НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри
_____комп'ютерних наук_____
(назва кафедри)

_____ /_____Голуб Б.Л._____/
(підпис) (ПІБ)

"___" _____ 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему

«Інформаційна система управління корпоративним навчанням»

Спеціальність 122 – «Комп'ютерні науки»

Гарант освітньої програми
_____д.е.н., професор_____ _____ _____Руденський Р.А.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи
_____ _____ _____Саяпін С.П._____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ _____Яковенко Олександр Сергійович_____
(підпис) (ПІБ студента)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____комп'ютерних наук_____
(назва кафедри)

___к.т.н., доцент_____Голуб Б.Л.
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"___" _____ 2025 р.

З А В Д А Н Н Я
на виконання бакалаврської кваліфікаційної роботи студенту
Яковенко Олександр Сергійович

Спеціальність 122 – «Комп'ютерні науки»

1. Тема бакалаврської кваліфікаційної роботи «Інформаційна система управління корпоративним навчанням» затверджена наказом ректора НУБіП України від 29.04.2025 №711с
2. Термін подання завершеної роботи на кафедру_____
(рік, місяць, число)
3. Вихідні дані до роботи: надання інформації про успішність проходження перевірки знань працівників/учнів у темах, що потребують закріплення знань, у вигляді балів.
4. Перелік питань, що розглядаються:
   - Аналіз проблемної області
   - Моделювання предметної області
   - Проєктування програмної системи
   - Впровадження та експлуатація системи

Дата видачі завдання "___"_____ 2025 р.

Керівник бакалаврської кваліфікаційної роботи _____/ Саяпін С.П. /
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання:_____/ Яковенко О.С. /
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

У цій бакалаврській роботі представлено розробку інформаційної системи для управління корпоративними навчальними процесами, спрямованої на підвищення ефективності оцінювання знань та відстеження навчання в організаціях.

Запропонований веб-додаток розроблено з використанням PHP (Symfony framework) та MySQL як системи управління базами даних. Система призначена для підтримки двох ролей користувачів: адміністраторів та студентів. Адміністратори відповідають за створення навчальних тестів, керування тестовими питаннями та призначення студентів до певних тестів. Студенти, у свою чергу, можуть входити в систему, проходити призначені тести та отримувати автоматичні оцінки на основі своїх відповідей.

Система забезпечує такі важливі функції, як безпечна автентифікація, створення та керування тестами, оцінювання в режимі реального часу та відстеження прогресу окремих студентів. Завдяки впровадженню цієї системи організації можуть оптимізувати свої внутрішні навчальні процеси, зменшити адміністративне навантаження та покращити оцінку знань співробітників.

ABSTRACT

This bachelor's thesis presents the development of an information system for managing corporate training processes, aimed at improving the efficiency of knowledge assessment and training tracking within organizations.

The proposed web-based application is developed using PHP (Symfony framework) and MySQL as the database management system. The system is designed to support two user roles: administrators and students. Administrators are responsible for creating training tests, managing test questions, and assigning students to specific tests. Students, in turn, are able to log in to the system, complete assigned tests, and receive automatic grading based on their responses.

The system provides essential functionalities such as secure authentication, test creation and management, real-time grading, and tracking of individual student progress. Through the implementation of this system, organizations can streamline their internal training processes, reduce administrative workload, and improve employee knowledge evaluation.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання бакалаврської роботи	Строк виконання етапів бакалаврської роботи	Примітка
1	Отримання завдання	21 лютого 2025	
2	Аналіз предметної області	березень 2025	
3	Моделювання предметної області	березень 2025	
4	Проєктування програмної системи	квітень 2025	
5	Розгортання та експлуатація програмного забезпечення	квітень- травень 2025	
6	Економічне дослідження розробки та експлуатації програмної системи	квітень- травень 2025	
7	Оформлення записки	травень 2025	
8	Перевірка на плагіат	2 червня 2025	
9	Проходження нормо контроля	29 травня – 4 червня 2025	
10	Проходження передзахисту	5-7 червня 2025	
11	Захист роботи	14-16 червня 2025	

Студент _____ Яковенко О.С.

(підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи

доцент к.т.н. _____ Саяпін С.П.

(науковий ступінь та вчене звання) (підпис) (ПІБ)

ЗМІСТ

ВСТУП

В епоху цифрових технологій успіх організації дедалі більше залежить від здатності її співробітників швидко адаптуватися до нових технологій, інструментів та процесів. Корпоративне навчання стало життєво важливим компонентом розвитку людських ресурсів, що дозволяє компаніям покращувати навички співробітників, підвищувати продуктивність та залишатися конкурентоспроможними на динамічних ринках. Однак традиційні форми навчання часто передбачають значні витрати часу та ресурсів, а також не мають ефективних інструментів для оцінки та відстеження прогресу.

Для вирішення цих проблем багато компаній звертаються до цифрових рішень, які пропонують гнучкість, масштабованість та ефективність в управлінні навчальними програмами. Системи управління навчанням (LMS) стали ключовим компонентом цієї трансформації, дозволяючи організаціям створювати, розповсюджувати та контролювати навчальні матеріали в електронному вигляді. Незважаючи на наявність різних комерційних платформ, багатьом підприємствам потрібні індивідуальні рішення, які відповідають їхнім конкретним робочим процесам, стандартам безпеки даних та потребам інтеграції.

У цій бакалаврській роботі представлено розробку індивідуальної інформаційної системи для управління корпоративним навчанням. Система реалізована як веб-додаток з використанням PHP (Symfony framework) та MySQL як бази даних.

Основною **метою** системи є сприяння створенню, розповсюдженню та оцінці навчальних тестів у компаніях задля покращення рівня знань на потрібні у внутрішніх колах теми. Платформа підтримує дві ролі користувачів: адміністратори, які можуть створювати та керувати тестами, а також

відстежувати успішність студентів; та студенти, які можуть складати тести та отримувати автоматичний зворотний зв'язок та оцінювання.

Актуальність цієї роботи полягає у зростаючому попиті на доступні та економічно ефективні навчальні рішення, які підтримують розвиток співробітників, одночасно зменшуючи ручні витрати. Запропонована система спрямована на оптимізацію процесів корпоративного навчання, забезпечення послідовної оцінки знань та покращення результатів навчання в організації.

Робота охоплює такі аспекти:

- аналіз існуючих рішень для корпоративного навчання та їх обмежень;
- проєктування функціональної та технічної архітектури системи;
- реалізація основних функцій, таких як управління користувачами, створення тестів та автоматичне оцінювання;
- тестування та оцінка системи з точки зору зручності використання, функціональності та масштабованості.

Ця робота робить внесок у сферу розробки програмного забезпечення для управління персоналом та освіти, пропонуючи практичне та адаптивне рішення для управління корпоративним навчанням у сучасному підприємстві.

1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Опис предметної області

У сучасному швидкозмінному бізнес-середовищі компанії постійно перебувають під тиском, щоб впроваджувати інновації, залишатися конкурентоспроможними та адаптуватися до нових технологій. Одним із критичних факторів, що забезпечують таку адаптивність, є здатність ефективно навчати співробітників. Корпоративне навчання відіграє життєво важливу роль у розвитку компетенцій персоналу, підтримці професійного зростання та забезпеченні ефективності та актуальності бізнес-процесів. Однак традиційні методи проведення навчання, такі як друковані матеріали, очні семінари та ручне оцінювання, більше не є достатніми. Вони часто виявляються трудомісткими, негнучкими та важкими для масштабування, особливо для компаній з географічно розподіленою робочою силою.

Зі зростанням популярності віддаленої роботи та цифрової трансформації зростає потреба в доступних, централізованих та автоматизованих системах, які можуть керувати повним циклом навчання — від призначення навчальних матеріалів до відстеження індивідуальної ефективності. На жаль, багато організацій досі покладаються на фрагментовані інструменти або універсальні рішення, які не повністю відповідають їхнім внутрішнім процесам. Ця розбіжність призводить до неефективності, невідповідних стандартів оцінювання та обмеженої видимості прогресу співробітників. У таких умовах адміністратори стикаються зі значними труднощами під час організації навчальних сесій, управління результатами тестів та визначення областей, які потребують покращення.

Щоб подолати ці труднощі, багато компаній впровадили системи управління навчанням (LMS). Ці платформи пропонують такі функції, як створення тестів, автоматичне оцінювання та аналітика ефективності.

Прикладами є Moodle, TalentLMS та SAP Litmos. Хоча ці системи є потужними, вони часто мають обмеження. Висока вартість підписки, обмежені можливості налаштування та складна інтеграція з внутрішньою інфраструктурою можуть бути значними перешкодами, особливо для малого або середнього бізнесу [1].

Визнаючи цю прогалину, стає очевидним, що індивідуальне рішення може запропонувати більшу гнучкість та точність. Система, спеціально розроблена для навчального процесу компанії, може гарантувати, що як адміністратори, так і учні отримають користь від зручного, структурованого та орієнтованого на результат середовища. Така система дозволить автоматизувати повторювані завдання, зменшити людські помилки під час оцінювання, надасть змістовне розуміння ефективності учнів та адаптуватиметься в міру зростання компанії.

Розробка спеціалізованої веб-системи корпоративного навчання може таким чином усунути недоліки існуючих інструментів. Зосереджуючись на простоті, автоматизації та зрозумілості, запропонований додаток має на меті покращити те, як організації забезпечують внутрішню освіту, оцінюють знання та керують результатами навчання в командах. Завдяки чіткому розподілу ролей користувачів — адміністратори керують контентом, а учні складають тести — система підтримує ефективну комунікацію, відстеження ефективності та довгостроковий розвиток навичок.

Корпоративне навчання перетворилося на стратегічну необхідність для організацій, які прагнуть залишатися конкурентоспроможними в динамічному бізнес-середовищі. Оскільки компанії все більше інвестують у розвиток співробітників, процес організації та управління навчальною діяльністю став складнішим та вимогливішим. Предметна галузь управління корпоративним навчанням охоплює широкий спектр функцій — від розробки освітнього контенту та планування навчальних сесій до моніторингу участі та оцінки результатів навчання.

Сучасне корпоративне навчання не обмежується традиційними очними семінарами. Зі зростанням цифровізації бізнес переходить до більш гнучких, масштабованих та економічно ефективних рішень. Онлайн-платформи навчання та цифрові оцінювання стали стандартними інструментами для передачі знань між відділами та географічними місцями. У цьому контексті потреба в централізованій системі, яка може керувати всім життєвим циклом навчання, є більш нагальною, ніж будь-коли [2].

Основна ідея системи управління корпоративним навчанням полягає в тому, щоб забезпечити структуроване середовище, де навчальні матеріали можна ефективно створювати, розповсюджувати та оцінювати. Такі системи зазвичай обслуговують дві основні групи користувачів: адміністраторів (зазвичай це співробітники відділу кадрів, тренери або менеджери) та стажерів (співробітників, які проходять навчання). Адміністратори відповідають за створення тестів або курсів, реєстрацію користувачів, відстеження результатів та створення аналітичних звітів. З іншого боку, стажери взаємодіють із системою для виконання навчальних заходів та перегляду своєї ефективності.

Ефективна система управління навчанням забезпечує не лише безперебійну передачу знань, але й підзвітність та прозорість. Вона допомагає організаціям узгоджувати цілі навчання з бізнес-цілями, виявляти прогалини в навичках та вести облік досягнень співробітників. Крім того, цифровий характер таких систем сприяє віддаленому доступу, оновленню даних у режимі реального часу та інтеграції з іншими корпоративними інструментами, такими як бази даних HR або програмне забезпечення для управління ефективністю [3].

Ця робота зосереджена на розробці веб-інформаційної системи, яка відповідає фундаментальним вимогам корпоративного управління навчанням. Система спрямована на спрощення адміністративних процесів, підвищення залученості учнів та надання змістовного розуміння ефективності навчання, таким чином слугуючи практичним інструментом для корпоративного навчання в цифрову епоху.

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**
Тест	Назва	Назва тесту, яка описує його тему чи мету.
	Опис	Короткий опис змісту тесту або його мети.
	Дата створення	Дата, коли тест був створений адміністратором.
Питання	Текст питання	Текст питання, яке задається в тесті.
	Тип	Тип питання: відкритий чи з варіантами відповідей.
	Правильна відповідь	Вказівка на правильну відповідь для питання.
Варіанти відповіді	Текст варіанту	Текст одного з варіантів відповіді на питання.
	Позначка правильного	Позначка, яка вказує, чи є варіант правильною відповіддю.
Результат тестування	Оцінка	Результат тесту у вигляді оцінки або кількості балів.
	Час завершення	Час, коли студент завершив тестування.
	Статус	Статус тесту (наприклад, "завершено", "не завершено").

1.2 Огляд існуючих рішень

У сучасних умовах цифровізації освіти та бізнес-процесів на ринку існує чимало рішень, які дозволяють організаціям ефективно організовувати процес корпоративного навчання. Більшість із них представлені у вигляді платформ для електронного навчання або систем управління навчанням

(Learning Management Systems, LMS), які забезпечують інструменти для створення курсів, тестів, контролю результатів і звітності.

Moodle — одна з найпоширеніших систем управління навчанням (LMS) з відкритим кодом, призначена для підтримки онлайн-освіти та навчання. Спочатку розроблена для академічних середовищ, вона знайшла широке застосування в корпоративному секторі завдяки своїй гнучкості, масштабованості та економічній ефективності. Її модульна архітектура дозволяє організаціям налаштовувати платформу відповідно до своїх конкретних потреб у навчанні без ліцензійних зборів [4].

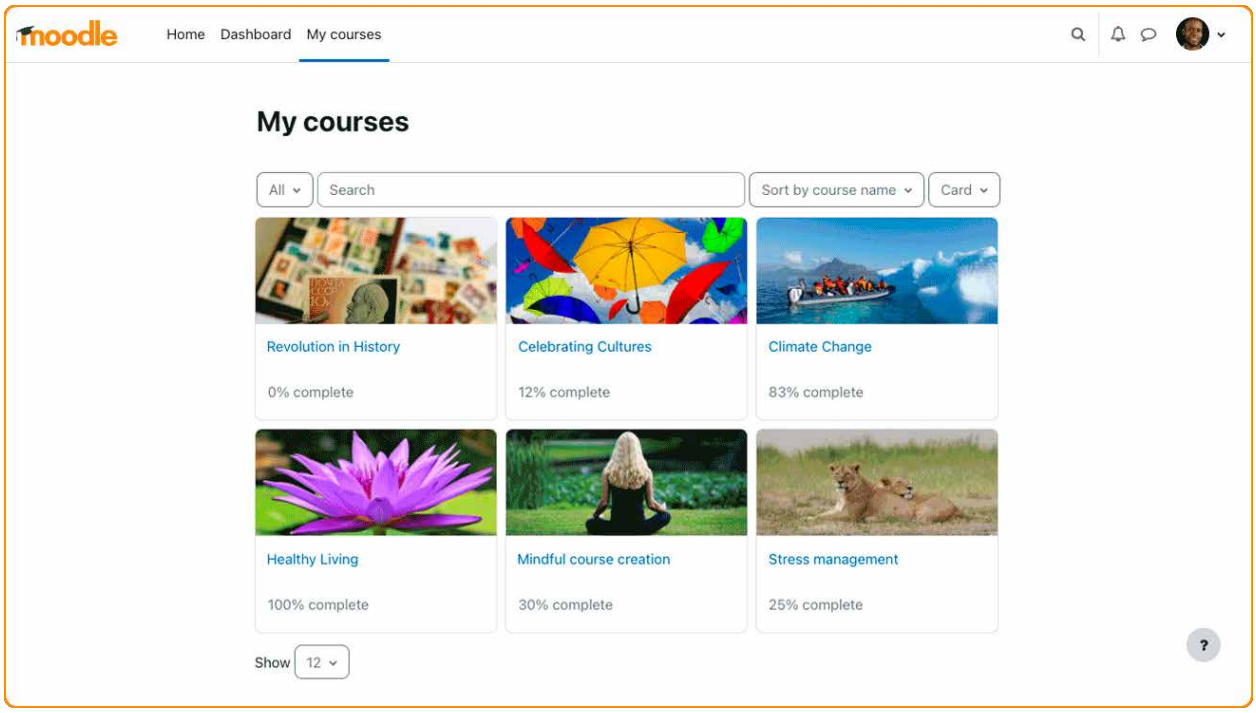

Рис. 1 Інформаційна система “Moodle”

Однією з ключових переваг Moodle є широкий спектр вбудованих функцій, які включають інструменти для створення курсів, вікторини та тести, дискусійні форуми, завантаження файлів, системи оцінювання та відстеження прогресу. Адміністратори можуть створювати структуровані навчальні шляхи, призначати ролі та контролювати залученість учнів у режимі реального часу. Студенти ж мають доступ до зручного інтерфейсу, де вони можуть проходити курси, подавати завдання та отримувати відгуки.

Незважаючи на свої потужні можливості, Moodle може бути складним для невеликих організацій або нетехнічних користувачів. Його налаштування

та персоналізація можуть вимагати спеціалізованих знань, особливо коли йдеться про хостинг, обслуговування або інтеграцію з іншими системами. Однак завдяки сильній підтримці спільноти та обширній документації, багато з цих проблем можна ефективно вирішити.

Відкритий код Moodle робить його переконливим вибором для компаній, які хочуть повного контролю над своїм навчальним середовищем, не покладаючись на дороге комерційне програмне забезпечення. Він підтримує різні плагіни та теми, що дозволяє розробникам розширювати його функціональність та адаптувати дизайн відповідно до корпоративного брендингу [4].

На завершення, Moodle виділяється як надійна та гнучка платформа для проведення та управління онлайн-навчанням. Хоча він може вимагати певних технічних інвестицій, його адаптивність та нульова вартість ліцензування роблять його привабливим варіантом для організацій з конкретними цілями навчання та розвитку.

Розглянемо інше програмне рішення на ринку.

TalentLMS — це хмарна система управління навчанням, розроблена спеціально для корпоративного навчання, яка пропонує інтуїтивно зрозумілу та просту у використанні платформу як для адміністраторів, так і для учнів. На відміну від традиційних рішень з відкритим кодом, таких як Moodle, TalentLMS надає повністю керований сервіс, що зменшує складність встановлення, налаштування та обслуговування системи. Це робить його привабливим вибором для організацій, яким бракує технічних ресурсів для управління власною інфраструктурою LMS [5].

