ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ТЕХНОЛОГІЙ ЦИФРОВОГО РОЗВИТКУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Інформаційна система управління навчальними курсами з онлайн-тестуванням, контролем дедлайнів та інтерактивною аналітикою прогресу на основі PHP, HTML, CSS та JavaScript»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Технології цифрового розвитку»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Максим ЄФІМОВ
(підпис)

Виконав: здобувач вищої освіти групи ТЦР-44

Максим ЄФІМОВ

Керівник: Андрій АРОНОВ
кандидат технічних наук

Рецензент: _____

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Технологій цифрового розвитку

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Технології цифрового розвитку»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Технологій цифрового розвитку
_____ Вікторія ЖЕБКА
« ____ » _____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Єфімову Максиму Олександровичу

1. Тема кваліфікаційної роботи: «Інформаційна система управління навчальними курсами з онлайн-тестуванням, контролем дедлайнів та інтерактивною аналітикою прогресу на основі PHP, HTML, CSS та JavaScript»
керівник кваліфікаційної роботи кандидат технічних наук Андрій АРОНОВ,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «_20_» лютого 2026 року № _51__.

2. Строк подання кваліфікаційної роботи «_18_» травня 2026 року.

3. Вихідні дані до кваліфікаційної роботи: документація PHP, SQL Server, HTML, CSS, JavaScript; матеріали проєкту LMS; наукові джерела з онлайн- тестування, дедлайнів та аналітики навчального процесу; вимоги до функціональності систем дистанційного навчання.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз предметної галузі та огляд існуючих LMS-систем.
2. Аналіз технологій реалізації веб-орієнтованих інформаційних систем.
3. Проєктування архітектури системи та структури бази даних.

4. Програмна реалізація модулів: управління курсами, онлайн-тестування, контроль дедлайнів, аналітика прогресу.
5. Тестування застосунку.

5. Перелік графічного матеріалу: *презентація*

- Аналіз аналогів.
- Вимоги до програмного забезпечення.
- Програмні засоби реалізації.
- Діаграма варіантів використання.
- Діаграма компонентів
- Схема структури бази даних
- Діаграма класів
- Екранні форми інтерфейсу

6. Дата видачі завдання «_20_» лютого 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	20.02.2026-24.02.2026	Виконано
2	Аналіз та дослідження існуючих аналогів	25.02.2026-04.03.2026	Виконано
3	Огляд існуючих алгоритмів обробки тексту та методів і технологій побудови тематичного словника	05.03.2026-10.03.2026	Виконано
4	Проектування застосунку для автоматизованої побудови тематичного словника навчальної дисципліни	11.03.2026-18.03.2026	Виконано
5	Програмна реалізація застосунку	19.03.2026-05.05.2026	Виконано
6	Тестування застосунку	19.03.2026-05.05.2026	Виконано
7	Оформлення роботи: вступ, висновки, реферат	06.05.2026-10.05.2026	Виконано
8	Розробка демонстраційних матеріалів	08.05.2026-10.05.2026	Виконано
9	Попередній захист роботи	11.05.2026	Виконано

Здобувач вищої освіти _____ Максим ЄФІМОВ
(підпис)

Керівник
кваліфікаційної роботи _____ Андрій АРОНОВ
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: __52__ стор., __1___ табл., _32___ рис., __16__ джерел.

Мета роботи – удосконалення процесу навчання шляхом створення веб-системи з підтримкою управління навчальними курсами, онлайн-тестування, контролю дедлайнів та роботи з файлами.

Об'єкт дослідження – організація навчання у веб-орієнтованих інформаційних системах.

Предмет дослідження – підходи, моделі та програмні рішення для розробки LMS із підтримкою тестування, дедлайнів, завдань і аналітики результатів.

Короткий зміст роботи: описано архітектуру веб-системи, структуру бази даних, модулі для керування курсами, завданнями, онлайн-тестами з таймером, повторною здачею файлів до дедлайну і переглядом результатів. Було програмно ключові компоненти: авторизація з ролями, управління завданнями, тестування, збереження результатів, а також окремий аналітичний модуль зі статистикою успішності студентів. В роботі використано технології PHP, HTML, CSS, JavaScript, Microsoft SQL Server; застосовані методи системного аналізу, проєктування баз даних і тестування ПЗ. Проведено функціональне тестування та перевірено працездатність системи.

Сферою використання застосунку є організація дистанційного та змішаного навчання в навчальних закладах, яким потрібні інструменти для організації курсів, контролю дедлайнів, проведення тестів і моніторингу результатів студентів.

КЛЮЧОВІ СЛОВА: LMS, ОНЛАЙН-ТЕСТУВАННЯ, ДЕДЛАЙНИ, АНАЛІТИКА ПРОГРЕСУ, PHP, SQL SERVER, ВЕБ-СИСТЕМА, НАВЧАЛЬНІ КУРСИ.

