параметри роботи системи:

- параметри підключення до бази даних MySQL;
- налаштування середовища виконання (development/production);
- ключі безпеки та параметри шифрування;
- конфігурацію сервера додатків.

Використання окремого конфігураційного файлу дозволяє забезпечити гнучкість налаштування системи та спрощує її перенесення між різними середовищами розгортання.

ВИСНОВКИ

У межах виконання бакалаврської кваліфікаційної роботи було розроблено програмне забезпечення системи рекомендацій навчальних матеріалів на основі аналізу прогресу студента, що поєднує функціональність навчального веб-порталу та механізми персоналізації освітнього контенту. Результатом роботи стало проєктування та часткова реалізація веб-орієнтованої інформаційної системи, яка забезпечує підтримку основних процесів дистанційного навчання, включаючи управління курсами, створення завдань, оцінювання робіт та формування індивідуальних рекомендацій для здобувачів освіти.

У процесі виконання роботи було проведено аналіз предметної області, досліджено існуючі рішення у сфері систем управління навчанням, зокрема Moodle та інші електронні освітні платформи, визначено їх основні переваги та недоліки. На основі цього було сформовано вимоги до розроблюваної системи та обґрунтовано необхідність створення механізму рекомендацій, що базується на аналізі навчального прогресу студента.

Також було здійснено проєктування програмної системи, включаючи побудову логічної моделі даних, визначення основних сутностей предметної області та їх взаємозв’язків, розробку архітектури програмного забезпечення та вибір технологічного стеку. Окрему увагу було приділено проєктуванню механізму зберігання даних та забезпеченню їх цілісності у межах реляційної бази даних MySQL.

Під час реалізації програмного забезпечення було використано сучасні технології веб-розробки, зокрема PHP, фреймворк Symfony, MySQL, Docker, а також HTML, CSS, JavaScript і Bootstrap для створення клієнтської частини системи. Це дозволило забезпечити структурованість коду, масштабованість рішення та зручність користувацького інтерфейсу.

Окремим важливим елементом роботи стала розробка механізму рекомендацій навчальних матеріалів, який дозволяє формувати персоналізовані пропозиції для студентів на основі їхніх оцінок, активності та результатів виконання завдань. Такий підхід сприяє підвищенню ефективності навчального процесу та індивідуалізації навчання.

Таким чином, у результаті виконання роботи було досягнуто поставленої мети та виконано всі визначені завдання, а саме:

- проаналізовано предметну область та існуючі системи дистанційного навчання;
- визначено функціональні та нефункціональні вимоги до програмної системи;
- розроблено модель предметної області та логічну структуру даних;
- спроєктовано архітектуру програмного забезпечення та обрано технологічний стек;
- реалізовано базові модулі веб-застосунку та механізм рекомендацій навчальних матеріалів.

Отримані результати підтверджують можливість ефективного використання розробленої системи для організації навчального процесу та підтримки індивідуального підходу до навчання студентів за рахунок аналізу їхнього прогресу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Moodle. URL: https://moodle.org (дата звернення: 16.03.2026).
2. eLearn. URL: https://elearn.nubip.edu.ua (дата звернення: 16.03.2026).
3. Багаторівнева архітектура. URL: https://simpleone.uaglossary/mnogourovnevaya-arhitektura (дата звернення: 16.03.2026).
4. Типи архітектури програмного забезпечення. URL: https://medium.com/nuances-of-programming/4-типи-архітектури-программного-забезпечення-917133174724 (дата звернення: 16.03.2026).
5. What is Unified Modeling Language (UML)? URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/ (дата звернення: 16.03.2026).
6. ПРОЄКТУВАННЯ ER-ДІАГРАМИ. URL: https://nationalteam.worldskills.ua/skills/proektirovanie-er-diagrammy/ (дата звернення: 16.03.2026).
7. Діаграма варіантів використання (UseCase diagram). URL: https://flexberry.github.io/ua/fd_use-case-diagram.html (дата звернення: 16.03.2026).
8. ПРОЄКТУВАННЯ USE CASE ДІАГРАМИ. ВИЗНАЧЕННЯ ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ СИСТЕМИ. URL: https://nationalteam.worldskills.ua/skills/proektirovanie-use-case-diagrammy-opredelenie-funktsionalnykh-vozmozhnostey-sistemy/ (дата звернення: 16.03.2026).
9. What is Sequence Diagram? URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/ (дата звернення: 16.03.2026).