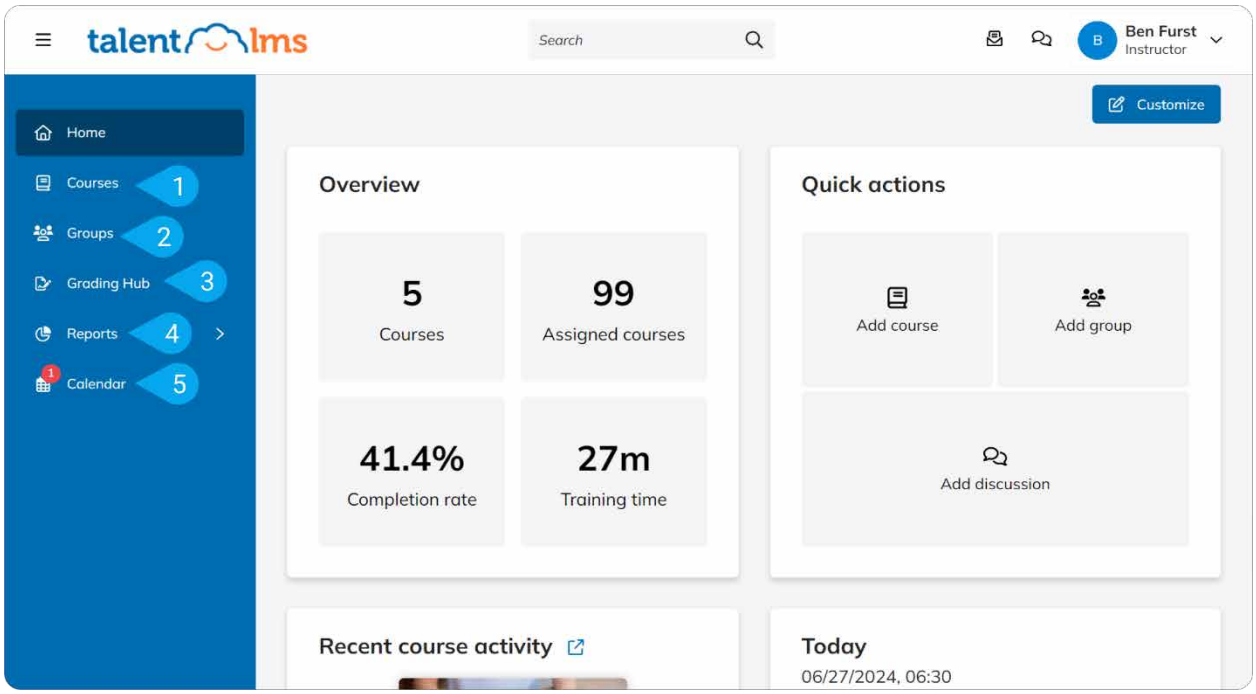

Рис. 2 Інформаційна система “TalentLMS”

Однією з видатних особливостей TalentLMS є зручний інтерфейс. Адміністратори можуть швидко створювати та проводити курси, відстежувати прогрес співробітників та генерувати детальні звіти без необхідності мати значні технічні знання. Платформа підтримує різноманітні формати контенту, включаючи текст, відео, пакети, сумісні з SCORM, та оцінювання, що дозволяє компаніям створювати різноманітний навчальний досвід.

TalentLMS також пропонує такі функції, як мобільне навчання, гейміфікація, інструменти соціального навчання та інтеграція з широким спектром сторонніх програм, включаючи HR-системи, інструменти CRM та платформи відеоконференцій. Це робить його гнучким рішенням, яке легко вписується в існуючу інфраструктуру організації. Крім того, TalentLMS підтримує кілька мов, що робить його ідеальним для глобальних підприємств з різноманітною робочою силою [5].

Однак, хоча TalentLMS простий у використанні та пропонує низку опцій налаштування, це не безкоштовне рішення. Платформа працює за підпискою, а цінові рівні залежать від кількості користувачів та необхідного рівня функціональності. Це може бути обмеженням для невеликих організацій або тих, хто має обмежений бюджет. Крім того, як власне рішення, TalentLMS

може не пропонувати такого ж рівня контролю та гнучкості, як платформи з відкритим кодом, такі як Moodle.

На завершення, TalentLMS – це потужна та ефективна платформа LMS, що підходить для організацій, які шукають просте у розгортанні хмарне рішення для управління корпоративним навчанням. Її надійні функції в поєднанні з простотою використання та масштабованістю роблять її популярним вибором для компаній, які прагнуть покращити свої програми навчання та розвитку співробітників.

1.3 Постановка завдання

Ця система відповідає за управління процесами корпоративного навчання, зосереджуючись на створенні, адмініструванні та оцінці навчальних програм. Вона дозволяє безперешкодно керувати користувачами, курсами, тестами та результатами. Платформа також надає комплексну базу даних, яка дозволяє адміністраторам ефективно відстежувати та оцінювати прогрес навчання кожного співробітника, одночасно встановлюючи та контролюючи цілі ефективності.

Основна програмна система, яка розробляється, надасть адміністраторам швидкий доступ до важливих даних, зокрема:

- ефективність навчальних програм та курсів;
- категоризація та розподіл навчальних матеріалів і тестів;
- встановлення навчальних цілей та цілей ефективності для кожного користувача;
- сповіщення про користувачів, які завершили або пропустили навчальні модулі.

Система буде налаштована для роботи через зручний інтерфейс зі зрозумілими меню та опціями. Адміністратори можуть легко створювати нові навчальні курси, призначати їх співробітникам та встановлювати терміни їх завершення. На основі даних, введених адміністраторами та користувачами,

система відображатиме звіти про хід виконання, статуси завершення курсів та показники ефективності для кожного учня. Адміністратори також зможуть коригувати курси, зміст або терміни на основі зворотного зв'язку в режимі реального часу.

Конфігурація розроблена для забезпечення безперебійного та персоналізованого процесу навчання, дозволяючи співробітникам брати участь у навчальних програмах у власному темпі, а адміністраторам – контролювати та оптимізувати загальну ефективність корпоративного навчального процесу. Завдяки зручному інтерфейсу та автоматичним сповіщенням, система дає змогу як користувачам, так і адміністраторам приймати рішення на основі даних та забезпечувати ефективне досягнення всіх навчальних цілей.

1.4 Функціональні та нефункціональні вимоги

Функціональні вимоги:

1. система повинна дозволяти створення, керування та видалення облікових записів користувачів (адміністраторів та співробітників);
2. адміністратори повинні мати можливість призначати кожному користувачеві певні ролі (наприклад, Адміністратор, Співробітник) з відповідними дозволами для керування контентом;
3. користувачі (співробітники) повинні мати доступ до своїх навчальних панелей, де вони можуть переглядати доступні курси, проходити тести та відстежувати свій прогрес;
4. адміністратори повинні мати можливість створювати, оновлювати та видаляти курси;

5. адміністратори повинні мати можливість створювати та керувати тестами, включаючи налаштування питань, варіантів відповідей та критеріїв оцінювання;
6. співробітники повинні мати доступ до доступних курсів та проходити пов'язані з ними тести в рамках свого навчання;
7. система повинна відстежувати прогрес співробітників у режимі реального часу, записуючи такі дані, як статус завершення курсу, результати тестів та загальна продуктивність;
8. адміністратори повинні мати доступ до звітів про прогрес співробітників, ефективність курсів та загальний успіх навчання;
9. співробітники повинні мати можливість бачити свій власний прогрес, включаючи завершені курси, майбутні завдання та досягнення.

Нефункціональні вимоги:

1. система повинна реагувати на дії користувача (наприклад, вхід, завантаження курсу, створення звіту) протягом кількох секунд, щоб забезпечити безперебійну роботу користувача;
2. система повинна підтримувати велику кількість одночасних користувачів (наприклад, співробітників, які одночасно проходять курси) без значного зниження продуктивності;
3. система повинна мати можливість масштабування, щоб враховувати зростаючу кількість користувачів, курсів та навчальних матеріалів у міру розширення організації;
4. вона повинна підтримувати додавання нових функцій та модулів без необхідності повного перероблення дизайну;
5. інтерфейс користувача має бути інтуїтивно зрозумілим та простим у навігації, дозволяючи як адміністраторам, так і співробітникам використовувати систему з мінімальним навчанням;

6. система повинна пропонувати функції доступності для користувачів з інвалідністю, гарантуючи, що кожен може ефективно взаємодіяти з системою;
7. система повинна бути доступною 99,9% часу, гарантуючи, що користувачі можуть отримати доступ до навчальних матеріалів у будь-який час, коли це необхідно [6];
8. вона повинна бути надійною, з мінімальним часом простою та без втрати даних, навіть у разі збоїв обладнання або програмного забезпечення.

Ці функціональні та нефункціональні вимоги гарантують, що система управління корпоративним навчанням є одночасно ефективною та результативною за своїм призначенням, зберігаючи при цьому високий рівень безпеки, зручності використання та продуктивності.

1.5 Вимоги до інтерфейсу користувача

Інтерфейс користувача (UI) Інформаційної системи для управління корпоративним навчанням має бути інтуїтивно зрозумілим, візуально привабливим та розробленим таким чином, щоб забезпечити безперебійну взаємодію з системою як адміністраторів, так і співробітників. Дизайн має бути пріоритетом простоти використання, забезпечуючи чітку та організовану структуру, яка дозволяє користувачам швидко отримувати доступ до ключових функцій, таких як курси, тести та звіти про ефективність.

Інтерфейс має бути простим, але ефективним, забезпечуючи легку навігацію користувачами, незалежно від їхньої ролі. Адміністраторам знадобиться швидкий доступ до розширених функцій, таких як створення курсів, керування користувачами та перегляд звітів, тоді як співробітники повинні мати легкий доступ до свого прогресу в навчанні, майбутніх завдань та доступних курсів. Персоналізована панель інструментів для кожної ролі користувача може зробити інформацію легкодоступною, з коротким оглядом

ключових даних, таких як завершення курсу, майбутні терміни або загальна ефективність.

Одним з ключових аспектів інтерфейсу користувача є його адаптивність. Система має бути сумісною з різноманітними пристроями, від настільних комп'ютерів та ноутбуків до планшетів та смартфонів. Це гарантує, що користувачі можуть взаємодіяти з системою, незалежно від того, чи перебувають вони в офісі чи в дорозі, без шкоди для зручності використання чи функціональності. Макет має бездоганно адаптуватися до різних розмірів екранів, зберігаючи чіткість та зручність використання на всіх пристроях [7].

Чіткі та добре розроблені кнопки заклику до дії є важливими для того, щоб допомогти користувачам користуватися системою. Ключові дії, такі як початок курсу, подання тесту або перегляд прогресу, повинні бути легко ідентифікованими та доступними. Ці кнопки повинні візуально виділятися, що спрощує для користувачів виконання необхідних дій без необхідності їх пошуку.

Форми, що використовуються для таких завдань, як створення курсу або подання відгуків, повинні бути простими для заповнення. Поля повинні бути чітко позначені, а система повинна надавати корисні підказки або пропозиції, щоб допомогти користувачам точно заповнити інформацію. Миттєвий зворотний зв'язок щодо помилок або відсутніх даних ще більше покращить взаємодію з користувачем, запобігаючи розчаруванню під час подання форми.

Щоб користувачі були в курсі подій, система повинна включати надійну систему сповіщень. Працівники повинні отримувати нагадування про важливі терміни або нові навчальні матеріали, а адміністратори повинні бути попереджені про будь-які оновлення системи або проблеми. Ці сповіщення повинні бути ненав'язливими, але помітними, гарантуючи, що користувачі залишатимуться в курсі подій, не відчуваючи себе перевантаженими.

2 МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 Загальні відомості

Уніфікована мова моделювання (UML) – це стандартизований інструмент, який використовується в програмній інженерії для візуалізації, специфікації, побудови та документування компонентів і поведінки системи. Він відіграє вирішальну роль у тому, щоб допомогти як розробникам, так і зацікавленим сторонам краще зрозуміти структуру та функціональність системи. UML надає широкий спектр діаграм, які можна розділити на дві основні категорії: структурні та поведінкові діаграми [8].

Структурні діаграми представляють статичні елементи системи, зосереджуючись на тому, як система організована та з яких компонентів вона складається. Ці діаграми описують внутрішню структуру та зв'язки між різними частинами системи. Діаграма класів, наприклад, є однією з найчастіше використовуваних діаграм в UML. Вона детально описує класи системи, їх атрибути, методи та зв'язки між цими класами, пропонуючи чітке уявлення про статичну структуру системи. Ще однією важливою діаграмою є діаграма об'єктів, яка ілюструє екземпляри класів у певний момент часу, показуючи стан та зв'язки об'єктів. Діаграма компонентів відображає організацію компонентів у системі, показуючи, як взаємодіють модулі, бібліотеки або служби. Тим часом, діаграми розгортання надають уявлення про фізичну архітектуру системи, ілюструючи, як компоненти розподілені між різними апаратними вузлами та як вони взаємодіють. Діаграми пакетів організовують системні класи в пакети, пропонуючи вищий рівень уявлення про залежності та зв'язки, що допомагає керувати складністю системи. Нарешті, діаграми складених структур описують внутрішню структуру класу та його взаємодію з іншими компонентами в системі.

Поведінкові діаграми, з іншого боку, зосереджуються на динамічних аспектах системи, фіксуючи, як компоненти взаємодіють та як система

поводиться з часом. Наприклад, діаграми варіантів використання надають чітке уявлення про те, як користувачі (або актори) взаємодіють із системою. Ці діаграми моделюють функціональні вимоги системи, зображуючи дії, які користувачі можуть виконувати. Ще однією важливою діаграмою є діаграма послідовності, яка ілюструє, як об'єкти взаємодіють у певному порядку, фіксуючи потік повідомлень між об'єктами та послідовність операцій. Діаграми діяльності зображують потік керування або даних у системі, моделюючи робочі процеси, процеси або бізнес-логіку та показуючи кроки, зроблені для виконання завдання або прийняття рішень. Діаграма станів описує різні стани, в яких може перебувати об'єкт або система, та як вона переходить між ними, що особливо корисно для моделювання життєвого циклу об'єктів. Діаграми зв'язку, хоча й подібні до діаграм послідовностей, більше зосереджені на зв'язках між об'єктами, а не на конкретному порядку взаємодій. Інший тип діаграми, діаграма огляду взаємодій, поєднує елементи діаграм активності та послідовності, надаючи ширше уявлення про те, як відбуваються взаємодії. Діаграми часу відображають поведінку компонентів з часом, що особливо корисно для візуалізації залежної від часу поведінки в системі [8].

UML пропонує кілька переваг, які покращують процес розробки програмного забезпечення. Він покращує чіткість, надаючи візуальне представлення системи, що полегшує всім залученим сторонам розуміння її структури та функціональності. UML сприяє ефективній комунікації, слугуючи універсальною мовою, яка дозволяє розробникам, дизайнерам та зацікавленим сторонам чітко висловлювати ідеї та вимоги. Він також допомагає в документації, слугуючи орієнтиром для майбутньої розробки або для залучення нових членів команди. Гнучкість UML дозволяє використовувати його на різних етапах життєвого циклу розробки програмного забезпечення, від раннього проєктування до остаточної реалізації. Використовуючи стандартизований набір символів та позначень,

UML забезпечує узгодженість між моделями та проєктами, що робить його надійним інструментом для розробки програмного забезпечення.

На завершення, UML – це універсальний та потужний інструмент, який відіграє ключову роль у представленні як структури, так і поведінки системи. Його комплексний набір діаграм забезпечує чітку основу для спілкування та співпраці дизайнерів, розробників та зацікавлених сторін, забезпечуючи добре організований та ефективний процес розробки.

2.2 Об’єктне та функціональне моделювання

2.2.1 Діаграма прецедентів. Діаграма варіантів використання — це тип діаграми UML (Уніфікованої мови моделювання), призначеної для ілюстрації функціональних аспектів системи, зосереджуючись на її взаємодії з користувачами або іншими зовнішніми сутностями. Мета цієї діаграми — забезпечити візуальне представлення функцій системи, підкреслюючи, як різні актори — користувачі чи інші системи — взаємодіють із системою, і які конкретні завдання чи операції виконуються у відповідь на ці взаємодії [9].

Актори на діаграмі варіантів використання представляють осіб, системи або пристрої, які взаємодіють із системою. Ці актори можуть бути первинними, такими як кінцеві користувачі, або вторинними, такими як зовнішні системи, які взаємодіють із системою за певних умов. З іншого боку, варіанти використання — це конкретні дії або функції, які система виконує, коли їх запускає актор. Наприклад, в системі онлайн-банкінгу варіанти використання можуть включати такі дії, як «Вхід», «Переказ коштів» або «Перегляд балансу рахунку».

Діаграма також відображає зв'язки між акторами та варіантами використання, що допомагає передати взаємодії в системі. Ці зв'язки представлені за допомогою різних типів конекторів, таких як асоціації, які просто пов'язують актора з варіантом використання, щоб показати, що актор

може ініціювати цей варіант використання. Інший тип зв'язку - це "включення", де один варіант використання інтегрує функціональність іншого, тобто певний процес завжди включатиме виконання іншого завдання. Існує також зв'язок "розширення", який вказує на те, що варіант використання може розширити або покращити функціональність іншого варіанту використання, але лише за певних умов.

Розроблена діаграма прецедентів використання представлена на рис. 3.