ЗМІСТ

ВСТУП

Актуальність: сьогодні, коли освіта все більше переходить у "цифру", дуже бракує веб-систем, які реально допомагають організовувати навчання – вести курси, швидко діставати матеріали, проводити онлайн-тести, слідкувати за дедлайнами, бачити прогрес, а ще – нормально спілкуватися між студентами та викладачами. Зараз багато хто вчиться дистанційно чи в змішаному форматі, тож потрібні інструменти, що поєднують зручність, достатню кількість функцій і швидке оновлення інформації про успішність. Це підтверджується сучасними дослідженнями щодо впровадження систем дистанційного навчання у закладах освіти [1].

LMS-платформи на кшталт Moodle [2], Google Classroom [3] чи Prometheus [4] покривають частину зазначених потреб, але не охоплюють їх у повному обсязі. Наприклад, у них немає зручної інтерактивної статистики для відстеження прогресу, функції тестування обмежені (немає, наприклад, таймера або складних типів завдань), дедлайни не дуже гнучкі. Також досі не вирішено питання: як організувати зручну передачу файлів до дедлайну, і щоб ці файли залишилися навіть після того, як студент залишив курс. В результаті – все це ускладнює роботу викладача, знижує швидкість та якість оцінювання. Подібні проблеми відзначаються у порівняльних дослідженнях сучасних LMS [5], а також у роботах присвячених використанню Moodle у дистанційному та змішаному навчанні [6].

Тому варто зібрати усе необхідне в одну веб-систему для управління навчальними курсами: яка об’єднує модуль для тестування з таймером, контроль дедлайнів, робота з завантаженням та зберіганням файлів, система для аналітики прогресу у студентів і все це в одному місці.

Об’єкт дослідження є організація навчання у веб-орієнтованих інформаційних системах.

Предмет дослідження є способи, моделі та інструменти для створення системи управління курсами, що підтримує онлайн-тести, контроль дедлайнів, роботу з файлами й аналітику прогресу.

Метою роботи є розроблення веб-системи управління курсами, яка дозволяє проводити тести з таймером, відслідковувати дедлайни, приймати та зберігати студентські файли (включно з можливістю перездачі до дедлайну), а ще – інтерактивно бачити прогрес, використовуючи технології PHP [7], HTML [8], CSS, JavaScript [9] і Microsoft SQL Server [10;11].

Щоб це реалізувати, треба:

- розібратися з темою та існуючими LMS-платформами: Moodle [2], Google Classroom [3], Prometheus [4];
- відмітити, де у них є проблеми, і сформувати власні вимоги до системи;
- створити архітектуру веб-системи та структуру бази даних [5; 6];
- спроєктувати та реалізувати модулі для управління курсами, матеріалами, завданнями, тестами й дедлайнами;
- продумати, як організувати прийом, збереження і перездачу студентських файлів до дедлайну;
- зробити так, щоб дані про тести та відповіді студентів зберігалися автоматично;
- реалізувати модуль аналітики прогресу із графіками та статистикою;
- перевірити, як система працює, і оцінити її зручність.

Методи дослідження: системний аналіз, порівняльний розбір LMS-платформ [5], проєктування бази даних, роботу з веб-технологіями (PHP [7], HTML [8], CSS, JavaScript [9]), серверні засоби обробки даних, SQL-запити, тестування програмного забезпечення.

Новизна роботи полягає у зробленні система, яка об’єднує управління курсами, тести з таймером, облік дедлайнів, завантаження й зберігання файлів, аналітику прогресу – все в одному середовищі. Особливо важливо: тут є механізм, що дозволяє зберігати файли після того, як студент залишає курс, можливість

перездачі файлів до дедлайну, а також модуль аналітики з базовими графіками та статистикою успішності.

Практична значущість результатів полягає у її застосуванні для організації дистанційного та змішаного навчання у закладах освіти. Система забезпечує легкий доступ до матеріалів, автоматизує тести, спрощує контроль дедлайнів, робить перевірку робіт простішою, допомагає викладачам оперативно слідкувати за прогресом студентів прямо у процесі навчання.

Апробація результатів та публікації. Основні положення і результати бакалаврської роботи доповідались на Всеукраїнських науково практичних конференціях:

1. Єфімов М.О., Аронов А.О., Розробка веб-орієнтованої LMS-системи для дистанційного навчання. // VII Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 23 квітня 2026 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез - К.: ДУІКТ, 2026, С.91-93.
2. Єфімов М.О., Аронов А.О., Особливості розробки модуля онлайн-тестування у веб-орієнтованій системі навчання. // Всеукраїнська науково-технічна конференція «Технології цифрового розвитку: програмні системи та децентралізація», 1–3 травня 2026 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез – (подано до друку)
3. Єфімов М.О., Аронов А.О., Інтелектуальний аналіз у веб-орієнтованій системі навчання. // VI Всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15 травня 2026 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез – (подано до друку)

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1. Характеристика предметної галузі

Цифровізація освіти і популярність дистанційного та змішаного навчання призвели до того, що веб-орієнтовані інформаційні системи стали просто необхідністю для нормальної організації навчального процесу. Learning Management System (LMS) зараз – це вже стандартний інструмент для керування курсами, онлайн-тестами, стеження за дедлайнами, спілкування між студентами й викладачами, а ще для збору статистики з результатів [1].