10. UML - Activity Diagrams – Tutorialspoint. URL: https://www.tutorialspoint.com/uml/uml_activity_diagram.htm (дата звернення: 16.03.2026).
11. Побудова діаграми класів. URL: https://flexberry.github.io/ua/gpg_class-diagram.html (дата звернення: 16.03.2026).
12. UML-діаграми класів. URL: https://prog-cpp.ua/uml-classes/ (дата звернення: 16.03.2026).
13. Entity Relationship Diagram - Data Modeling. URL: https://www.visualparadigm.com/VPGallery/datamodeling/EntityRelationshipDiagram.html (дата звернення: 16.03.2026).
14. UML 2 Tutorial - Package Diagram - Sparx Systems. URL: https://sparxsystems.com/resources/tutorials/uml2/package-diagram.html (дата звернення: 16.03.2026).
15. Документація Bootstrap (офіційний сайт) . URL: http://www.getbootstrap.com/documentation (дата звернення: 16.03.2026).
16. W3Schools. URL: http://www.w3schools.com (дата звернення: 16.03.2026).
17. Object Diagram in UML: Definition & Examples. URL: https://study.com/academy/lesson/object-diagram-in-uml-definition-examples.html (дата звернення: 16.03.2026).
18. Logical Data Model. Dataedo. URL: https://dataedo.com/kb/data-glossary/logical-data-model (дата звернення: 16.03.2026).
19. Logical Data Model. Techopedia. URL: https://www.techopedia.com/definition/23969/logical-data-model (дата звернення: 16.03.2026).
20. Layered architecture pattern / Microsoft Docs. URL: https://docs.microsoft.com/en-us/azure/architecture/patterns/layered (дата звернення: 16.03.2026).

21. What is Component Diagram? URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/ (дата звернення: 16.03.2026).
22. Deployment Diagram in UML: Definition, Examples & Components. URL: https://study.com/academy/lesson/deployment-diagram-in-uml-definition-examples-components.html (дата звернення: 16.03.2026).
23. What is a data flow diagram? URL: https://www.lucidchart.com/pages/data-flow-diagram (дата звернення: 16.03.2026).
24. Сміт Дж. А. Впровадження систем управління навчанням у вищій освіті: практичний приклад. Міжнародний журнал освітніх технологій. 2020. Т. 15, № 3. С. 32–50.
25. Браун Л. К., Грін Т. Д. Роль технологій у сучасній освіті: тенденції та інновації. Журнал розвитку та обміну освітніми технологіями. 2018. Т. 11, № 1. С. 15–29.
26. Гарсія Р. Створення ефективного середовища онлайн-навчання: стратегії успіху. Journal of Online Learning and Teaching. 2019. Т. 15, № 4. С. 12–22.
27. Міллер С., Тернер П. Електронне навчання та його вплив на залучення здобувачів вищої освіти. Journal of Educational Research and Practice. 2021. Т. 8, № 2. С. 67–79.
28. Томпсон Х. Майбутнє освітніх технологій: тенденції, на які варто звернути увагу. Журнал освітніх технологій. 2020. Т. 20, № 1. С. 24–30.
29. Вілсон Р. Оцінка ефективності інформаційних систем здобувачів вищої освіти. Журнал інформаційних технологій в освіті. 2019. Т. 17, № 3. С. 40–55.
30. Андерсон Т. Орієнтований на користувача дизайн у розробці освітнього програмного забезпечення. Міжнародний журнал взаємодії людини та комп’ютера. 2021. Т. 37, № 5. С. 550–564.

ДОДАТОК А

Фрагменти програмного коду. Функція створення завдання

```php
class TaskController extends AbstractController
{
    #[Route('/course/{id}/task/create', name: 'task_create')]
    public function createTask(
        int $id,
        Request $request,
        EntityManagerInterface $entityManager,
        CourseRepository $courseRepository,
        CategoryRepository $categoryRepository
    ) {
        $course = $courseRepository->find($id);
        $categories = $categoryRepository->findAll();

        if ($request->isMethod('POST')) {
            $title = $request->request->get('title');
            $description = $request->request->get('description');
            $categoryId = $request->request->get('category_id');
            $dueDateString = $request->request->get('due_date');

            if (!$course) {
                throw $this->createNotFoundException('Курс не знайдено');
            }

            $dueDate = new \DateTime($dueDateString);

            $task = new Task();
            $task->setCourse($course);
            $task->setTitle($title);
            $task->setDescription($description);
            $task->setCategory($categoryRepository->find($categoryId));
            $task->setDueDate($dueDate);

            $entityManager->persist($task);
            $entityManager->flush();

            return $this->redirectToRoute('course_show', ['id' => $id]);
        }

        return $this->render('task/create.html.twig', [
            'course' => $course,
            'categories' => $categories
        ]);
    }

    #[Route('/task/{id}', name: 'task_show')]
```

```
    public function showTask(int $id, EntityManagerInterface $entityManager,
Security $security): Response
    {
        $task = $entityManager->getRepository(Task::class)->find($id);

        if (!$task) {
            throw $this->createNotFoundException('Завдання не знайдено');
        }

        $user = $security->getUser();
        if (!$user) {
            throw $this->createAccessDeniedException('Ви не авторизовані');
        }

        $submission = $entityManager->getRepository(Submission::class)-
>findOneBy(['task' => $task, 'student' => $user]);
        $grade = $submission ? $entityManager->getRepository(Grade::class)-
>findOneBy(['submission' => $submission]) : null;

        $submissionsToCheck = $entityManager->getRepository(Submission::class)-
>findBy(['task' => $task, 'status' => 1]);

        return $this->render('task/show.html.twig', [
            'task' => $task,
            'submission' => $submission,
            'grade' => $grade,
            'course' => $task->getCourse(),
            'submissionsToCheck' => $submissionsToCheck,
        ]);
    }
}
```