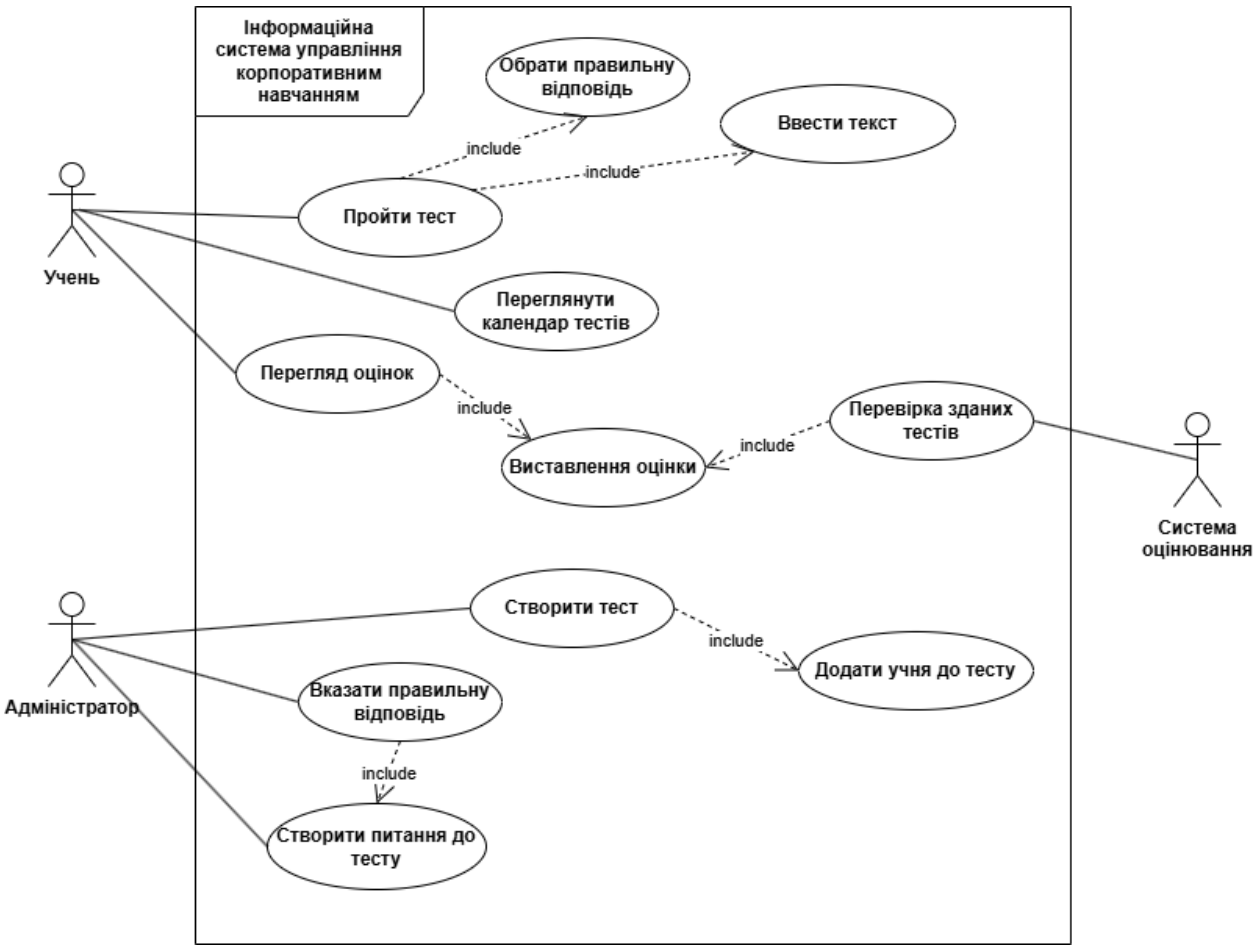

Рис. 3 Діаграма прецедентів

Створена діаграма прецедентів містить акторів:

- “Учень”;
- “Адміністратор”;
- “Система оцінювання”.

Актор «Учень» включає такі прецеденти:

- пройти тест;
- обрати правильну відповідь;
- ввести текст;
- переглянути календар тестів;
- перегляд оцінок;

Актор «Адміністратор» включає такі прецеденти:

- створити тест;
- додати учня до тесту;
- створити питання до тесту;
- вказати правильну відповідь.

Актор «Система оцінювання» включає такі прецеденти:

- перевірка зданих тестів;
- виставлення оцінки.

Прецеденти певним чином залежать одне від одного.

Розглянемо детальніше вищеописані прецеденти.

Назва випадку використання: Пройти тест

Актор: Учень

Опис: Цей варіант використання описує процес, за допомогою якого студент отримує доступ, завершує та надсилає тест у корпоративній системі навчання. Студент взаємодіє із системою, щоб пройти тест, отримати зворотний зв'язок та зрозуміти свою успішність.

Основний потік:

1. учень входить у систему, використовуючи свої облікові дані;
2. учень переходить до розділу навчання та вибирає тест, який він хоче пройти;
3. система відображає деталі тесту, включаючи назву, тривалість та інструкції;
4. учень починає тест, відповідаючи на запитання. Система може дозволити студенту перемикатися між запитаннями;

5. учень відповідає на всі запитання та натискає кнопку «Надіслати» після завершення;
6. система оцінює відповіді студента та підраховує бал;
7. система надає студенту зворотний зв'язок, включаючи бал та будь-які коментарі чи пояснення;
8. якщо учень складає тест, система повідомляє його про успіх, і він може отримати сертифікат або значок;
9. якщо учень не складає тест, система повідомляє його про невдачу, і йому може бути надана можливість перескласти тест або отримати доступ до додаткових навчальних матеріалів.

Альтернативні потоки:

1. якщо учень вирішує пропустити питання, система дозволяє йому перейти до наступного питання, не перевіряючи його. Студент може повернутися до пропущеного питання пізніше, перш ніж здати тест;
2. якщо учень не завершить тест у відведений час, система автоматично здає тест і надає остаточний бал. Студент отримує повідомлення про те, що час тестування закінчився;
3. якщо учень не складає тест, система може запропонувати можливість перескласти тест негайно або після отримання доступу до додаткових навчальних матеріалів.

Розглянемо ще один сценарій використання.

Назва випадку використання: Створення тесту

Актор: Адміністратор

Опис: Цей варіант використання описує процес, за допомогою якого адміністратор створює новий тест у корпоративній системі управління навчанням. Тест буде призначено студентам і використовуватиметься для оцінки їхніх знань з певної теми.

Основний потік:

1. адміністратор входить у систему, використовуючи свої облікові дані;
2. адміністратор переходить до розділу «Керування тестами» в інтерфейсі системи;
3. адміністратор вибирає опцію створення нового тесту;
4. система пропонує адміністратору ввести інформацію;
5. для кожного питання адміністратор надає можливі відповіді (якщо такі є), вибирає правильну відповідь і визначає правила оцінювання;
6. адміністратор встановлює критерії успішного складання тесту (наприклад, мінімальний бал, необхідний для успішного складання тесту);
7. адміністратор може переглянути тест, щоб переконатися, що вміст правильний, а питання відформатовані належним чином;
8. адміністратор зберігає тест, роблячи його доступним для складання студентами;
9. система підтверджує створення тесту та додає його до списку доступних тестів у навчальному модулі;
10. адміністратор може призначити тест певним студентам або зробити його доступним для всіх студентів певного курсу чи навчальної програми.

Альтернативні потоки:

1. якщо адміністратор вводить недійсні або неповні дані (наприклад, відсутня назва, питання або тривалість), система відобразить повідомлення про помилку та запропонує адміністратору виправити введені дані, перш ніж продовжити;
2. якщо адміністратор зіткнеться з помилкою під час попереднього перегляду тесту (наприклад, непрацюючі посилання, проблеми з форматуванням), система дозволить адміністратору виправити ці проблеми перед завершенням тесту;

3. якщо адміністратор вирішить не створювати тест після початку процесу, система запропонує можливість скасувати створення, відкинувши будь-яку незбережену інформацію.

2.2.2 Діаграма послідовності. Діаграма послідовності – це тип діаграми UML (Уніфікована мова моделювання), яка моделює потік повідомлень або взаємодій між різними компонентами чи об'єктами системи з плином часу. Її головна мета – показати, як процеси взаємодіють один з одним і в якому порядку. Діаграми послідовності особливо корисні для розуміння складної поведінки системи та для візуалізації взаємодії між користувачами, підсистемами та потоками даних [8].

На діаграмі послідовності вертикальна вісь представляє час, що просувається зверху вниз, тоді як горизонтальна вісь показує різні об'єкти або учасників, які беруть участь у взаємодії. Кожен учасник зображений як лінія життя – вертикальна пунктирна лінія – а його дії або виклики методів представлені горизонтальними стрілками між лініями життя. Ці стрілки позначені назвою повідомлення або виклику функції, і вони часто містять параметри, що передаються.

Наприклад, у корпоративній системі навчання діаграма послідовності може ілюструвати, як студент ініціює тест, як система отримує тестові питання з бази даних, і як модуль оцінювання обробляє відповіді та повертає бал. Кожен крок, включаючи додаткові або альтернативні потоки, такі як тайм-аути або обробка помилок, може бути представлений на цій діаграмі, щоб показати детальний огляд логіки взаємодії. Загалом, діаграми послідовностей допомагають розробникам програмного забезпечення, аналітикам та зацікавленим сторонам чітко розуміти послідовність операцій та шляхів зв'язку, що є критично важливим для проєктування, документування та перевірки функціональності системи.

Діаграма послідовності, зображена на рис. 4, включає наступні об'єкти:

- “Учень”;

- “Адміністратор”;
- “Тест”;
- “Система оцінення”.

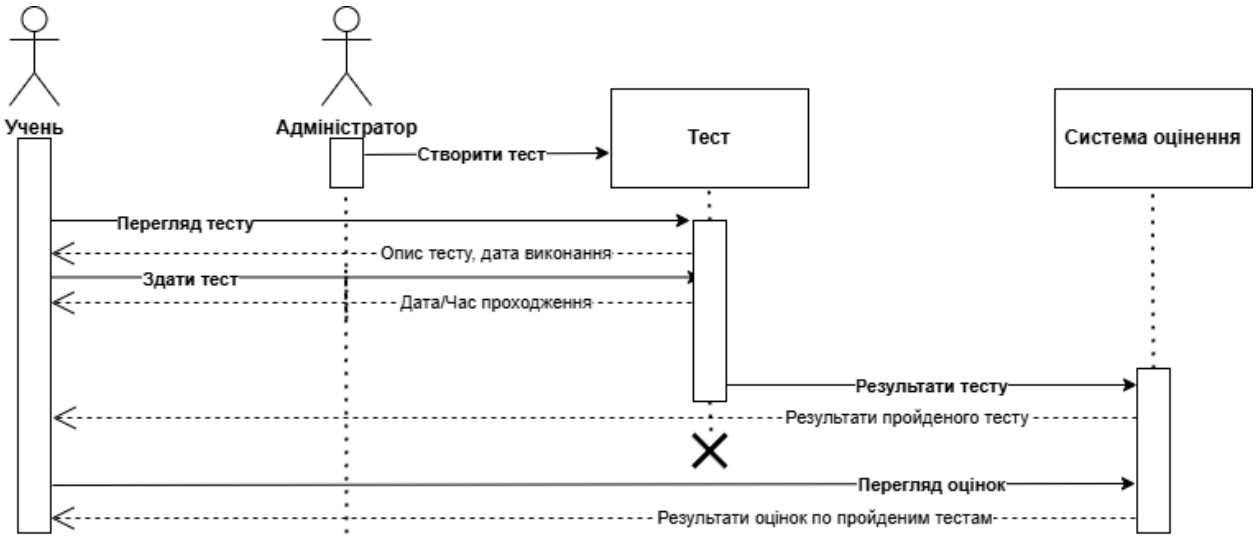

Рис. 4 Діаграма послідовності

Ця діаграма послідовності ілюструє взаємодію між студентом, адміністратором, тестом та системою оцінювання. Вона показує, як тест створюється, проходить, оцінюється та перевіряється. Адміністратор ініціює процес, створюючи тест, після чого студент отримує доступ до його деталей, включаючи опис та терміни. Після завершення тестування студент завершує та надсилає його, вказуючи відповідний час виконання. Потім тест передає результати до системи оцінювання, яка обробляє дані та надсилає оцінені бали. Нарешті, студент переглядає оцінки та отримує уявлення про свою успішність. Ця діаграма ефективно відображає основний потік інформації, забезпечуючи чітке розуміння механізмів, що лежать в основі системи тестування та оцінювання.

2.2.3 Діаграма активності. Діаграма діяльності – це візуальний інструмент, який використовується в UML (Уніфікована мова моделювання) для представлення динамічної поведінки системи шляхом окреслення її робочого процесу або бізнес-процесів. Замість того, щоб зосереджуватися на структурі, вона фіксує, як завдання виконуються та взаємодіють, підкреслюючи потік керування або даних між різними

кроками. Цей тип діаграми особливо корисний під час моделювання складних сценаріїв, таких як потоки варіантів використання, поведінка методів або системні операції [9].

Структура діаграми діяльності зазвичай включає кілька ключових елементів. Окремі завдання або операції, відомі як дії, зображуються у вигляді заокруглених прямокутників і представляють дії, що виконуються користувачем або системою. Діаграма починається із суцільного чорного кола, що вказує на початок процесу, і закінчується обведеним колом, що позначає його завершення. Коли потік розгалужується залежно від певних умов або рішень, використовуються вузли у формі ромба. Стрілки з'єднують усі ці компоненти, щоб вказати логічний порядок виконання. У складніших діаграмах можна додавати доріжки, щоб призначити певні обов'язки певним користувачам або компонентам системи, допомагаючи уточнити, хто відповідає за кожен крок.

Наприклад, у контексті системи управління корпоративним навчанням діаграма діяльності може детально описувати кроки, які студент виконує для завершення тесту. Це може включати вхід у систему, вибір тесту, відповідь на кожне запитання, надсилання відповідей та перегляд кінцевого результату. Діаграма також може виділяти альтернативні шляхи,

наприклад, що станеться, якщо тест закінчиться або надсилання буде недійсним.

Розроблена діаграма активності представлена на рис. 5

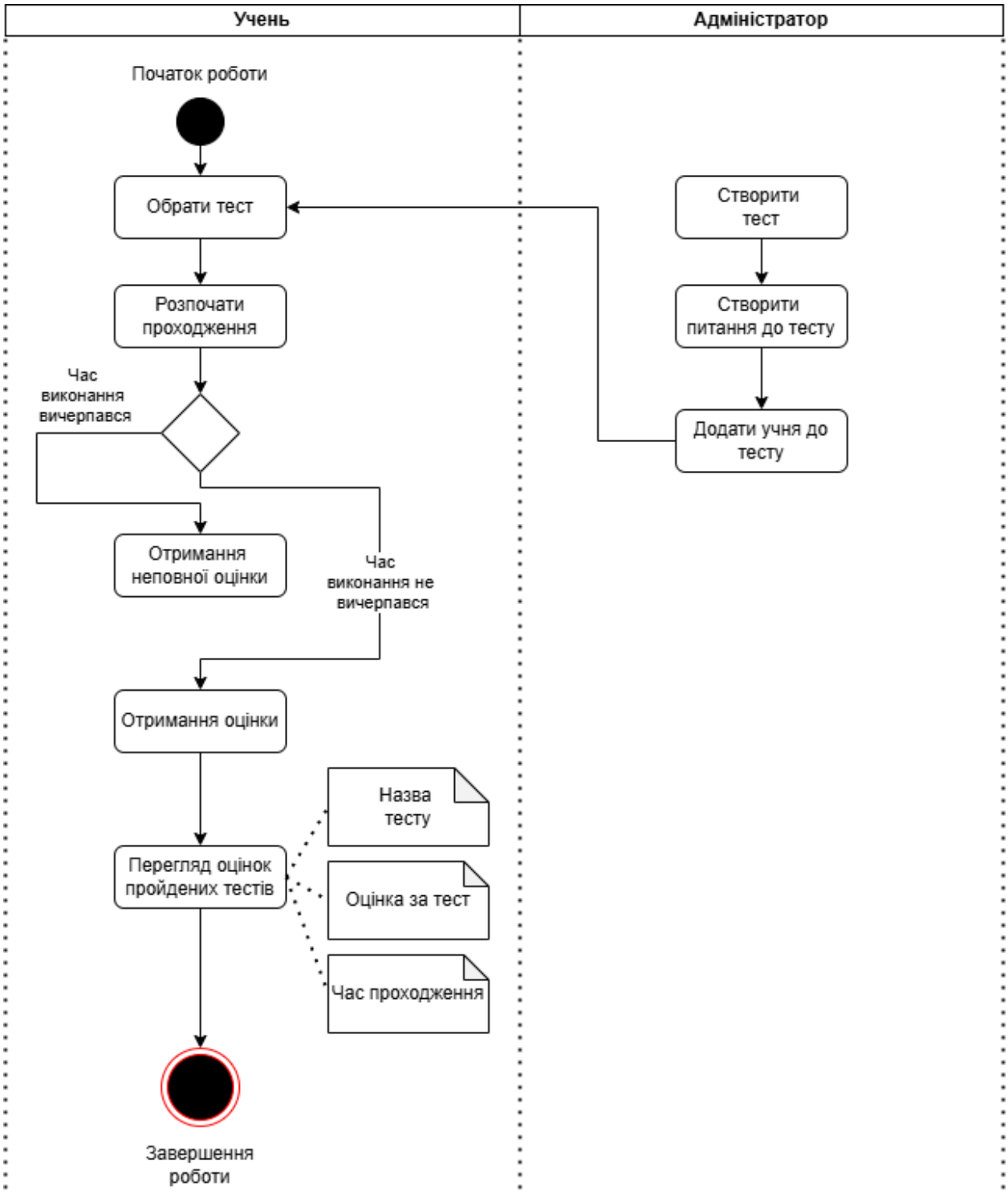

Рис. 5 Діаграма активності

Ця діаграма активності відображає процеси, що стосуються виконання тестів студентами та адміністрування тестування. Вона поділена на дві частини: учень і адміністратор.

Учень починає з вибору тесту, після чого розпочинає його проходження. Якщо час тестування вичерпано, він отримує неповну оцінку, а якщо ні — оцінку за тест. Після цього студент має можливість переглянути свої результати, включаючи назву тесту, отримані бали та час проходження. На цьому цикл завершується.

Адміністратор, у свою чергу, створює тест, додає до нього питання та включає учня до списку учасників. Цей процес формує структуру тестування, забезпечуючи його коректне проведення.

Ця діаграма дає чітке уявлення про порядок дій у системі тестування, що допомагає зрозуміти механізми її функціонування.

2.3 Абстракції предметної області

У розробці інформаційної системи для управління корпоративним навчанням, абстракції предметної області відіграють ключову роль у визначенні основних компонентів та їх взаємозв'язків. Ці абстракції представляють реальні концепції в навчальному середовищі та допомагають структурувати архітектуру системи логічним та керованим чином.

В основі системи лежить абстракція Користувача, яка охоплює дві основні ролі: Адміністратора та Студента. Адміністратори відповідають за управління навчальним контентом та контроль за прогресом студентів, тоді як студенти взаємодіють із системою, виконуючи тести та відстежуючи свої результати.

Ще однією центральною абстракцією є сутність Тест. Тест включає набір питань, кожне з яких потенційно пов'язане з кількома варіантами відповідей, з однією або кількома правильними відповідями. Тести

створюються адміністраторами та призначаються певним студентам або групам, формуючи основну освітню взаємодію в системі.

Абстракція Результату фіксує результат взаємодії студента з тестом. Вона зберігає бал, час виконання та, можливо, детальний зворотний зв'язок. Ці дані є важливими для оцінки успішності студентів та виявлення прогалин у навчанні.