Сучасні LMS здебільшого вміють стандартні речі: завантажувати навчальні матеріали, створювати завдання, проводити тести, ставити оцінки й заповнювати журнали. Але коли навчальної інформації стає більше, а процеси ускладнюються – стандартних можливостей замало. Звідси й з’являється потреба в додаткових фішках, таких як:

- інтерактивна аналітика прогресу студентів у реальному часі;
- автоматичний контроль дедлайнів для завдань і тестів;
- зручна робота з файлами студентів, включаючи можливість повторно завантажити до дедлайну;
- стійкий модуль онлайн-тестування з таймером, різними типами запитань та збереженням відповідей;
- збереження історії виконаних завдань і результатів тестів;
- єдиний інтерфейс, який збирає всі навчальні елементи в одному середовищі.

Часто LMS не справляються з усіма цими завданнями без додаткових модулів. Наприклад, якщо треба працювати з файлами студентів після завершення курсу чи дозволити повторну здачу до дедлайну – система може підвести. З аналітикою часто взагалі проблема: дані розкидані або зроблені поспіхом, і ніякої

цілісної картини. Все це створює труднощі як для студентів, так і для викладачів: відстежувати прогрес складно, а ефективність навчання помітно падає [5;6].

Якщо коротко, онлайн-навчання передбачає комплексну організацію роботи з курсами, тестами, дедлайнами, файлами та аналітикою. Справді сучасна LMS має природно об’єднувати веб-технології з базами даних, продуманим інтерфейсом та обробкою даних – щоб все було зручно, надійно й інтерактивно. Але не завжди виходить все покрити, частина функцій залишається недостатньо розвиненими. Відповідно, контроль прогресу ускладнюється, і ефективність падає. Щоб цього не траплялося, підхід до розробки має бути комплексним, без пропуску критично важливих моментів.

1.2 Аналіз існуючих аналогів систем дистанційного навчання

Сьогодні дистанційна освіта багато в чому тримається на великих LMS-платформах, які допомагають організувати навчання, тестування, роботу з матеріалами та спілкування між студентами й викладачами. Щоб визначити, що саме потрібно для власної системи, було здійснено порівняння популярні рішення: Moodle [1], Google Classroom [3] та Prometheus [4]. В кожній з цих платформ свої підходи до структури курсів, моделей організації роботи та набору інструментів для підтримки навчальних завдань.

Порівняння показало: які опції вже добре реалізовані в сучасних LMS, а яких функцій або бракує, або вони потребують доопрацювання. Це дало розуміння, як сформулювати вимоги до власної системи, яка має об’єднувати тестування, дедлайни, роботу з файлами й аналітику прогресу в одному середовищі [5].

1.2.1 Аналіз платформи Moodle

Moodle – одна з найпоширеніших систем управління навчанням з відкритим кодом, яку активно використовують у навчальних закладах та бізнесі. Платформа дозволяє працювати з різними форматами навчання: лекції, завдання, тести,

форуми, опитування, SCORM-модулі, журнали оцінок - і ще багато інструментів, потрібних для побудови навчального процесу.

Курс у Moodle має модульну структуру: його ділять на теми, у межах яких викладач додає різні ресурси й активності. Такий підхід дає гнучкість та дозволяє створити як простий, так і складний навчальний сценарій. Функції системи можна розширювати через безліч плагінів - це допомагає адаптувати Moodle під конкретні освітні потреби.

Платформа підтримує створення тестів із різноманітними типами запитань: є одиночний і множинний вибір, співставлення, числові відповіді, есе та інші варіанти. Журнал оцінок допомагає викладачам відстежувати успішність студентів, виставляти підсумкові бали, експортувати результати. Головна сторінка курсу Moodle наведена на рисунку 1.1.

Проте є і моменти, що впливають на зручність роботи. Інтерфейс складний для новачків, деякі функції вимагають окремої установки додаткових модулів. Робота з файлами студентів реалізована базово: повторне надсилання залежить від налаштувань викладача, історію завантажень як окремий інструмент не виділено. Аналітики прогресу за замовчуванням немає - для цього потрібні сторонні розширення [6].

Переваги Moodle:

- підтримує багато типів навчальних активностей;
- можна створювати складні тести;
- гнучка система ролей і доступу;
- активна спільнота та постійні оновлення;
- можливість додавати плагіни для нових функцій;
- є журнали оцінок і звіти.

Недоліки Moodle:

- інтерфейс незручний для нових користувачів;
- залежність від плагінів для частини функцій;
- базова робота з файлами студентів без розширених функцій;
- аналітика прогресу відсутня без додаткових модулів;

- повторне подання файлів залежить від налаштувань викладача.