ДОДАТОК Б

Фрагменти програмного коду. Функціонал оцінювання завдання

```
#[Route('/submission/{id}/grade', name: 'submission_grade', methods: ['POST'])]
    public function gradeSubmission(
        int $id,
        Request $request,
        EntityManagerInterface $entityManager
    ): JsonResponse {
        $submission = $entityManager->getRepository(Submission::class)-
>find($id);

        if (!$submission) {
            return new JsonResponse(['success' => false, 'message' => 'Робота не
знайдена'], 404);
        }

        $data = json_decode($request->getContent(), true);
        $score = $data['score'] ?? null;
        $comment = $data['comment'] ?? null;

        if ($score === null || $score < 0 || $score > 100) {
            return new JsonResponse(['success' => false, 'message' => 'Некоректна
оцінка'], 400);
        }

        $grade = new Grade();
        $grade->setSubmission($submission);
        $grade->setComment($comment);
        $grade->setScore($score);
        $grade->setRatedDate(new \DateTime());

        $submission->setStatus(2);

        $entityManager->persist($grade);
        $entityManager->flush();

        return new JsonResponse(['success' => true]);
```

ДОДАТОК В

Фрагменти програмного коду. Розрахунок оцінок та рекомендацій

```
class GradeController extends AbstractController
{
    #[Route('/grades', name: 'grades')]
    public function index(EntityManagerInterface $entityManager, Security
$security, GradeService $gradeService): Response
    {
        $user = $security->getUser();
        if (!$user) {
            return $this->redirectToRoute('login');
        }

        $courseStudents = $entityManager->getRepository(CourseStudents::class)-
>findBy(['student' => $user]);
        $submissionRepository = $entityManager->getRepository(Submission::class);

        $grades = [];

        foreach ($courseStudents as $courseStudent) {
            $course = $courseStudent->getCourse();
            $submissions = $submissionRepository-
>findSubmissionsByStudentAndCourse($user, $course);

            $finalGrade = $gradeService->calculateFinalGrade($submissions);

            $grades[] = [
                'course' => $course,
                'finalGrade' => $finalGrade,
            ];
        }

        return $this->render('grades/show.html.twig', [
            'grades' => $grades,
        ]);
    }

    #[Route('/grades/course/{courseId}', name: 'grades_course')]
    public function courseGrades(EntityManagerInterface $entityManager, Security
$security, GradeService $gradeService, int $courseId): Response
    {
        $user = $security->getUser();
        if (!$user) {
            return $this->redirectToRoute('login');
        }

        $submissionRepository = $entityManager->getRepository(Submission::class);
```

```
        $submissions = $submissionRepository-
>findSubmissionsByStudentAndCourse($user, $courseId);

        $grades = [];

        foreach ($submissions as $submission) {
            $grades[] = [
                'task' => $submission->getTask(),
                'grade' => $submission->getGrade(),
            ];
        }

        $finalGrade = $gradeService->calculateFinalGrade($submissions);

        return $this->render('grades/course.html.twig', [
            'grades' => $grades,
            'finalGrade' => $finalGrade,
        ]);
    }

class GradeService
{
    /**
     * Розрахунок загального балу студента за дисципліну
     *
     * @param Submission[] $submissions
     * @return float
     */
    public function calculateFinalGrade(array $submissions): float
    {
        $totalScore = 0;

        foreach ($submissions as $submission) {
            $grade = $submission->getGrade();

            if ($grade) {
                $category = $submission->getTask()->getCategory();
                $score = $grade->getScore();

                if (in_array($category->getId(), [1, 2, 3, 4])) {
                    $totalScore += $score * 0.7;
                } elseif ($category->getId() === 5) {
                    $totalScore += $score;
                }
            }
        }

        return $totalScore;
    }
}
```

ДОДАТОК Г

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій Декларація про академічну доброчесність та використання ШІ

Я, Харченко Антон Володимирович, здобувач НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота на тему «Програмне забезпечення системи рекомендацій навчальних матеріалів на основі аналізу прогресу студента.

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

3. Щодо використання інструментів ШІ зазначаю, що:

інструменти ШІ Gemini були використані для:

- + перекладу іншомовних джерел;
- + стилістичного редагування та перевірки граматики;
- + допомоги у написанні/відлагодженні програмного коду;
- + інше: пошуку додаткових інструментів у розробці програмного продукту.

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

«22» Травня 2026 р. _____ **Антон ХАРЧЕНКО**