Крім того, існують допоміжні абстракції, такі як навчальний модуль або курс, які групують тести в ширші тематичні або предметні колекції, що дозволяє структурувати навчальний шлях. Кожен модуль може містити навчальні матеріали та послідовність оцінювань.

Визначаючи ці абстракції предметних областей – користувачів, тести, питання, відповіді, результати та навчальні модулі – система отримує чітку концептуальну модель, яка спрощує розробку, забезпечує узгодженість та підтримує масштабованість у міру впровадження нових функцій. Ці абстракції відображають реальні процеси корпоративного навчання та забезпечують основу для створення інтуїтивно зрозумілого, функціонального та ефективного програмного забезпечення.

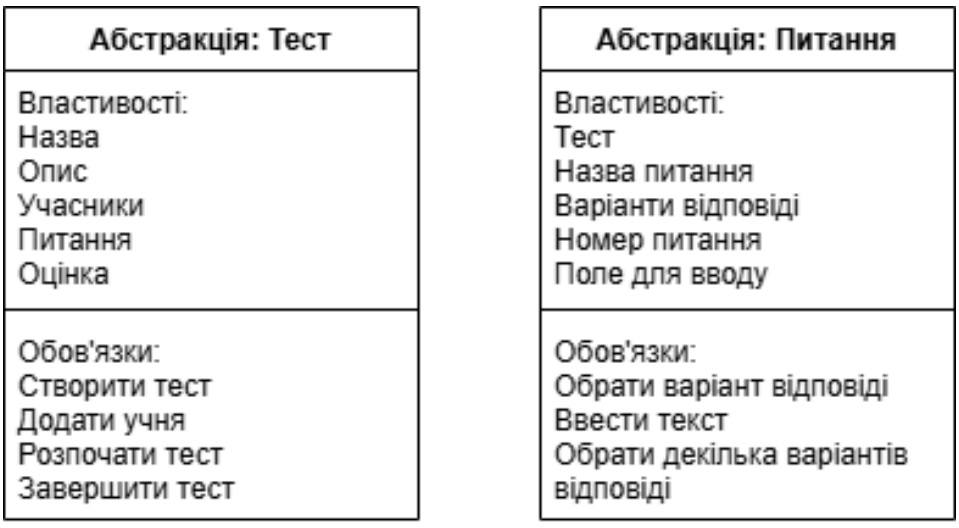

Рис. 6 Абстракції предметної області

На діаграмі представлено дві ключові абстракції предметної області: "Тест" і "Питання", кожна з яких має набір властивостей та обов’язків.

"Тест" містить назву, опис, список учасників, набір питань та оцінку результатів. Його основні функції включають створення тесту, додавання учнів, запуск тестування та завершення тесту після його проходження.

"Питання", у свою чергу, прив’язане до тесту та має назву, варіанти відповіді, номер у тесті та поле для введення тексту. Основні обов’язки цього елемента включають вибір відповіді, введення тексту та можливість вибору декількох варіантів відповідей.

Ця структура забезпечує логічну організацію процесу тестування, встановлюючи чіткі межі відповідальності між об'єктами системи.

2.4 Діаграма класів

Діаграма класів – це основний компонент UML (Уніфікованої мови моделювання), який забезпечує статичне представлення структури системи. Вона візуально представляє класи в системі разом з їхніми атрибутами, методами (операціями) та зв'язками між ними. Діаграми класів слугують кресленням для розробників, пропонуючи детальний огляд того, як різні частини системи взаємодіють та залежать одна від одної [10].

У контексті інформаційної системи для управління корпоративним навчанням діаграма класів зазвичай включає ключові сутності, такі як Користувач, Тест, Запитання, Відповідь та Результат. Кожен клас визначає певну роль у системі. Наприклад, клас Користувач може бути розділений на спеціалізовані ролі, такі як Адміністратор та Студент, кожна з яких має свої власні атрибути (такі як ім'я, електронна пошта або роль) та методи (такі як login або viewResults).

Клас Тест містив би такі атрибути, як заголовок, опис та тривалість, і був би пов'язаний з кількома об'єктами Запитання. Кожне Запитання може бути пов'язане з кількома варіантами відповідей, індикатори яких є правильними. Клас Результат представляв би результат спроби студента, зберігаючи бал та дату спроби.

Зв'язки між класами є критично важливими: асоціації, узагальнення (спадкування) та залежності ілюструють, як компоненти пов'язані один з одним та залежать один від одного. Наприклад, зв'язок «один до багатьох» може існувати між класами Test та Question, тоді як Student може успадковувати від більш загального класу User.

Організовуючи компоненти системи візуальним та логічним чином, діаграми класів допомагають розробникам зрозуміти та ефективно впроваджувати систему. Вони також допомагають виявляти потенційні проблеми на ранніх етапах проєктування та забезпечують послідовну комунікацію між членами команди під час процесу розробки програмного забезпечення.

Для цього проєкту було створено спеціальну діаграму класів з використанням об’єктно-орієнтованого підходу (рис. 7).

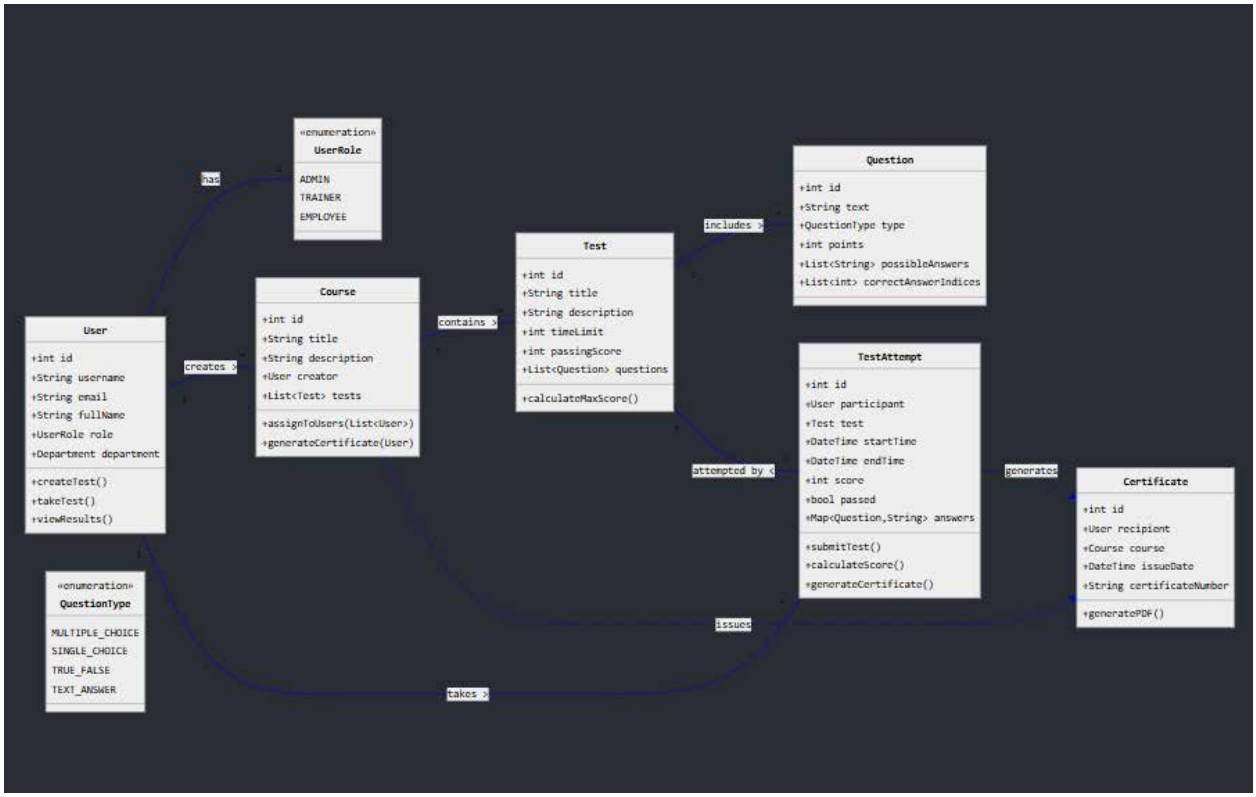

Рис. 7 Діаграма класів

Ця діаграма класів ілюструє структуру системи управління навчанням, визначаючи основні компоненти та їх взаємодію. В основі лежить клас User

(Користувач), який включає такі атрибути, як ID (Ідентифікатор), ім'я, електронна пошта, роль та приналежність до відділу. Користувачі можуть створювати та проходити тести, переглядати результати та брати участь у курсах. Клас Course (Курс) надає такі деталі, як назва, опис, автор та колекція тестів. Він керує призначенням користувачів та сприяє сертифікації після успішного завершення.

Клас Test (Тест) інкапсулює ключові параметри тесту, включаючи його назву, опис, тривалість та список питань. Він тісно пов'язаний з класом TestAttempt (Тестова спроба), який відстежує деталі участі, такі як користувач, який проходив тест, час початку та завершення, отриманий бал та чи було тест успішно завершено. Клас Question (Питання) обробляє окремі тестові запити, вказуючи текст, тип, варіанти відповідей та правильні варіанти.

Сертифікація представлена класом Certificate (Сертифікат), який зберігає інформацію про одержувачів, пов'язаний курс, дату видачі та унікальні номери сертифікації. Додаткові класифікації уточнюють ролі користувачів до адміністратора, тренера та співробітника, тоді як типи питань варіюються від відповідей з вибором відповідей та текстових відповідей до форматів «істина/неправда».

Ця діаграма надає структуроване уявлення про архітектуру системи, демонструючи логічні зв'язки між ключовими елементами. Вона встановлює чітку основу для управління тестуванням, оцінюванням та сертифікацією, забезпечуючи організований та ефективний навчальний процес.

3 ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 Логічна модель даних

Логічна модель даних забезпечує чітке та структуроване представлення даних у системі, зосереджуючись на значенні та взаємозв'язках інформації, а не на технічній реалізації. Вона визначає основні сутності, їхні атрибути та те, як вони пов'язані один з одним, що робить її важливим кроком у перетворенні бізнес-вимог у дизайн бази даних. На відміну від фізичних моделей, логічні моделі не залежать від систем управління базами даних, що підкреслює концептуальне розуміння потоку та структури даних [11].

У контексті корпоративної системи управління навчанням ця модель допомагає проілюструвати, як взаємопов'язані елементи даних, такі як користувачі, тести, запитання та результати. Система включає два основні типи користувачів: адміністраторів та студентів. Адміністратори відповідають за створення та керування тестами, які включають набір запитань. Кожне запитання, у свою чергу, має кілька можливих відповідей, деякі з яких позначені як правильні. Студенти взаємодіють із системою, отримуючи доступ до цих тестів, надсилаючи свої відповіді та отримуючи результати на основі своїх результатів.

Кожна сутність у моделі, така як Користувач, Тест, Запитання, Відповідь та Результат, містить певні атрибути, які визначають її властивості. Наприклад, користувач має ім'я, електронну пошту та роль; тест включає посаду та обмеження за часом; Результат відображає бали студента та час виконання. Зв'язки між цими сутностями пояснюють, як вони залежать один від одного. Наприклад, один адміністратор може керувати кількома тестами, кожен тест пов'язаний з кількома питаннями, і кожен студент може мати кілька результатів залежно від пройдених тестів.

Логічно організовуючи структуру даних, ця модель забезпечує узгодженість, підтримує точну поведінку системи та готує основу для

створення надійної фізичної бази даних. Вона також покращує комунікацію між розробниками, аналітиками та зацікавленими сторонами, пропонуючи спільний, технологічно-незалежний погляд на інформаційний ландшафт системи.

Логічна модель системи представлена на рисунку 8.

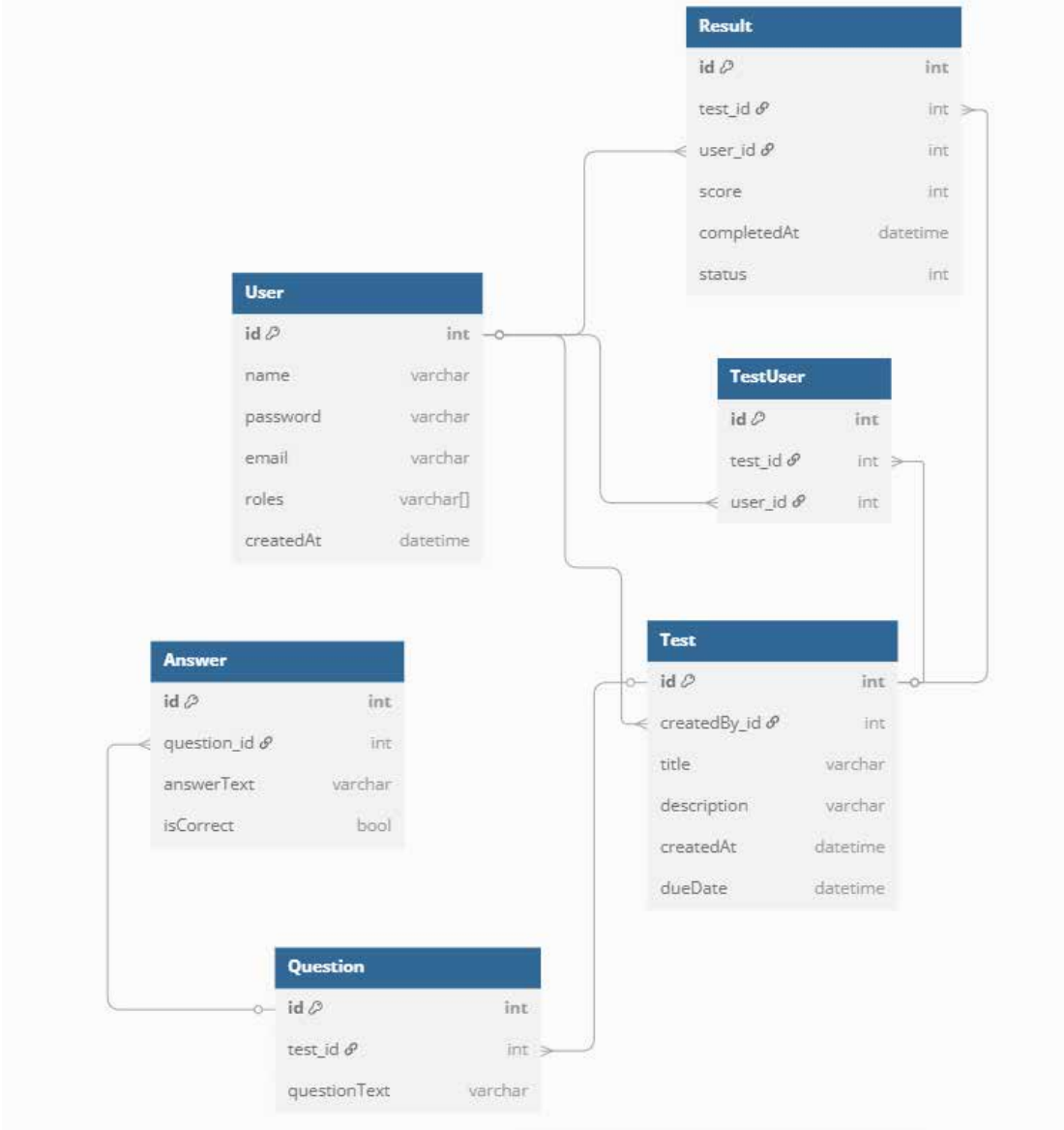

Рис. 8 ER-діаграма

User

Ця таблиця містить інформацію про користувачів. Кожен користувач має унікальний ідентифікатор (id), а також атрибути, пов'язані з його обліковим записом.

Поля:

1. id (int): Первинний ключ, автоматично збільшується;
2. name (varchar): Ім'я користувача;
3. password (varchar): Пароль користувача;
4. email (varchar, nullable): Адреса електронної пошти користувача;
5. roles (varchar[]): Масив ролей, призначених користувачеві (наприклад, "admin", "user");
6. createdAt (datetime): Дата та час створення користувача.

Test

Ця таблиця зберігає інформацію про тести, створені користувачами. Кожен тест пов'язаний з автором (createdBy_id, який є зовнішнім ключем до таблиці User).

Поля:

1. id (int): Первинний ключ, автоматично збільшується;
2. createdBy_id (int): Зовнішній ключ, що посилається на ідентифікатор користувача, який створив тест (посилається на User.id);
3. title (varchar): Назва тесту;
4. description (varchar, nullable): Опис тесту;
5. createdAt (datetime): Дата та час створення тесту;
6. dueDate (datetime, nullable): Дата виконання тесту.

Зв'язки:

1. багато до одного: Кожен тест створюється одним користувачем. (Test.createdBy_id → User.id).

TestUser

Ця таблиця використовується для відстеження того, які користувачі призначені до яких тестів. Вона діє як таблиця зв'язків «багато до багатьох» між User та Test.

Поля:

2. id (int): Первинний ключ, автоматично збільшується;

3. test_id (int): Зовнішній ключ, що посилається на ідентифікатор тесту (посилається на Test.id);
4. user_id (int): Зовнішній ключ, що посилається на ідентифікатор користувача (посилається на User.id).

Зв'язки:

1. багато до одного: Кожен запис TestUser пов'язує користувача з тестом;
2. багато до одного: Кожен запис TestUser пов'язує тест з користувачем.

Question

У цій таблиці зберігаються питання, пов'язані з тестом. Кожне питання належить до одного конкретного тесту.

Поля:

1. id (int): Первинний ключ, автоматично збільшується;
2. test_id (int): Зовнішній ключ, що посилається на ідентифікатор тесту (посилається на Test.id);
3. questionText (varchar): Текст питання.

Зв'язки:

1. багато до одного: Кожне питання належить до одного тесту (Question.test_id → Test.id).

Answer

У цій таблиці зберігаються можливі відповіді на кожне питання. Кожна відповідь пов'язана з одним питанням.

Поля:

1. id (int): Первинний ключ, автоматично збільшується;
2. question_id (int): Зовнішній ключ, що посилається на ідентифікатор питання (посилається на Question.id);
3. answerText (varchar): Текст відповіді;
4. isCorrect (bool): Логічне значення, що вказує, чи є відповідь правильною.

Зв'язки:

1. багато до одного: Кожна відповідь належить до одного питання (Answer.question_id → Question.id).

Result

У цій таблиці зберігаються результати тесту, проведеного користувачами. Кожен результат пов'язаний з тестом та користувачем, а також з отриманим ним балом.