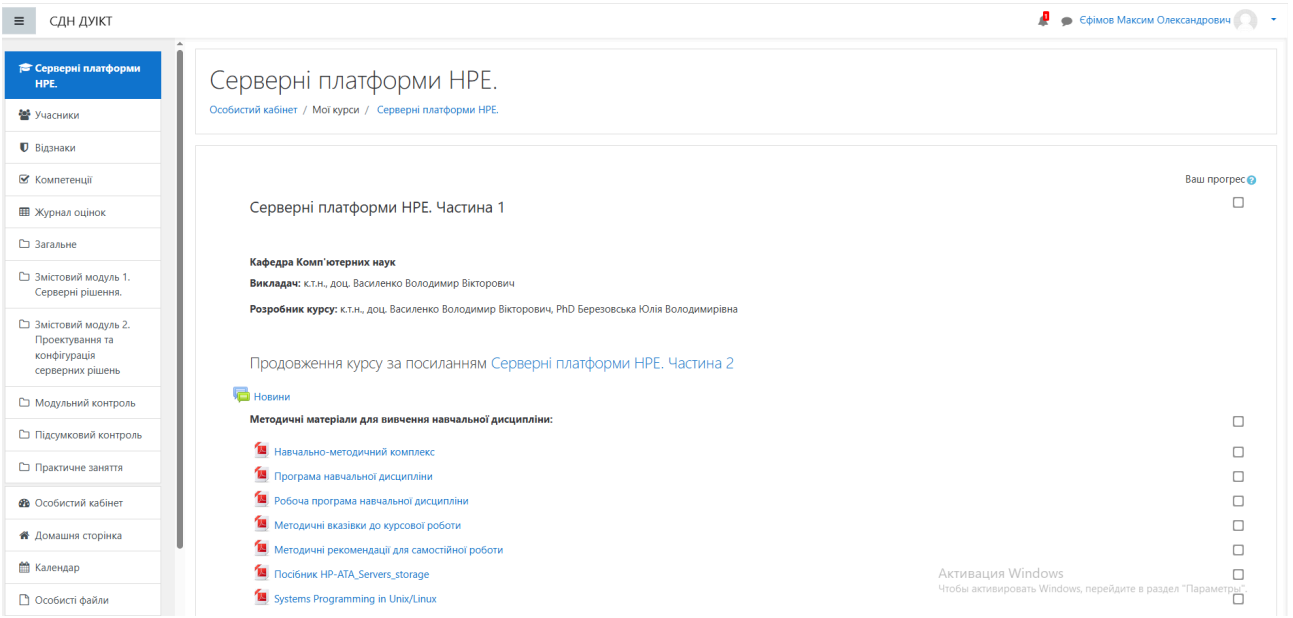


Рис. 1.1 Головна сторінка курсу в системі Moodle

1.2.2 Аналіз платформи Google Classroom

Google Classroom – хмарна платформа для організації навчального процесу, інтегрована з екосистемою Google Workspace [3]. Вона орієнтована на простоту використання та швидку взаємодію між викладачами й студентами. Classroom дозволяє створювати завдання, публікувати оголошення, коментувати роботи, прикріплювати файли та використовувати Google Drive як основне сховище навчальних матеріалів.

Структура курсу складається з трьох основних розділів: «Стрічка», «Завдання» та «Люди». У розділі «Завдання» викладач може створювати різні типи активностей: завдання з файлами, матеріали для перегляду та тести, які реалізовані через Google Forms. Усі файли автоматично зберігаються у Google Drive, що забезпечує зручний доступ і централізоване зберігання.

Google Classroom широко використовується у школах завдяки простому інтерфейсу, мінімальним вимогам до налаштування та тісній інтеграції з іншими сервісами Google. Платформа підтримує коментування робіт, виставлення оцінок,

перегляд статусу виконання завдань і спільну роботу з документами. Інтерфейс Google Classroom наведено на рисунку 1.2.

Разом із тим Classroom не є повноцінною LMS у класичному розумінні. Платформа не підтримує складні навчальні сценарії та модульну структуру курсів. Онлайн-тестування реалізоване через Google Forms, що обмежує можливості створення складних типів запитань і не передбачає таймера у межах самої LMS [5]. Аналітика успішності подана у спрощеному вигляді та не містить розширених інструментів для аналізу прогресу [6]. Робота з файлами повністю залежить від Google Drive, а після виходу студента з курсу доступ до його робіт може бути втрачений.

Переваги Google Classroom:

- простий та інтуїтивний інтерфейс;
- зручна робота з файлами через Google Drive;
- швидке створення завдань;
- автоматичне збереження матеріалів у хмарі;
- інтеграція з Google Docs, Sheets, Forms;
- доступність на різних пристроях.

Недоліки Google Classroom:

- система не підтримує складні навчальні сценарії;
- тестування реалізоване через Google Forms і має обмежений функціонал;
- відсутність таймера тестування у межах LMS;
- аналітика успішності представлена у спрощеному вигляді;
- структура курсів не передбачає модульності;
- робота з файлами повністю залежить від Google Drive;
- студент не може самостійно вийти з курсу, а у разі його видалення або зміни прав доступу файли можуть стати недоступними.