Поля:

1. id (int): Первинний ключ, автоматично збільшується;
2. test_id (int): Зовнішній ключ, що посилається на ідентифікатор тесту (посилається на Test.id);
3. user_id (int): Зовнішній ключ, що посилається на ідентифікатор користувача (посилається на User.id);
4. score (int): Бал, отриманий користувачем;
5. completedAt (datetime, nullable): Дата та час завершення тесту;
6. status (int): Статус результату (наприклад, 1 для завершеного, 0 для незавершеного).

Зв'язки:

1. багато до одного: Кожен результат пов'язаний з одним тестом (Result.test_id → Test.id);
2. багато до одного: Кожен результат пов'язаний з одним користувачем (Result.user_id → User.id).

3.2 Вибір системи управління базою даних та її реалізація

Під час розробки інформаційної системи, особливо тієї, що призначена для управління корпоративним навчанням, вибір системи керування базами даних (СКБД) стає одним із найважливіших рішень. СКБД виступає центральним компонентом для ефективної організації, зберігання та

пошуку даних. Вибір правильної системи вимагає чіткого розуміння характеру програми та того, як вона буде використовуватися [12].

Для структурованої системи, де користувачі, навчальні модулі, результати тестів та ролі чітко визначені, реляційна база даних зазвичай є найбільш підходящим вибором. Такі системи пропонують добре налагоджені моделі для управління структурованими даними та забезпечують механізми забезпечення узгодженості та цілісності. Також необхідно враховувати здатність обробляти зростаючі обсяги користувачів і даних, особливо якщо очікується, що платформа з часом зростатиме. У цих сценаріях СКБД повинна мати можливість плавного масштабування без шкоди для продуктивності.

Не менш важливою є швидкість обробки запитів системою, особливо коли потрібне часте отримання даних, таке як створення звітів або відображення результатів тестування. Розширене індексування та ефективна оптимізація запитів значно підвищують продуктивність та зручність використання. З точки зору безпеки, СУБД повинна пропонувати надійне керування ролями та контроль доступу до даних для підтримки різних рівнів дозволів, гарантуючи, що кожен користувач взаємодіє лише з тими даними, до яких він має доступ.

Транзакційна підтримка є ще однією важливою функцією, оскільки багато корпоративних процесів вимагають надійності та точності. Незалежно від того, чи завершує співробітник навчальний модуль, чи тренер оновлює навчальну програму, ці операції повинні оброблятися таким чином, щоб гарантувати точність даних навіть у разі перерв.

Більше того, сумісність СУБД з існуючим стеком розробки відіграє важливу роль в оптимізації робочого процесу проєкту. Системи, побудовані на таких фреймворках, як Laravel або Symfony, наприклад, безперешкодно інтегруються з добре підтримуваними реляційними базами даних. Окрім технічних характеристик, не менш важливими є практичні аспекти, такі як ліцензування, витрати на обслуговування, підтримка спільноти та доступність документації. Рішення з відкритим кодом, такі як MySQL або PostgreSQL,

часто є кращими завдяки своїй гнучкості, надійності та активним спільнотам користувачів.

Під час розробки інформаційної системи для управління корпоративним навчанням рішення використовувати MySQL як систему управління базами даних було прийнято на основі поєднання практичних, технічних та стратегічних факторів.

Перш за все, MySQL — це надійна та перевірена часом реляційна СУБД, яка пропонує всі основні функції, необхідні для управління структурованими даними. Система, що розробляється, передбачає чітку структуру сутностей, таких як користувачі, навчальні модулі, тести, результати та ролі, — всі з яких природно підходять для реляційної моделі даних. Дотримання MySQL стандарту SQL робить його ідеальним інструментом для управління цими зв'язками та забезпечення цілісності даних за допомогою обмежень та ключів [13].

Ключовою перевагою MySQL є його продуктивність та масштабованість. Зі зростанням кількості користувачів та обсягу збережених даних у системі важливо, щоб база даних могла обробляти зростаюче навантаження без шкоди для швидкості. MySQL оптимізовано для операцій з великим обсягом читання та може ефективно обслуговувати динамічний контент, такий як результати тестів, відстеження прогресу та панелі інструментів користувачів — критично важливі функції в навчальній платформі.

Ще одним важливим фактором вибору MySQL є його широка підтримка та можливості інтеграції. Він плавно інтегрується з популярними бекенд-технологіями, такими як PHP, Laravel або Symfony, які зазвичай використовуються в розробці веб-додатків. Ця сумісність скорочує час розробки та забезпечує стабільний зв'язок між прикладним рівнем та базою даних.

На вибір також вплинув відкритий вихідний код MySQL, що означає відсутність ліцензійних зборів, що робить його економічно ефективним

рішенням. Для студентського проєкту або впровадження в малому та середньому підприємстві уникнення додаткових витрат на програмне забезпечення, водночас покладаючись на зрілий продукт, є значною перевагою. Крім того, екосистема MySQL підтримується великою, активною спільнотою та обширною документацією, що надає цінну допомогу під час розробки та налагодження.

Безпека є ключовим аспектом корпоративних систем, особливо при роботі з даними користувачів та оцінці продуктивності. MySQL підтримує детальний контроль доступу, зашифровані з'єднання та механізми резервного копіювання, що сприяє створенню безпечного та стійкого додатка.

MySQL пропонує надійний набір інструментів для адміністрування, налаштування продуктивності та резервного копіювання, що спрощує обслуговування системи та забезпечує довгострокову стійкість.

Підсумовуючи, MySQL було обрано для цього проєкту, оскільки він забезпечує оптимальний баланс надійності, продуктивності, масштабованості, безпеки та економічної ефективності, що робить його добре придатним для потреб корпоративної системи управління навчанням.

Таким чином було створено базу даних в середовищі MySQL Workbench (Рис. 9).

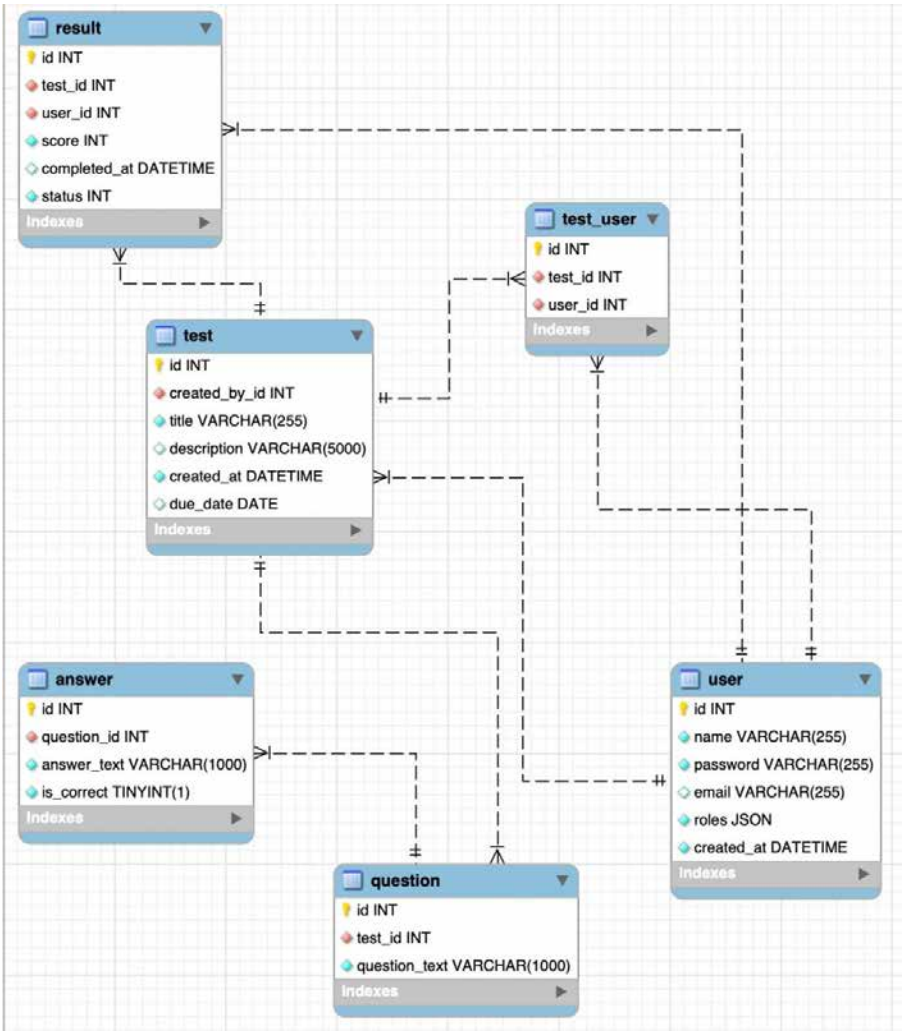

Рис. 9 База даних

Ця база даних побудована для керування тестуванням і містить шість основних таблиць: result, test_user, test, answer, question та user. Їхні взаємозв’язки визначають логіку роботи системи.

Таблиця user містить ключову інформацію про користувачів: ідентифікатор, ім’я, електронну пошту, пароль, ролі та дату створення акаунта. Вона пов’язана з тестами та результатами, дозволяючи визначати авторів тестів і відстежувати оцінки. Таблиця test містить загальні параметри тесту, як-от назва, опис, час створення та дедлайн виконання. Вона взаємодіє з питаннями та учасниками тестування.

Question прив’язана до тесту й зберігає текст питання разом із його унікальним ідентифікатором. Відповіді на питання записуються в таблиці answer, де кожен варіант відповіді має власний текст та позначку правильності. test_user виступає проміжною сутністю між тестами та користувачами,

реєструючи участь у тестуванні. Таблиця result зберігає результати проходження тестів, включаючи набраний бал, статус проходження та дату завершення тесту.

Завдяки такій структурі база даних ефективно керує інформацією про користувачів, тестування та оцінювання, забезпечуючи збереження всіх важливих даних у процесі навчання.

3.3 Архітектура програмного забезпечення

Архітектура інформаційної системи для управління корпоративним навчанням розроблена для забезпечення модульності, масштабованості та зручності обслуговування, а також для забезпечення безперебійного користувацького досвіду. Для досягнення цих цілей система дотримується багаторівневої архітектури, заснованої на розділенні завдань, яка розділяє програму на окремі рівні: презентація, бізнес-логіка та доступ до даних [14].

На рівні презентації користувачі взаємодіють із системою через адаптивний веб-інтерфейс. Цей інтерфейс розроблений таким чином, щоб бути інтуїтивно зрозумілим та доступним на різних пристроях. Залежно від їхньої ролі — наприклад, адміністратор, тренер або співробітник — користувачам надаються персоналізовані панелі інструментів та функції, включаючи реєстрацію на курси, участь у тестуванні, відстеження ефективності та управління контентом.

Рівень бізнес-логіки служить ядром програми, обробляючи запити, забезпечуючи дотримання правил та обробляючи всі операції, пов'язані з навчанням. Цей рівень керує такими функціями, як автентифікація користувачів, контроль доступу на основі ролей, оцінювання тестів, оцінка прогресу та автоматичні сповіщення. Він діє як місток між користувацьким інтерфейсом та базою даних, гарантуючи, що всі дії відповідають визначеним правилам та робочим процесам.

Під бізнес-рівнем знаходиться рівень доступу до даних, який обробляє всі взаємодії з базою даних. У цьому проєкті MySQL слугує системою керування реляційними базами даних. Структура бази даних ретельно нормалізована для підтримки ефективних запитів та збереження цілісності даних. Основні сутності, такі як користувачі, курси, тести, запитання, відповіді, результати та ролі, взаємопов'язані за допомогою зовнішніх ключів, що забезпечує узгодженість та логічну узгодженість на всій платформі.

Система реалізована з використанням сучасних технологій веб-розробки, з PHP (Symfony або Laravel) як бекенд-фреймворком та HTML/CSS/JavaScript для фронтенду. Цей вибір забезпечує високу сумісність з MySQL та забезпечує гнучке середовище розробки. API також можуть використовуватися для забезпечення інтеграції із зовнішніми сервісами або мобільними додатками в майбутньому.

Безпека є невід'ємною частиною архітектури. Система реалізує автентифікацію, зашифровану передачу даних (HTTPS) та дозволи користувача для захисту конфіденційних навчальних даних та особистої інформації.

Архітектура розроблена з урахуванням розширюваності, що дозволяє майбутні доповнення, такі як аналітика в режимі реального часу, модулі зворотного зв'язку або інтеграція з HR-системами. Завдяки чіткому розподілу обов'язків між компонентами, система залишається легкою в обслуговуванні, тестуванні та розширенні.

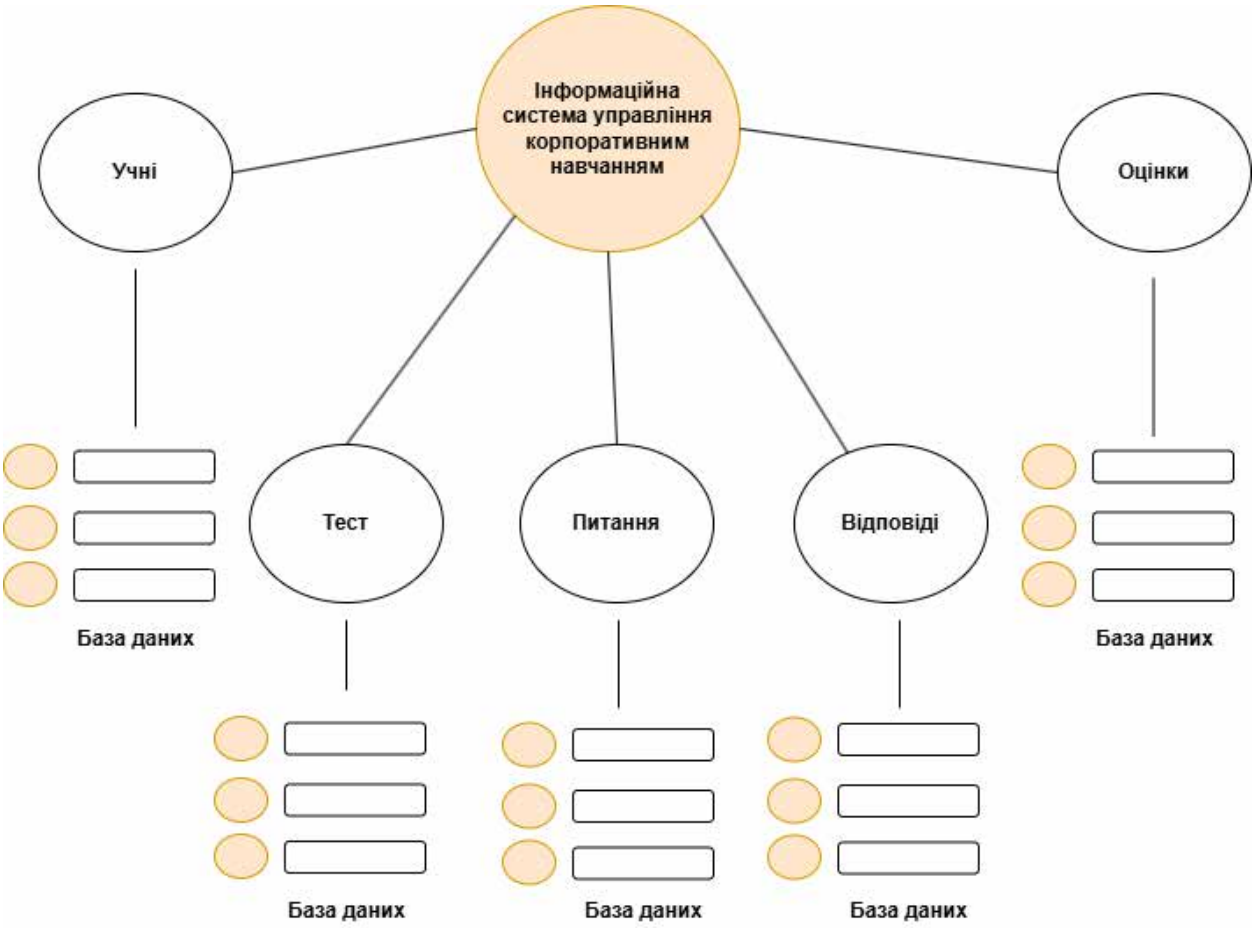

Рис. 10 Багатошарова архітектура

Ця багатошарова архітектура представляє інформаційну систему управління корпоративним навчанням, демонструючи її основні компоненти та їх взаємодію. Вона складається з декількох рівнів, кожен з яких відповідає за певний аспект роботи системи. Центральним елементом виступає інформаційна система управління навчанням, яка координує всі процеси.

Рівень учня взаємодіє з базою даних, отримуючи доступ до навчального контенту та тестових завдань. Тестовий модуль зберігає структуру тестів, забезпечуючи їх коректне проведення. Система питань містить усі запитання, пов’язані з тестами, зберігаючи їх разом із відповідями. Рівень відповідей обробляє введені користувачами варіанти, перевіряючи правильність відповідей. Оцінювання відповідає за обрахунок результатів та їх збереження.

Завдяки такій структурі система гарантує ефективне управління навчальним процесом, розділяючи функціональність на логічні компоненти.

Це дозволяє оптимізувати роботу та зберігати дані у відповідних модулях, забезпечуючи зручність для учасників навчання.

3.4 Організаційна структура програмного забезпечення

3.4.1 Діаграма пакетів. Схема пакета Інформаційної системи для управління корпоративним навчанням відображає добре структуровану модульну конструкцію, яка забезпечує масштабованість, легкість обслуговування та чіткий розподіл обов'язків. По суті, система організована в кілька взаємопов'язаних логічних компонентів, кожен з яких відповідає за певний аспект функціональності програми [15].

Пакет керування користувачами контролює все, що пов'язано з користувачами — від процесів реєстрації та входу до призначення ролей та налаштувань особистого профілю. Він забезпечує безпечний доступ та персоналізовану взаємодію для адміністраторів, тренерів та співробітників.

Обробляючи основний навчальний контент, модуль керування курсами відповідає за створення та організацію курсів, модулів та уроків. Він пов'язує навчальні матеріали з потрібними користувачами та забезпечує логічну структурованість та доступність контенту.

Для оцінки результатів навчання пакет тестування та оцінювання підтримує створення тестів, управління питаннями та відповідями, а також запис результатів користувачів. Він відіграє вирішальну роль у відстеженні прогресу та формуванні результатів на основі відповідей користувачів.

Сегмент звітності та аналітики системи надає уявлення про ефективність користувачів за допомогою комплексних звітів та візуальної аналітики. Він дозволяє адміністраторам та менеджерам відстежувати тенденції навчання, виявляти прогалини в прогресі та підтримувати прийняття рішень на основі даних.

Спеціальний рівень бази даних та персистенції забезпечує ефективне управління даними за допомогою методів ORM. Цей пакет обробляє зберігання даних, пошук та узгодженість транзакцій у всіх частинах системи.

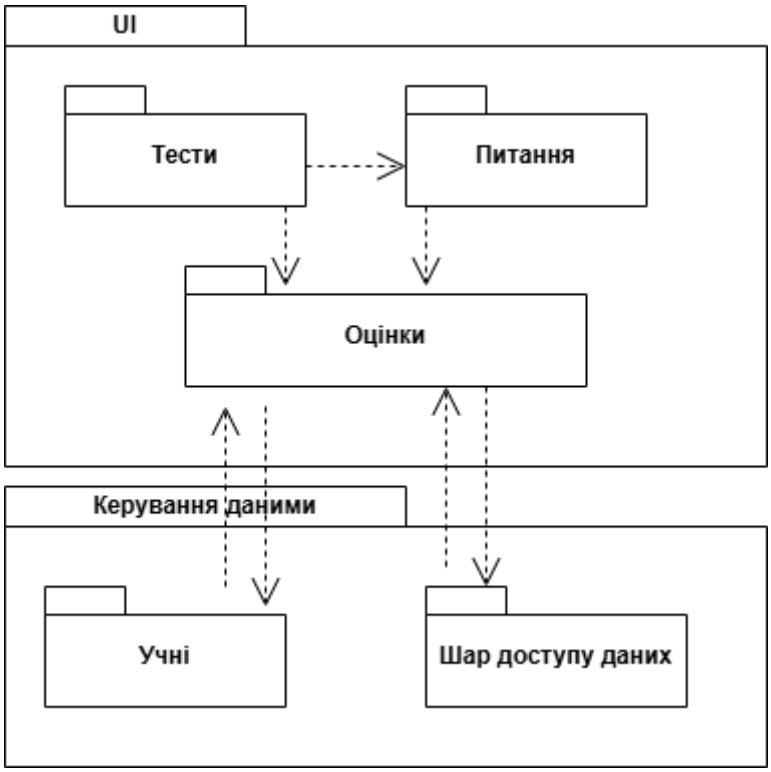

Рис. 11 Діаграма пакетів

Ця діаграма пакетів відображає структуру програмного забезпечення, розділяючи його компоненти на логічні модулі та показуючи взаємозв’язки між ними. Вона включає кілька основних пакетів, серед яких Тести, Питання, Оцінки, Учні та Шар доступу даних.

Пакет "Тести" відповідає за управління тестовими завданнями та їх запуском, а пакет "Питання" містить структуру питань, включаючи варіанти відповідей. Важливу роль виконує пакет "Оцінки", який обробляє результати тестів та визначає оцінки для учасників. "Учні" займаються зберіганням інформації про студентів, їхній прогрес та участь у тестуванні. "Шар доступу даних" забезпечує зв’язок із базою даних, передаючи необхідну інформацію для всіх інших компонентів.

Стрілки між пакетами ілюструють взаємодію: тестові завдання передаються на оцінювання, питання взаємодіють із відповідями, а дані користувачів та результати тестів синхронізуються через рівень доступу. Ця

багаторівнева структура дозволяє ефективно управляти тестуванням, забезпечуючи чіткий розподіл відповідальності між компонентами.

3.5 Вибір інструментарію для створення програмного забезпечення

Для розробки Інформаційної системи управління корпоративним навчанням було обрано набір сучасних та надійних інструментів і технологій, що забезпечують масштабованість, зручність обслуговування та швидкий розвиток. Кожен інструмент відіграє певну роль в архітектурі системи, сприяючи розробці як серверної, так і фронтендної частин, взаємодії з базами даних та контролю версій.

Бекенд програми побудовано за допомогою PHP, широко поширеної серверної мови сценаріїв, відомої своєю ефективністю в розробці динамічних веб-додатків. Вона добре підходить для створення структурованих, об'єктно-орієнтованих систем, особливо в поєднанні з сучасними фреймворками.

Для покращення організації та структури кодової бази було обрано фреймворк Symfony. Symfony пропонує модульну компонентну архітектуру, чітке розділення MVC та вбудовані інструменти для маршрутизації, шаблонизації, обробки форм, безпеки тощо. Він також бездоганно інтегрується з іншими бібліотеками PHP та заохочує використання найкращих практик та шаблонів проєктування.

Для роботи з базою даних як реляційну систему керування базами даних (RDBMS) було обрано MySQL. Це перевірене, високопродуктивне рішення з відкритим кодом, яке підтримує структуроване зберігання даних, ефективне запитування та надійну цілісність даних. MySQL має високу сумісність із Symfony завдяки Doctrine ORM, який був прийнятий як інструмент об'єктно-реляційного відображення. Doctrine спрощує взаємодію з даними, дозволяючи розробникам працювати з об'єктами PHP замість написання сирого SQL, що робить код чистішим та легшим в обслуговуванні.

Контроль версій здійснюється за допомогою Git, розподіленої системи, яка дозволяє розробникам відстежувати зміни, ефективно співпрацювати та керувати версіями проєктів на всіх етапах розробки. Це забезпечує стабільність коду та забезпечує надійне резервне копіювання та історію процесу розробки.

На фронтенді обрані технології включають HTML, CSS та JavaScript, які формують основу веб-інтерфейсів. HTML структурує контент, CSS стилізує візуальний вигляд, а JavaScript забезпечує інтерактивність та динамічну поведінку на стороні клієнта. Ці інструменти працюють разом, щоб забезпечити адаптивний та зручний інтерфейс для всіх користувачів системи, від адміністраторів до стажерів.

Цей набір технологій був обраний на основі таких критеріїв, як системні вимоги, досвід команди, доступність документації, підтримка спільноти та довгострокова стійкість проєкту. Разом вони пропонують збалансоване середовище для побудови надійної та безпечної веб-системи управління навчанням.

4 ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЯ СИСТЕМИ

4.1 Вимоги до апаратного та програмного забезпечення

Для забезпечення надійної роботи, сумісності та простоти підтримки під час розробки та розгортання Інформаційної системи для управління корпоративним навчанням, як апаратне, так і програмне забезпечення повинні відповідати певним технічним критеріям. Ці вимоги дещо відрізняються залежно від мети – чи використовується середовище для розробки, тестування чи повномасштабного виробництва.

На стороні сервера сучасна високопродуктивна конфігурація є важливою для підтримки логіки додатків, взаємодії з користувачами та зберігання даних. Для ефективної обробки одночасних процесів рекомендується процесор щонайменше з чотирма ядрами, такий як Intel i5 або AMD Ryzen 5. Об'єм пам'яті повинен бути не менше 8 ГБ, хоча 16 ГБ є кращим для систем, які очікують більшого трафіку користувачів або більших наборів даних. Твердотільний накопичувач (SSD) з мінімум 100 ГБ доступного простору допоможе забезпечити швидкий доступ до даних та продуктивність додатків. Стабільне підключення до Інтернету з достатньою пропускною здатністю є життєво важливим, особливо в середовищах з багатьма активними користувачами, а для захисту від втрати даних необхідне рішення для резервного копіювання – зовнішнє або хмарне.

Клієнтські пристрої, включаючи ті, що використовуються стажерами, інструкторами або адміністраторами, повинні мати можливість запускати сучасні веб-браузери та безперебійно обробляти динамічний веб-контент. Для щоденної взаємодії з системою достатньо двоядерного процесора з щонайменше 4 ГБ оперативної пам'яті, роздільної здатності екрана 1366x768 або вище та надійного підключення до Інтернету.

З точки зору програмного забезпечення, серверне середовище має базуватися на надійній операційній системі, такій як Ubuntu Server 20.04 LTS або сумісному дистрибутиві Linux. Як альтернатива, можна використовувати Windows Server, залежно від уподобань організації. Для обслуговування програми необхідно встановити відповідний веб-сервер, такий як Apache 2.4 або Nginx. Система працюватиме на PHP версії 8.1 або пізнішої, налаштований з усіма необхідними розширеннями, такими як PDO, mbstring та intl, для забезпечення сумісності з фреймворком та інструментами ORM. MySQL 8.0 обрано як систему керування базами даних завдяки її надійності, масштабованості та широкому розповсюдженню. Symfony, обраний PHP-фреймворк, буде використовуватися разом із Composer для керування залежностями та Doctrine ORM для оптимізації взаємодії з базою даних.

Розробники отримають вигоду від використання інтегрованих середовищ розробки (IDE), таких як PhpStorm або Visual Studio Code, які надають інструменти для підвищення продуктивності та підтримку синтаксису. Додаткові інструменти, такі як Git для контролю версій, MySQL Workbench або DBeaver для операцій з базами даних та Postman для тестування API, стануть частиною основного набору інструментів розробки. Для тестування та налагодження ідеально підходять браузери, такі як Chrome або Firefox, завдяки своїм потужним інструментам розробника.

Це збалансоване поєднання апаратних можливостей та ретельно підібраного програмного стеку забезпечує стабільну основу для побудови безпечної, масштабованої та ефективної системи управління корпоративним навчанням.

Також було зроблено діаграму розгортання для візуального огляду деплою системи (Рис. 12).

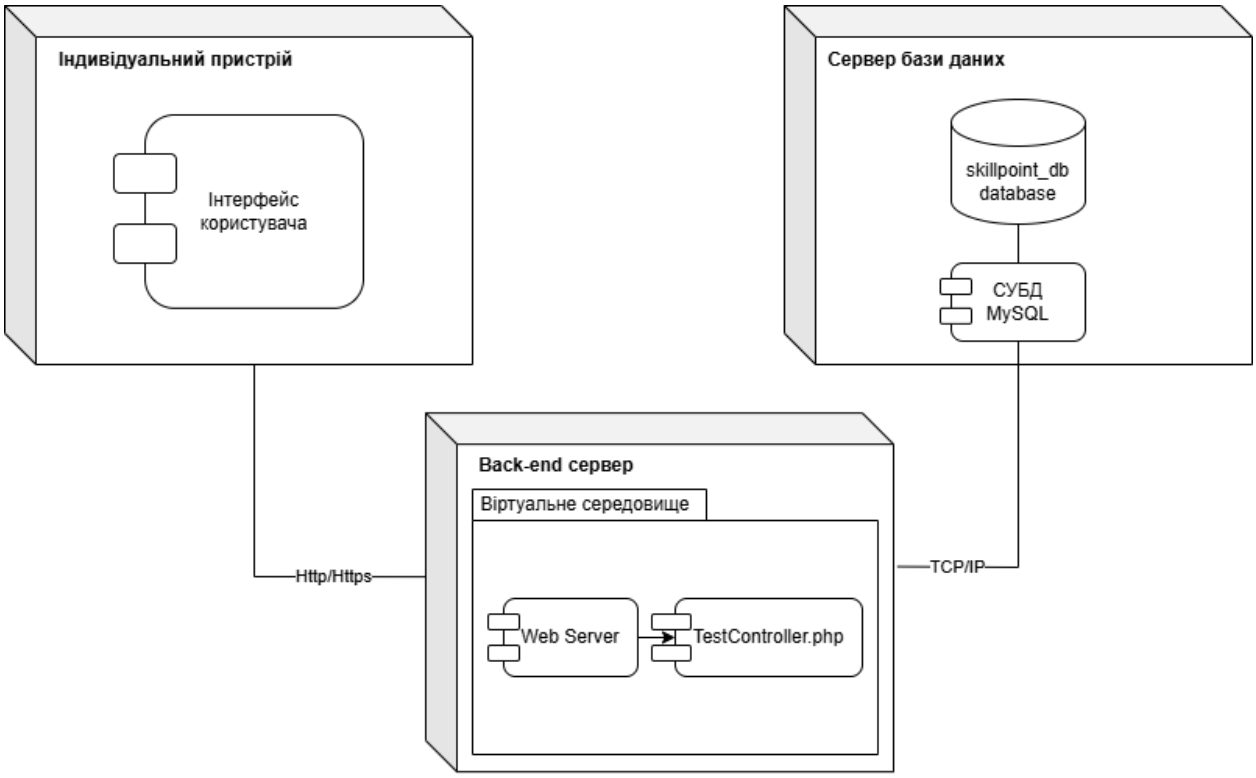

Рис. 12 Діаграма розгортання

4.2 Тестування системи

Запустивши систему, потрапляємо на форму авторизації, тут нам треба ввести логін та пароль користувача (рис. 13).

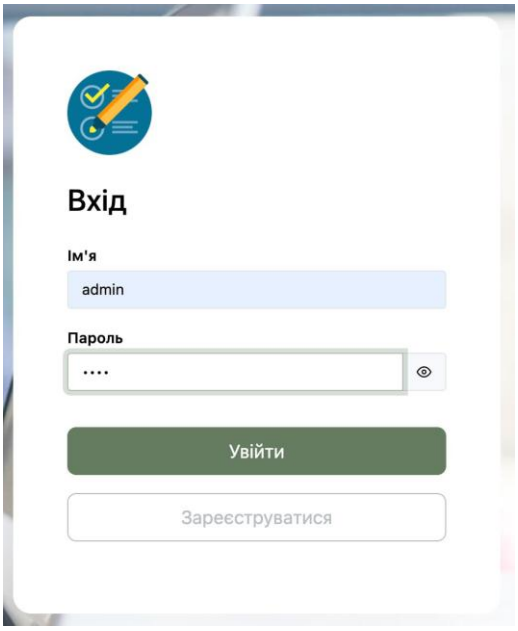

Рис. 13 Форма авторизації

Після авторизації потрапляємо на головну сторінку, де у нас виведена якась інформаційна панель у вигляді карток з корисними відомостями: скільки активних тестів, який найближчий тест, який середній бал тощо (Рис. 14).

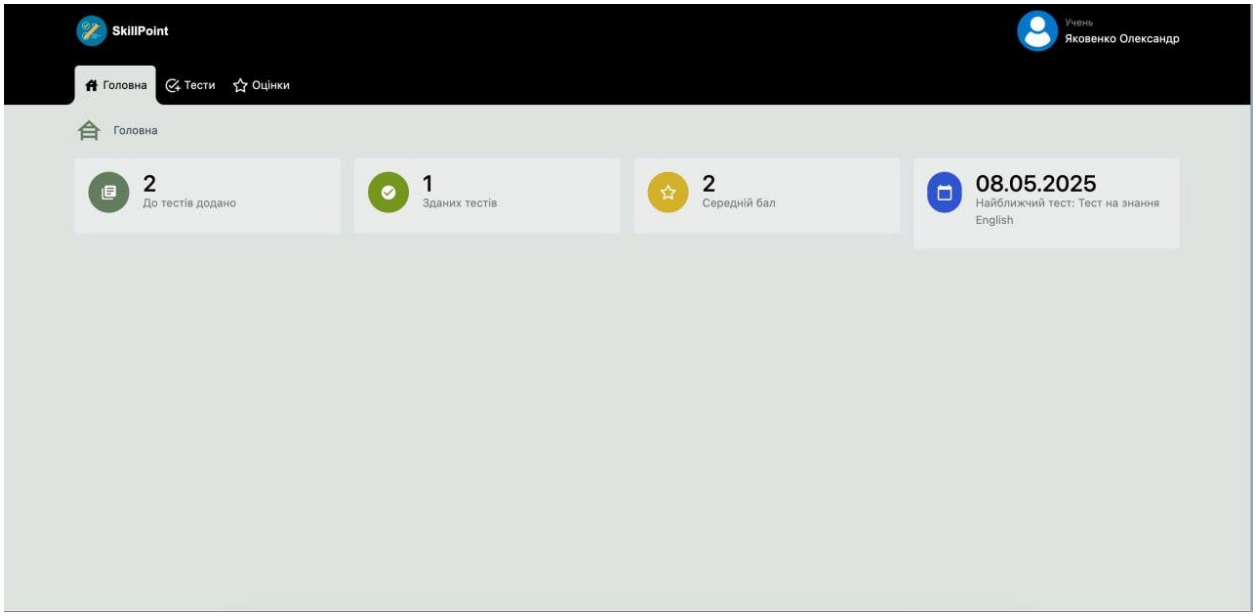

Рис. 14 Головна сторінка

Перейдемо на сторінку з тестами і побачимо всі тести, які ми додали або в які ми додані іншими користувачами. Тут виводиться назва тесту, його короткий опис та кнопка відкриття тесту (Рис. 15).

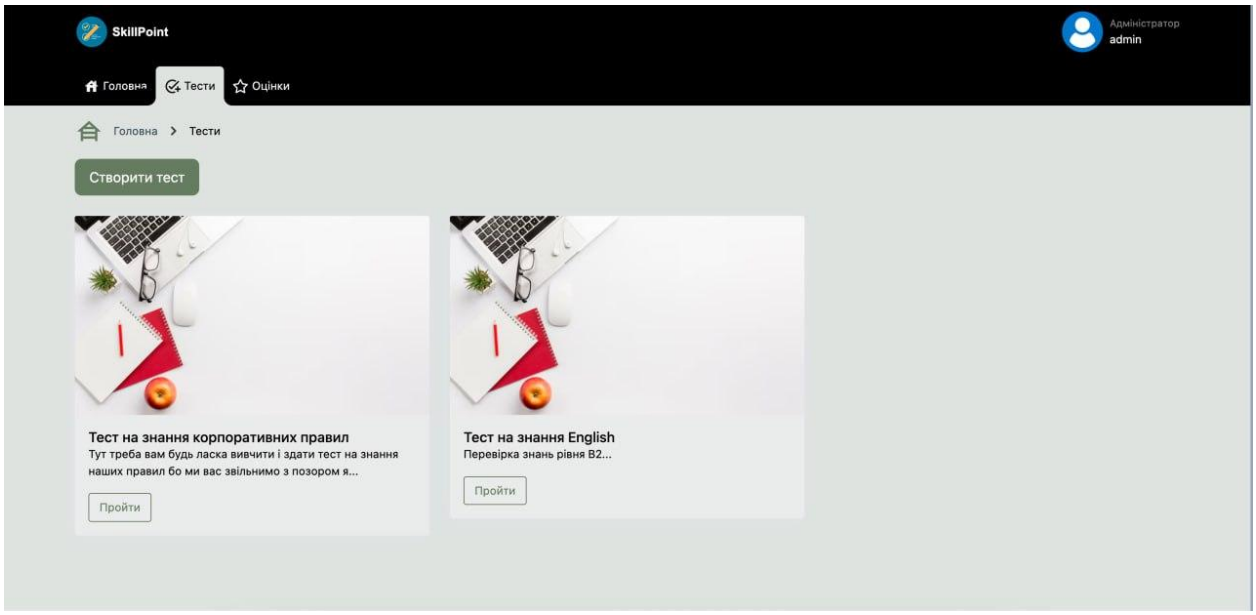

Рис. 15 Сторінка “Тести”

Натисніть на кнопку створення тесту і у нас відкриється форма з полями, які потрібно заповнити, щоб створити тест (Рис. 16).