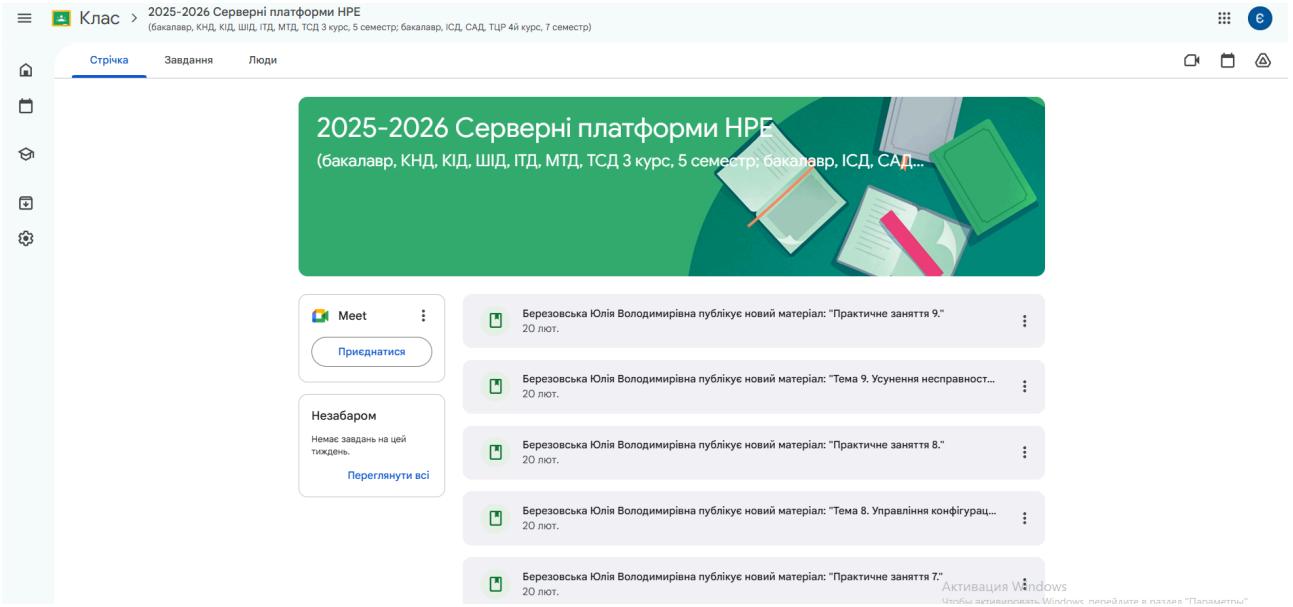


Рис. 1.2 Інтерфейс курсу Google Classroom

1.2.3 Аналіз платформи Prometheus

Prometheus - українська платформа масових відкритих онлайн-курсів (МООС), де навчальні матеріали подаються через відеолекції, тести та інтерактивні завдання [4]. Тут можна вчитися у власному темпі, що особливо цінують ті, хто націлений здобувати знання самостійно.

Курс розбитий на модулі, у кожному з яких чекають відеолекції, додаткові матеріали та перевірочні завдання. Пройшовши курс до кінця, користувач отримує електронний сертифікат. Prometheus об’єднує зусилля з університетами й державними структурами, щоб забезпечити якість своїх програм. Головна сторінка на платформі Prometheus наведена на рисунку 1.3.

Попри це, Prometheus - не класична LMS. Платформа фактично не працює із навчальними групами, викладачами або оцінюванням у старому розумінні. Немає підтримки роботи з файлами, фіксації дедлайнів і перевірки студентських завдань. Аналітика успіхів тут проста, а створювати власні курси прямо на платформі не можна [5;6]. Основна мета - дати доступ до матеріалів для самостійного навчання, а не організовувати повноцінний навчальний процес у межах закладу освіти.

Переваги Prometheus:

- простий зрозумілий інтерфейс;

- багато безкоштовних курсів;
- можливість працювати у власному темпі;
- якісні відеолекції;
- тестування після модулів.

Недоліки Prometheus:

- платформа не є LMS у класичному розумінні;
- курс побудований переважно на відеолекціях і тестах;
- майже немає взаємодії між студентом і викладачем;
- аналітика доволі базова;
- відсутні механізми контролю дедлайнів;
- не можна завантажувати і перевіряти студентські файли;
- користувачі не мають доступу до створення власних курсів.