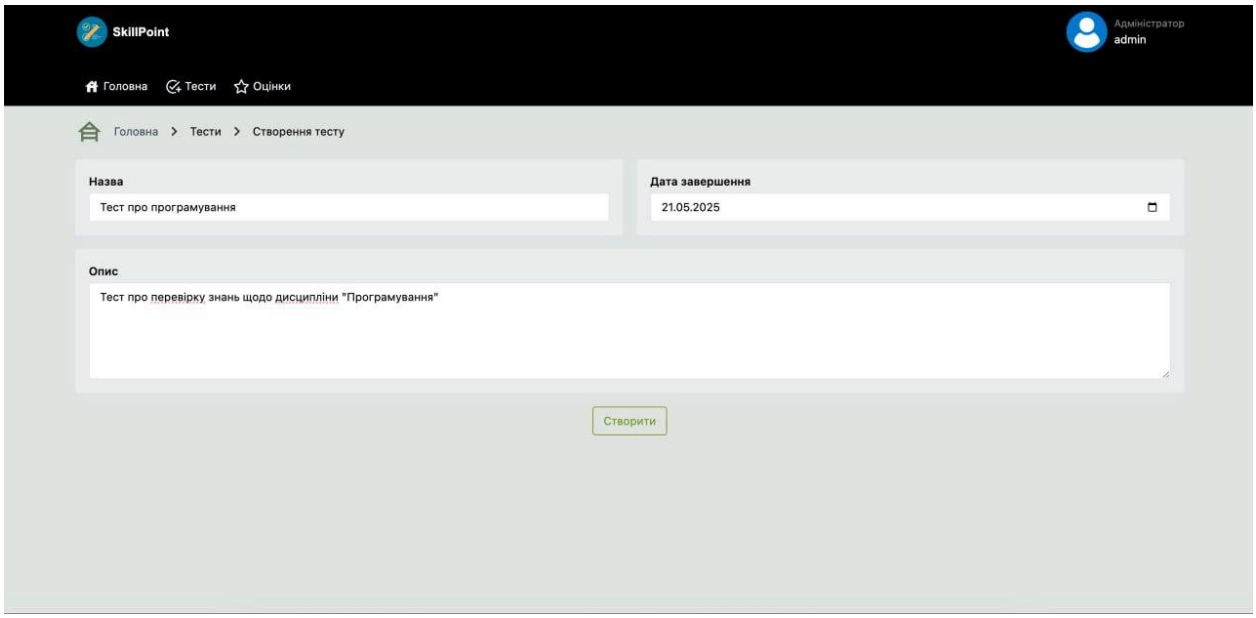

Рис. 16 Форма створення тесту

Після підтвердження, нас переносить на цей тест, де ми бачимо його інформацію та кнопки редагування тесту або початку проходження тесту (Рис. 17).

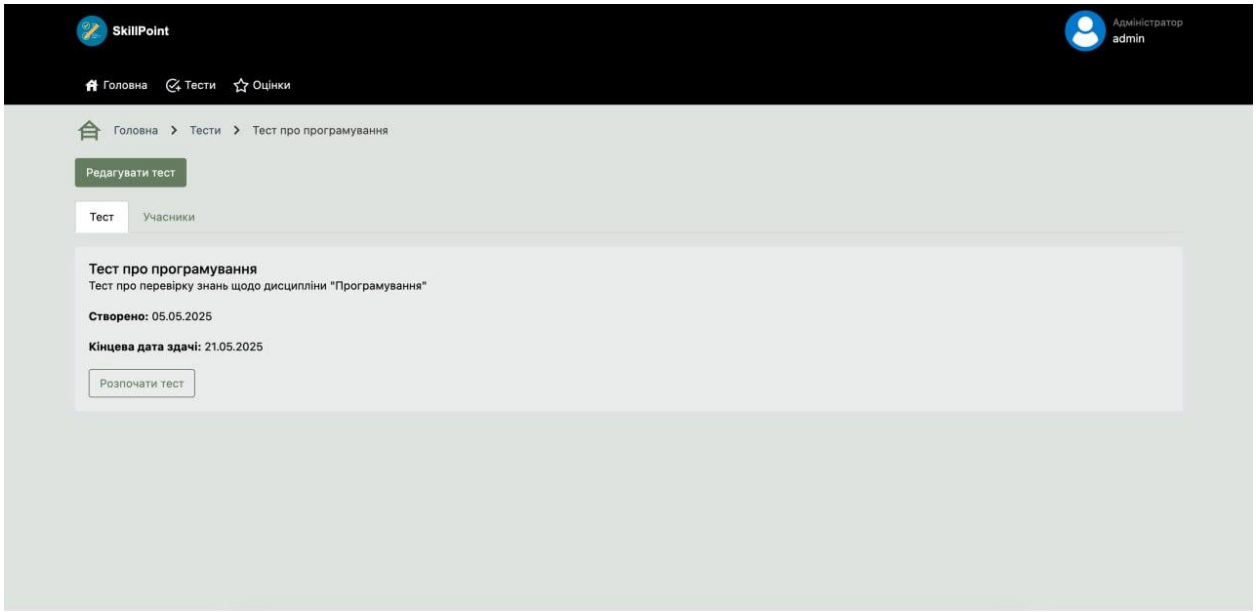

Рис. 17 Сторінка обраного тесту

Відкриємо вкладку з учасниками та побачимо список доданих учасників на цей тест, додавати учасників може лише користувач за участю адміністратора (Рис. 18).

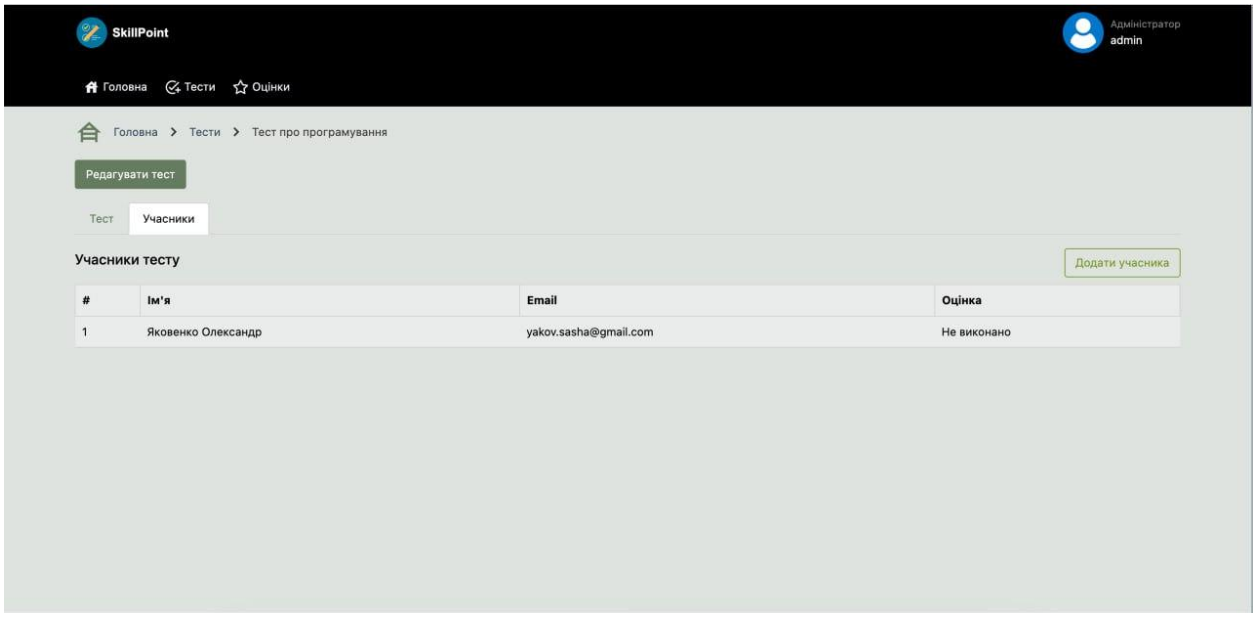

Рис. 18 Сторінка обраного тесту – Вкладка “Учасники”

Спробуємо додати нового учасника і у нас відкриється модальне вікно зі списком усіх учасників у базі даних та полем для пошуку (Рис. 19).

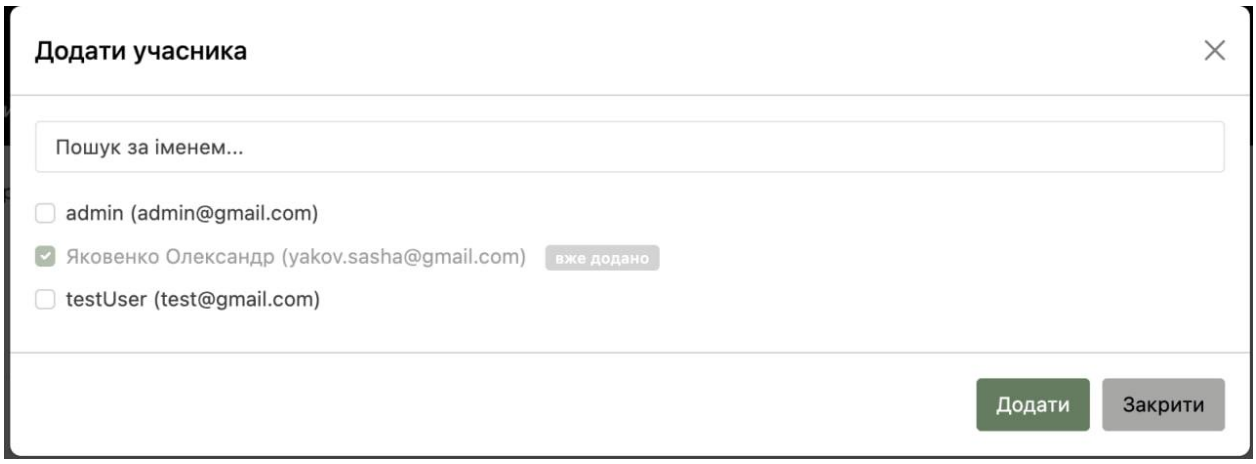

Рис. 19 Модальна форма додавання учасника до тесту

Тепер почнемо проходження тесту. Натиснувши кнопку, у нас почнеться відлік часу і перед нами буде список усіх питань тесту, де нам потрібно вибрати правильну відповідь (Рис. 20).

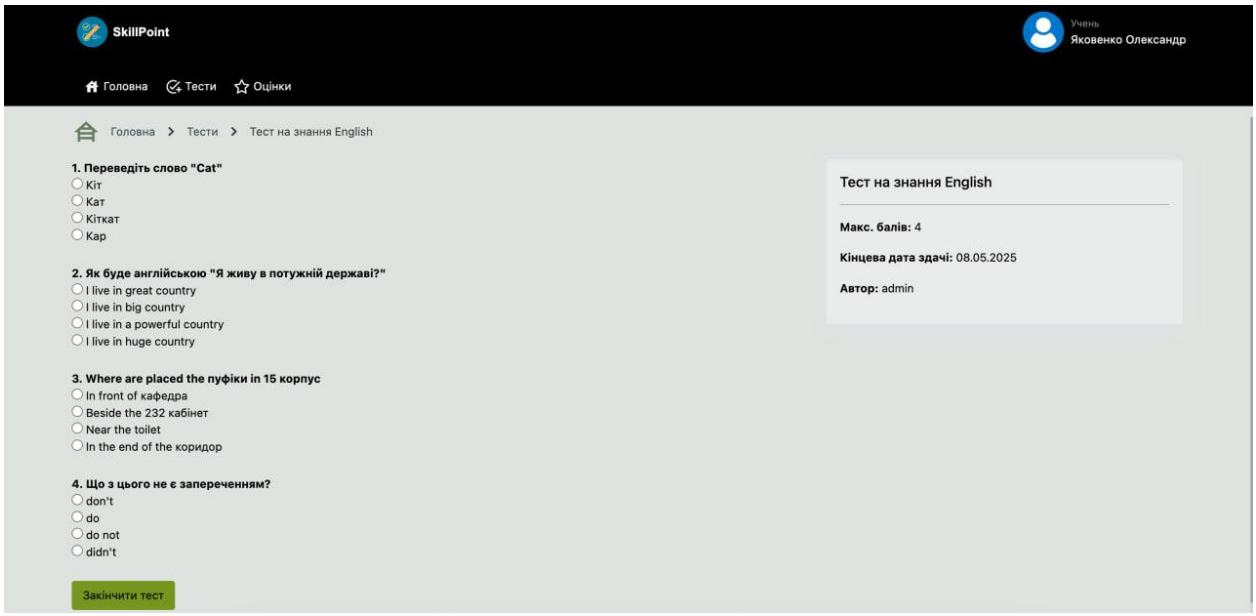

Рис. 20 Проходження тесту

Після завершення тесту нам відкриється сторінка з результатами тесту де буде показаний наш бал і час проходження (Рис. 21).

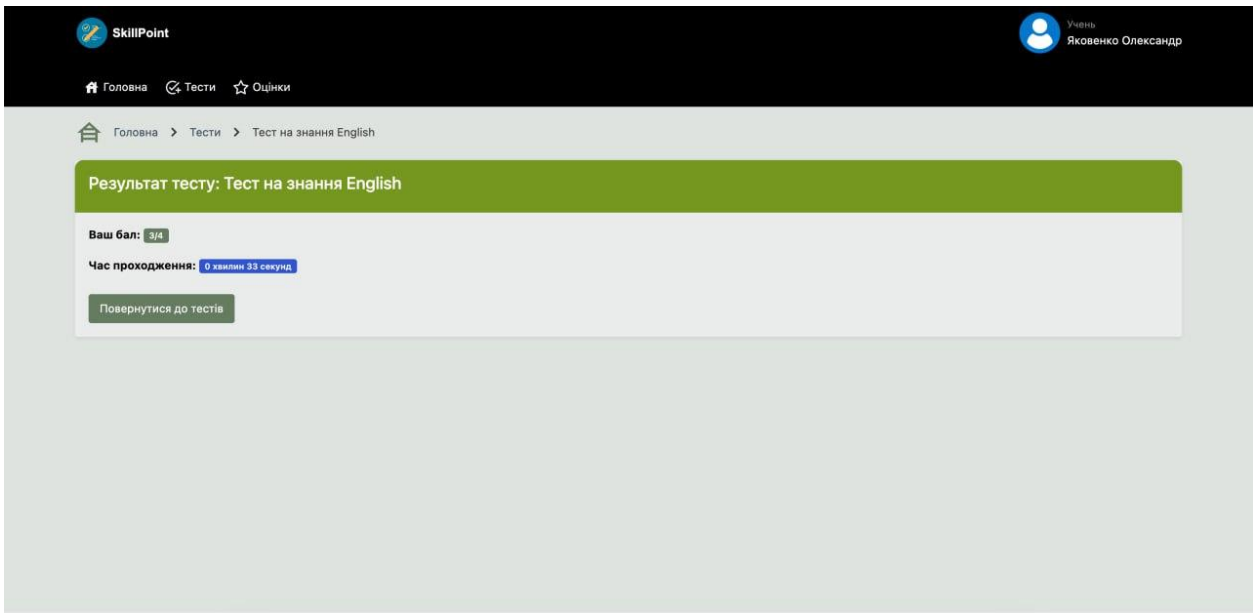

Рис. 21 Завершення тесту

Тепер перейдемо на сторінку з оцінками і тут ми можемо подивитися в таблиці список усіх наших тестів, де ми додані, можемо дізнатися, що ми пройшли і на яку оцінку (Рис. 22).

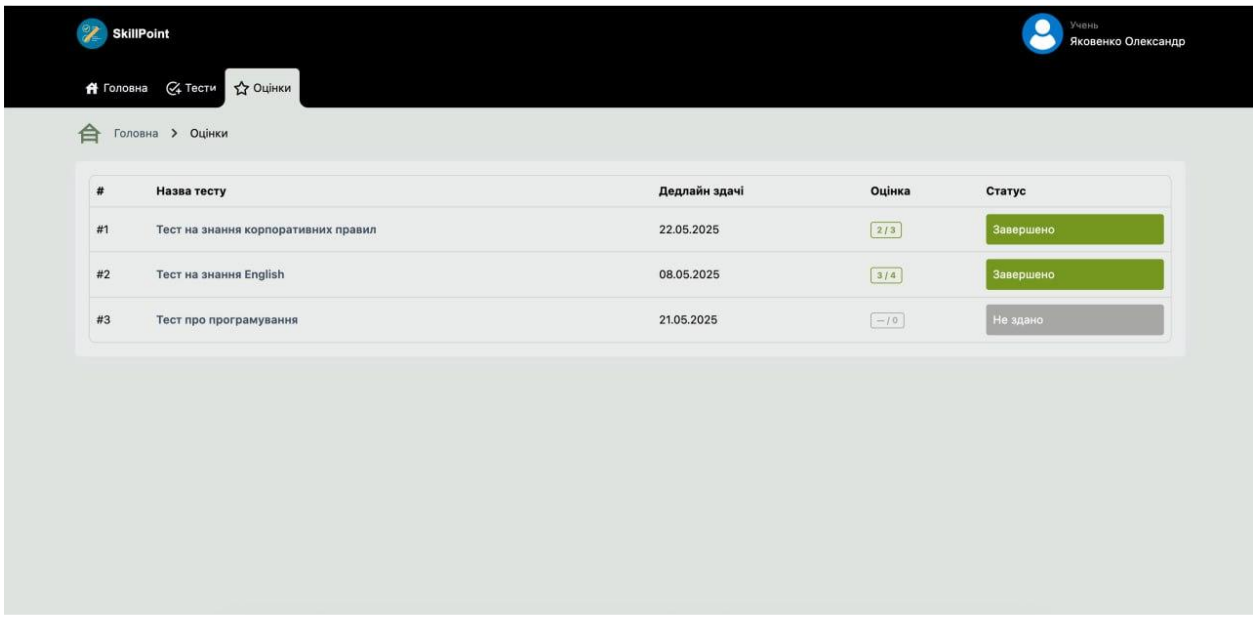

Рис. 22 Сторінка “Оцінки”

Якщо ми повернемося на вже пройдений тест, ми не зможемо його пройти ще раз, користувачеві в такому разі буде показуватись повідомлення, що цей тест у нього вже пройдено (Рис. 23).