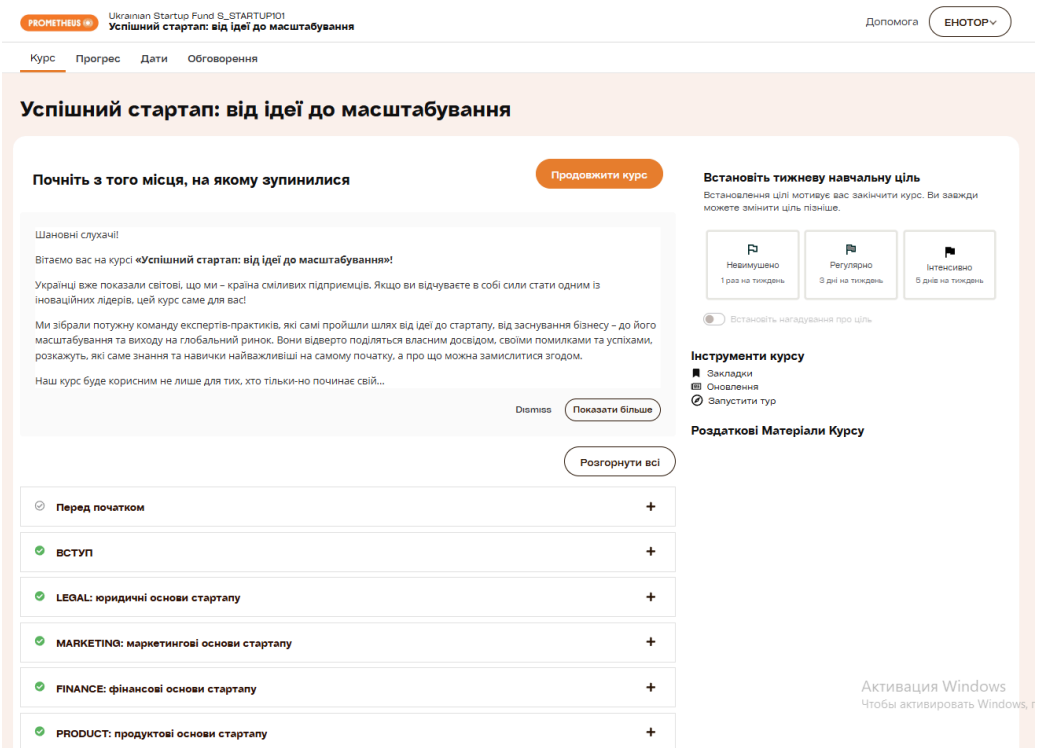


Рис. 1.3 Головна сторінка курсу на платформі Prometheus

Щоб підсумувати аналіз, було проведено порівняння трьох популярних платформ для дистанційного навчання - Moodle [2], Google Classroom [3] та

Prometheus [4] - із власною розробленою системою. Порівняння наведено в таблиці 1.1.

Таблиця 1.1

Порівняння LMS-систем за ключовимим характеристиками

Критерій	**Moodle**	**Google Classroom**	**Prometheus**	**Розроблена система**
Простота інтерфейсу	– складний	+ дуже простий	+ простий	+ простий
Гнучкість тестування	+ багато типів питань, таймер	– обмежено	– базово	+ таймер, типи питань
Дедлайни	– негнучкі	+ гнучкі	– базові	+ гнучкі
Передача файлів	– залежить від налаштувань	+ можна	– немає	+ можна до дедлайнів
Робота з файлами	– базово	+ Drive	– залежить від курсу	+ збереження файлу після виходу з курсу
Аналітика	– потребує плагінів	– мінімальна	– мінімальна	+ базова інтерактивна аналітика
Розширюваність	+ плагіни	– обмежена	– обмежена	+ модульна
Поріг входу	– високий	+ низький	+ низький	+ низький

Як видно з таблиці, жодна з розглянутих платформ не поєднує одночасно гнучкі дедлайни, можливість передачі файлів до дедлайну, інтерактивну аналітику та розширені можливості тестування. Це обґрунтовує необхідність розробки власної LMS, яка усуває виявлені недоліки аналогів.

1.3 Визначення проблем предметної галузі

Під час аналізу сучасних LMS-платформ спливла ціла низка труднощів, які ускладнюють проведення навчання у веб-середовищі. Навіть попри широке впровадження подібних систем, більшість із них реалізують ці можливості неповноцінно: тести, завдання, дедлайни, аналітика, робота з файлами часто залишаються розрізненими та не складаються у цілісну структуру [5].

Часто можна зіткнутися з дуже обмеженим функціоналом для роботи з файлами студентів. У багатьох системах передати файл доводиться у найпростішому вигляді, після чого варіантів взаємодії майже немає. Типова ситуація: не видно, коли саме була надіслана робота, повторне завантаження файлу є незручним, а самі матеріали можуть зберігатися на зовнішніх сервісах, що ускладнює контроль над матеріалами [6].

Ще одна проблема - дедлайни. У частині LMS їх задають лише в спрощеному варіанті: без календаря, без зведеного списку термінів. Так і студенти заплутуються з плануванням, і викладачі - з перевірками та своєчасністю здачі [5].

З аналітикою теж негусто: зазвичай це просто перелік оцінок без будь-яких діаграм, статистики чи деталей. Оцінити прогрес або зрозуміти, в яких темах виникають труднощі, майже неможливо [6].

У розділі онлайн-тестування ситуація подібна: частина платформ дозволяє створювати лише прості типи запитань, не дає таймерів, відповіді не зберігаються в самій LMS. Через це можливостей для варіацій і аналізу результатів майже немає [6].

Дратує і те, що немає цілісної структури курсу, де все зібране разом - матеріали, завдання, тести, дедлайни, результати. Замість цього користувачі постійно перескакують між різними розділами або навіть сервісами, і все це створює додаткові труднощі в роботі [5].

Загалом проблеми цієї сфери можна звести до такого:

- незручна робота з файлами і контролем подачі;
- розрізнене або малоінформативне відображення дедлайнів;
- мінімальна функціональність тестування;
- відсутність живої, інтерактивної аналітики прогресу;
- фрагментована структура навчальних матеріалів.

Усе це ще раз доводить - потрібна веб-орієнтована інформаційна система, яка нарешті об’єднає процес навчання в одну зручну, логічно побудовану і справді працюючу цілісність, без тих недоліків, що зустрічаються у звичних рішеннях.

1.4 Постановка завдання на розробку системи

Після знайомства з предметною галуззю, огляду літератури й порівняння платформ Moodle, Google Classroom і Prometheus стало видно: більшість LMS досі мають одні й ті самі недоліки. Зручність роботи зі студентськими файлами слабка, контроль за дедлайнами майже завжди поверховий, аналітика прогресу на рівні "для галочки", інструменти тестування обмежені [5;6]. Усе це ускладнює організацію навчання, породжує непорозуміння між студентами й викладачами і взагалі не додає ефективності процесу.

Ця кваліфікаційна робота ставить собі мету створити веб-систему для керування навчальними курсами, яка охоплює весь процес - від роботи з навчальними матеріалами до дедлайнів і аналітики. Ставка саме на ті можливості, які в популярних платформах відсутні або реалізовані половинчасто: повторне завантаження файлів до дедлайну, збереження робіт навіть після того, як студент покинув курс, нормальне та зрозуміле відображення прогресу, онлайн-тести з таймером.

Для виконання завдання планується:

- зробити модуль управління курсами: створення, додавання матеріалів, перегляд структури;

- створити модуль завдань, що дасть змогу студентам завантажувати файли, подавати їх повторно до дедлайну, а викладачу - переглядати роботи й писати коментарі до оцінки;
- реалізувати модуль онлайн-тестування з різними типами питань, таймером, автоматичним підрахунком результатів; студент зможе бачити власні відповіді;
- вбудувати календар дедлайнів, щоб студенти бачили терміни, а викладач міг контролювати, хто і що здав вчасно;
- додати модуль аналітики: результати тестів, виконання завдань, загальний прогрес - усе це буде показане через інтерактивні графіки й прості показники;
- реалізувати систему ролей із власним інтерфейсом для студента, викладача та адміністратора і різними правами доступу;
- подбати про захист даних та авторизацію: перевіряти доступ, забезпечити приватність персональних даних;
- спроєктувати базу даних, яка надійно триматиме інформацію про курси, матеріали, завдання, тести, результати й студентські файли;
- застосувати сучасний стек: PHP, HTML, CSS, JavaScript та Microsoft SQL Server - це дозволить створити зручну, продуктивну та масштабовану систему.

Завдання - зробити систему, що поєднує управління матеріалами, контроль дедлайнів, онлайн-тести та оцінювання. Головне тут - зручність для користувача, акуратна робота з файлами, точні результати й коректна аналітика.

У результаті реалізації має вийти повноцінна веб-система, яка спростить навчальний процес для студентів і викладачів - усі курси, завдання та тести опиняться в одному місці й будуть доступні в будь-який момент.

1.5 Визначення вимог до системи

Після того, як було проаналізовано основні проблеми та цілі проекту, стало ясно, які функції й якості потрібні майбутній веб-системі для керування навчальними курсами. Ці вимоги враховують сучасні особливості LMS-платформ, їхні слабкі місця та побажання основних користувачів: студентів, викладачів і адміністратора [5;6]. Всі вимоги було поділено на дві групи: функціональні та нефункціональні.

Функціональні вимоги:

1. Система дозволяє створювати акаунти, входити та визначати рівень доступу для різних ролей: студент, викладач або адміністратор.
2. Викладачі можуть створювати курси, додавати матеріали, редагувати структуру курсу й бачити, хто зі студентів приєднався.
3. Є можливість завантажувати й переглядати матеріали курсу (файли, документи, зображення), розташовані у зручному порядку.
4. Студенти можуть надсилати файли із виконаними завданнями, переглядати свої роботи та надсилати оновлення до завершення дедлайну.
5. Викладачі переглядають подані завдання, виставляють оцінки й залишають коментарі.
6. Файли студентів залишаються доступними, навіть якщо студент залишає курс.
7. Система показує строки виконання завдань і тестів у вигляді календаря або списку дедлайнів, який бачать і студенти, і викладачі.
8. Можна створювати тести з різними типами питань, студенти проходять їх, працює таймер, відповіді зберігаються, система формує результати.
9. Викладач бачить результати тестів і завдань, а також основні показники успішності студентів.

10. Адміністратор має змогу керувати користувачами та переглядати події, які стосуються виходу студентів із курсу.