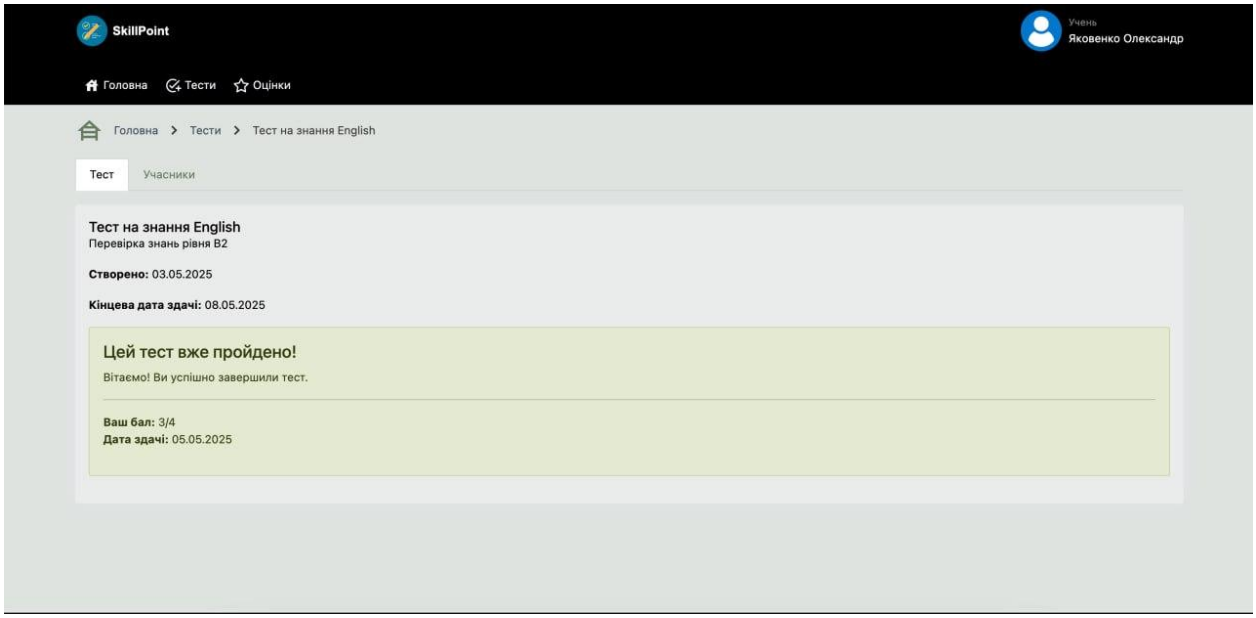

Рис. 23 Сторінка з пройденим тестом

Також продемонструємо форму створення питань до тесту, тут ми вводимо в поля текст запитання та текст для варіантів відповіді, вибираючи через радіокнопку правильну відповідь (Рис. 24).

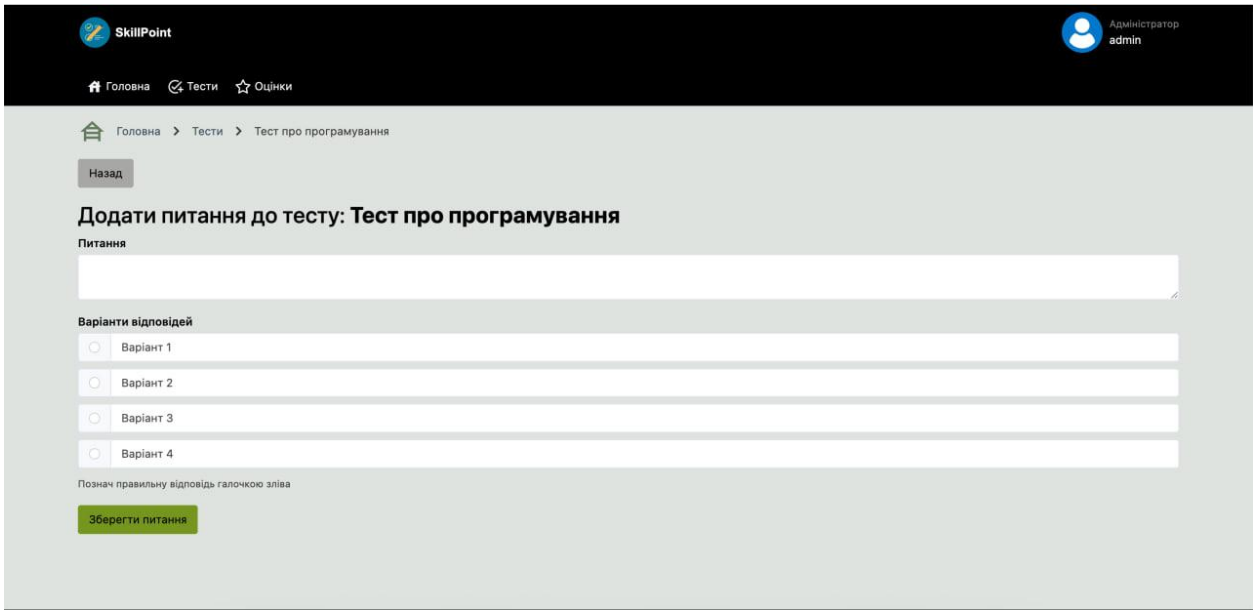

Рис. 24 Форма додання питань до тесту

ВИСНОВКИ

На завершення, розробка інформаційної системи для управління корпоративним навчанням є значним кроком до оптимізації процесів навчання в організації. Завдяки інтеграції різних функцій, система дозволяє ефективно керувати навчальними програмами, відстежувати прогрес співробітників та створювати змістовні звіти. Ця автоматизація не тільки спрощує адміністративні завдання, але й покращує загальний досвід навчання співробітників, гарантуючи, що організація може ефективно досягати своїх цілей у сфері освіти та професійного розвитку.

Вибір таких технологій, як PHP, Symfony, MySQL та Doctrine ORM, для розробки цієї системи забезпечує масштабованість, продуктивність та безпеку. Ці інструменти широко використовуються в галузі, забезпечуючи надійні рішення для створення безпечних, гнучких та зручних у підтримці додатків. Крім того, використання реляційної системи управління базами даних, такої як MySQL, підтримує цілісність даних та дозволяє виконувати складні запити, необхідні для відстеження прогресу співробітників та управління навчальними матеріалами.

Завдяки цьому проєкту стало зрозуміло, що добре розроблена система управління корпоративним навчанням може сприяти підвищенню кваліфікації співробітників, підвищенню продуктивності та більшій задоволеності. Крім того, вона забезпечує комплексне уявлення про ландшафт навчання в компанії, допомагаючи менеджерам приймати обґрунтовані рішення щодо майбутніх інвестицій у навчання.

Крім того, система була розроблена з урахуванням масштабованості, що гарантує її можливість адаптуватися до майбутнього зростання та збільшення використання. У міру розширення організації та збільшення кількості співробітників система продовжуватиме функціонувати ефективно, зберігаючи свою продуктивність та надійність. Модульна архітектура

дозволяє легко додавати нові функції або інтегрувати сторонні інструменти, що дозволяє системі розвиватися разом зі зміною вимог організації. Крім того, зручний інтерфейс гарантує, що як адміністратори, так і співробітники можуть безперешкодно взаємодіяти з системою, роблячи її доступною та інтуїтивно зрозумілою для всіх користувачів, незалежно від їхньої технічної експертизи.

Впровадження цієї інформаційної системи також відповідає ширшим тенденціям цифрової трансформації в бізнесі. Застосовуючи технологічне рішення для управління навчанням, організація може забезпечити більш інформаційно-орієнтований підхід до розвитку співробітників. Цей зсув не тільки оптимізує ресурси, але й сприяє культурі безперервного навчання та вдосконалення. Завдяки впровадженню системи компанія може краще узгоджувати навчальні програми з бізнес-цілями, вимірювати ефективність навчання та, зрештою, сприяти формуванню більш кваліфікованої та компетентної робочої сили. Оскільки корпоративне навчання продовжує розвиватися, ця система є фундаментальним інструментом для підтримки розвитку співробітників та сприяння успіху організації.

Загалом, ця система має потенціал значно підвищити ефективність корпоративного навчання, зменшити адміністративні накладні витрати та надати співробітникам інструменти, необхідні для успіху в їхньому професійному розвитку. Подальші вдосконалення можуть включати інтеграцію додаткових функцій, таких як автоматизовані навчальні шляхи, розширені можливості звітності та інтеграція із зовнішніми платформами, що дозволить системі зростати та адаптуватися відповідно до розвитку потреб організації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kapoor, J.R., Dlabay, L.R., Hughes, R.J. Фінансові спеціалісти. — McGraw-Hill Education, 2020.
2. Lahti, A., Wehling, E. Електронне навчання в корпоративному середовищі: переваги та виклики. — Springer, 2018.
3. Saini, S., Sharma, A. Корпоративне навчання та розвиток: стратегії та інструменти. — Pearson Education, 2019.
4. Moodle – [Електронний ресурс] – Режим доступу: https://moodle.org/
5. TalentLMS – [Електронний ресурс] – Режим доступу: https://help.talentlms.com/hc/en-us/articles/13389256517148-Exploring-the-new-TalentLMS-interface
6. Armstrong, M. Довідник з практики управління людськими ресурсами. — Kogan Page, 2020.
7. Noe, R.A. Навчання та розвиток співробітників. — McGraw-Hill Education, 2017.
8. DeVaro, J. Економіка навчання: дані корпоративного сектору. — Routledge, 2016.
9. Buchanan, D.A., Huczynski, A.A. Організаційна поведінка. — Pearson Education, 2019.
10. Doctrine ORM – [Електронний ресурс] – Режим доступу: https://www.doctrine-project.org/projects/orm.html
11. Документація Symfony – [Електронний ресурс] – Режим доступу: https://symfony.com/doc/current/index.html
12. Документація MySQL – [Електронний ресурс] – Режим доступу: https://dev.mysql.com/doc/
13. Посібник з PHP – [Електронний ресурс] – Режим доступу: https://www.php.net/manual/en/

14. Документація Git – [Електронний ресурс] – Режим доступу: https://git-scm.com/doc
15. Посібник з HTML W3Schools – [Електронний ресурс] – Режим доступу: https://www.w3schools.com/html/
16. Посібник з CSS W3Schools – [Електронний ресурс] – Режим доступу: https://www.w3schools.com/css/
17. Посібник з JavaScript W3Schools – [Електронний ресурс] – Режим доступу: https://www.w3schools.com/js/
18. "«Важливість систем управління корпоративним навчанням» – [Електронний ресурс] – Режим доступу: https://elearningindustry.com/importance-corporate-learning-management-system
19. «Найкращі практики управління корпоративним навчанням» – [Електронний ресурс] – Режим доступу: https://www.simplilearn.com/corporate-training-best-practices-article
20. Інструмент для створення діаграм Draw.io – [Електронний ресурс] – Режим доступу: https://app.diagrams.net/
21. ПРОЄКТУВАННЯ ER-ДІАГРАМИ – [Електронний ресурс] – Режим доступу: https://nationalteam.worldskills.ua/skills/proektirovanie-er-diagrammy/
22. Діаграма варіантів використання (UseCase diagram) – [Електронний ресурс] – Режим доступу: https://flexberry.github.io/ua/fd_use-case-diagram.html
23. ПРОЄКТУВАННЯ USE CASE ДІАГРАМИ. ВИЗНАЧЕННЯ ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ СИСТЕМИ – [Електронний ресурс] – Режим доступу: https://nationalteam.worldskills.ua/skills/proektirovanie-use-case-diagrammy-opredelenie-funktsionalnykh-vozmozhnostey-sistemy/

24. What is Sequence Diagram? – [Електронний ресурс] – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/
25. UML - Activity Diagrams – Tutorialspoint – [Електронний ресурс] – Режим доступу: https://www.tutorialspoint.com/uml/uml_activity_diagram.htm
26. Побудова діаграми класів – [Електронний ресурс] – Режим доступу: https://flexberry.github.io/ua/gpg_class-diagram.html
27. UML-діаграми класів – [Електронний ресурс] – Режим доступу: https://prog-cpp.ua/uml-classes/
28. Entity Relationship Diagram - Data Modeling – [Електронний ресурс] – Режим доступу: https://www.visualparadigm.com/VPGallery/datamodeling/EntityRelationshipDiagram.html
29. UML 2 Tutorial - Package Diagram - Sparx Systems – [Електронний ресурс] – Режим доступу: https://sparxsystems.com/resources/tutorials/uml2/package-diagram.html
30. Документація Bootstrap (офіційний сайт) – [Електронний ресурс] – Режим доступу: www.getbootstrap.com/documentation.

ДОДАТОК А

Фрагменти програмного коду. Функція створення тесту

```
#[Route('/test/create', name: 'app_test_create')]
    #[IsGranted('ROLE_ADMIN')]
    public function createTest(Request $request, EntityManagerInterface
$entityManager): Response
    {
        if ($request->isMethod('POST')) {
            $title = $request->request->get('title');
            $description = $request->request->get('description');
            $dueDate = $request->request->get('due_date');

            $test = new Test();
            $test->setTitle($title);
            $test->setDescription($description);
            $test->setCreatedAt(new \DateTimeImmutable());
            $test->setUser($this->getUser());

            if (!empty($dueDate)) {
                $test->setDueDate(new \DateTime($dueDate));
            }

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

            flash()->success('Тест успішно створено');

            return $this->redirectToRoute('app_test');
        }

        return $this->render('test/create.html.twig');
    }

    #[Route('/test/{id}', name: 'app_test_show')]
    public function show(int $id, EntityManagerInterface $entityManager):
Response
    {
        $test = $entityManager->getRepository(Test::class)->find($id);

        if (!$test) {
            throw $this->createNotFoundException('Тест не знайдено');
        }
```

```
        $participants = $entityManager->getRepository(TestUser::class)-
>findBy(['test' => $test]);
        $participantIds = array_map(fn($p) => $p->getUser()->getId(), $participants);

        $allUsers = $entityManager->getRepository(User::class)->findAll();

        $user = $this->getUser();
        $result = $entityManager->getRepository(Result::class)->findOneBy([
            'test' => $test,
            'user' => $user,
        ]);

        $scoreResults = $entityManager->getRepository(Result::class)->findBy(['test'
=> $test]);

        $participantScores = [];
        foreach ($scoreResults as $scoreResult) {
            $participantScores[$scoreResult->getUser()->getId()] = [
                'score' => $scoreResult->getScore(),
                'completionDate' => $scoreResult->getCompletedAt(),
            ];
        }

        $isTestPassed = false;
        $score = null;
        $completionDate = null;

        if ($result && $result->getStatus() === Result::STATUS_FINISHED) {
            $isTestPassed = true;
            $score = $result->getScore();
            $completionDate = $result->getCompletedAt();
        }

        $maxScore = count($test->getQuestions());

        return $this->render('test/show.html.twig', [
            'test' => $test,
            'participants' => $participants,
            'participantIds' => $participantIds,
            'allUsers' => $allUsers,
            'isTestPassed' => $isTestPassed,
            'score' => $score,
            'completionDate' => $completionDate,
            'maxScore' => $maxScore,
```

```
            'participantScores' => $participantScores,
        ]);
    }

#[Route('/test/{id}/add-participant', name: 'app_test_add_participant', methods:
['POST'])]
    #[IsGranted('ROLE_ADMIN')]
    public function addParticipant(int $id, Request $request,
EntityManagerInterface $entityManager): Response
    {
        $test = $entityManager->getRepository(Test::class)->find($id);

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

        $userIds = $request->request->all('users');
        foreach ($userIds as $userId) {
            $user = $entityManager->getRepository(User::class)->find($userId);

            if ($user) {
                $existing = $entityManager->getRepository(TestUser::class)-
>findOneBy(['user' => $user, 'test' => $test]);

                if (!$existing) {
                    $testUser = new TestUser();
                    $testUser->setUser($user);
                    $testUser->setTest($test);
                    $entityManager->persist($testUser);
                }
            }
        }

        $entityManager->flush();
        flash()->success('Учасник успішно доданий');

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

ДОДАТОК Б

Фрагменти програмного коду. Функціонал проходження тесту

```
#[Route('/test/{id}/start', name: 'app_test_start')]
    public function startTest(int $id, EntityManagerInterface $em): Response
    {
        $test = $em->getRepository(Test::class)->find($id);

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

        $questions = $em->getRepository(Question::class)->findBy(['test' => $test]);

        $maxScore = count($test->getQuestions());

        return $this->render('test/start.html.twig', [
            'test' => $test,
            'questions' => $questions,
            'startTime' => time(),
            'maxScore' => $maxScore,
        ]);
    }

    #[Route('/test/{id}/finish', name: 'app_test_finish', methods: ['POST'])]
    public function finishTest(Request $request, int $id, EntityManagerInterface
$em): Response
    {
        $test = $em->getRepository(Test::class)->find($id);
        $user = $this->getUser();

        if (!$test || !$user) {
            throw $this->createNotFoundException('Тест не знайдено або
користувач не авторизований');
        }

        $userAnswers = $request->request->all('answers');
        $startTime = (int)$request->request->get('start_time');
        $endTime = time();
        $duration = $endTime - $startTime;

        $score = 0;
        foreach ($userAnswers as $questionId => $answerId) {
            $answer = $em->getRepository(Answer::class)->find($answerId);
```

```
            if ($answer && $answer->isCorrect()) {
                $score++;
            }
        }

        $result = new Result();
        $result->setTest($test);
        $result->setUser($user);
        $result->setScore($score);
        $result->setStatus(Result::STATUS_FINISHED);
        $result->setCompletedAt(new \DateTimeImmutable());

        $em->persist($result);
        $em->flush();

        return $this->redirectToRoute('app_test_result', [
            'id' => $result->getId(),
            'duration' => $duration,
        ]);
    }

    #[Route('/test/result/{id}', name: 'app_test_result')]
    public function testResult(int $id, Request $request, EntityManagerInterface
$em): Response
    {
        $result = $em->getRepository(Result::class)->find($id);

        if (!$result || $result->getUser() !== $this->getUser()) {
            throw $this->createNotFoundException('Результат не знайдено або
немає доступу');
        }

        $duration = $request->query->get('duration');
        $test = $result->getTest();
        $maxScore = count($result->getTest()->getQuestions());

        return $this->render('test/result.html.twig', [
            'result' => $result,
            'duration' => $duration,
            'test' => $test,
            'maxScore' => $maxScore,
        ]);
    }
```