Нефункціональні вимоги:

1. Система захищає акаунти, коректно працює авторизація та чітко розмежовано доступ за ролями.
2. Вся інформація - курси, матеріали, завдання, результати тестів і студентські файли - надійно зберігається у базі даних без ризику втрати.
3. Система стабільно працює при роботі багатьох користувачів, а основні сторінки відкриваються швидко.
4. Інтерфейс правильно відображається в сучасних браузерах (Google Chrome, Mozilla Firefox, Microsoft Edge).
5. Архітектура системи дозволяє додавати нові можливості без серйозних змін на майбутнє.
6. Інтерфейс має бути простий і зрозумілий для студентів і викладачів, з логічною навігацією й мінімальною кількістю кроків для основних дій.

Ці вимоги стануть базою для проєктування архітектури, розробки модулів та реалізації функцій. Все це потрібно, щоб організувати ефективне й зручне навчання онлайн.

2 АНАЛІЗ ЗАСОБІВ РЕАЛІЗАЦІЇ ТА СТРУКТУРИ ПРОГРАМНОЇ СИСТЕМИ

2.1 Середовище розробки

Для створення веб-системи керування навчальними курсами було використано локальне середовище Laragon. Воно дозволяє швидко запускати PHP-проєкти, автоматично налаштовує всі потрібні серверні компоненти і дає зручні інструменти для роботи з базами даних. Завдяки Laragon не треба возитися із ручною конфігурацією веб-сервера, тож застосунок стабільно працює на всіх етапах розробки. Головне вікно середовища Laragon наведено на рисунку 2.1.

У складі Laragon вже є Apache, PHP-інтерпретатор і інструменти для адміністрування баз - це робить середовище гарним вибором для локальної розробки PHP-застосунків. Ще Laragon автоматично створює віртуальні хости, швидко піднімає сервери і дозволяє доволі просто налаштувати потрібні параметри для кожного проєкту.

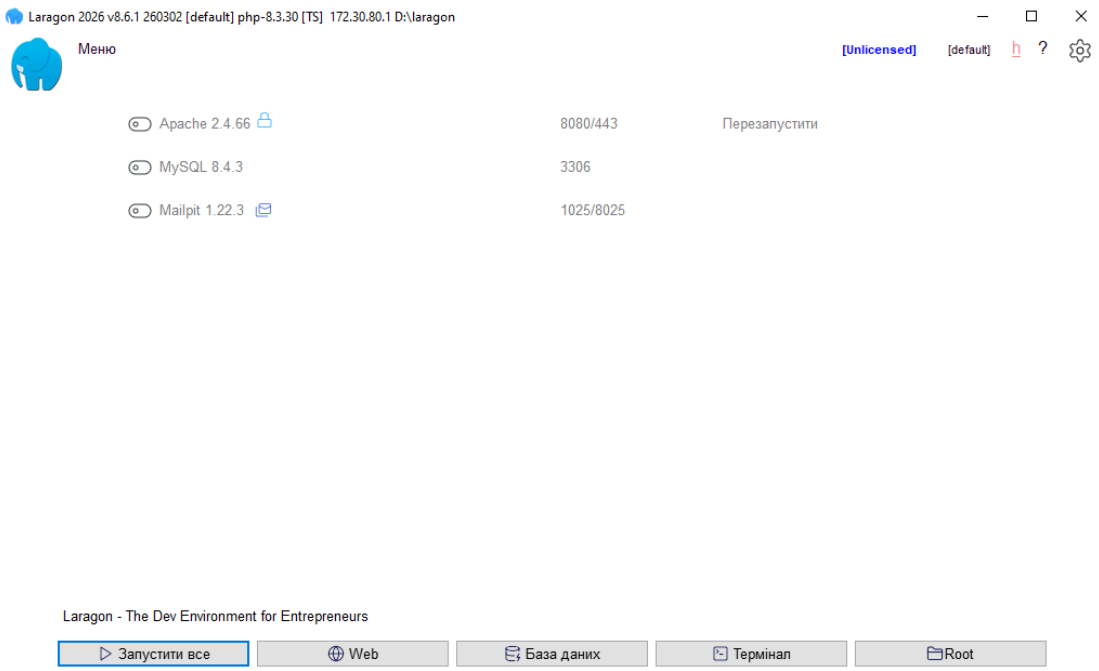


Рис. 2.1 Головне вікно середовища Laragon

Для написання коду використовувалося середовище Visual Studio Code: тут зручну управляти файлами проєкту, працювати з PHP, HTML, CSS і JavaScript, а розширення допомагають підключити роботу з Microsoft SQL Server. Виходить універсальний інструмент для розробки як серверної, так і клієнтської частини системи. У редакторі просто перемикатися між файлами, дивитися структуру, працювати з терміналом і налагоджувати код - це відчутно прискорює розробку і дозволяє комфортно тримати під контролем масивні модулі. Робоче середовище Visual Studio Code під час розробки LMS показано на рисунку 2.2.

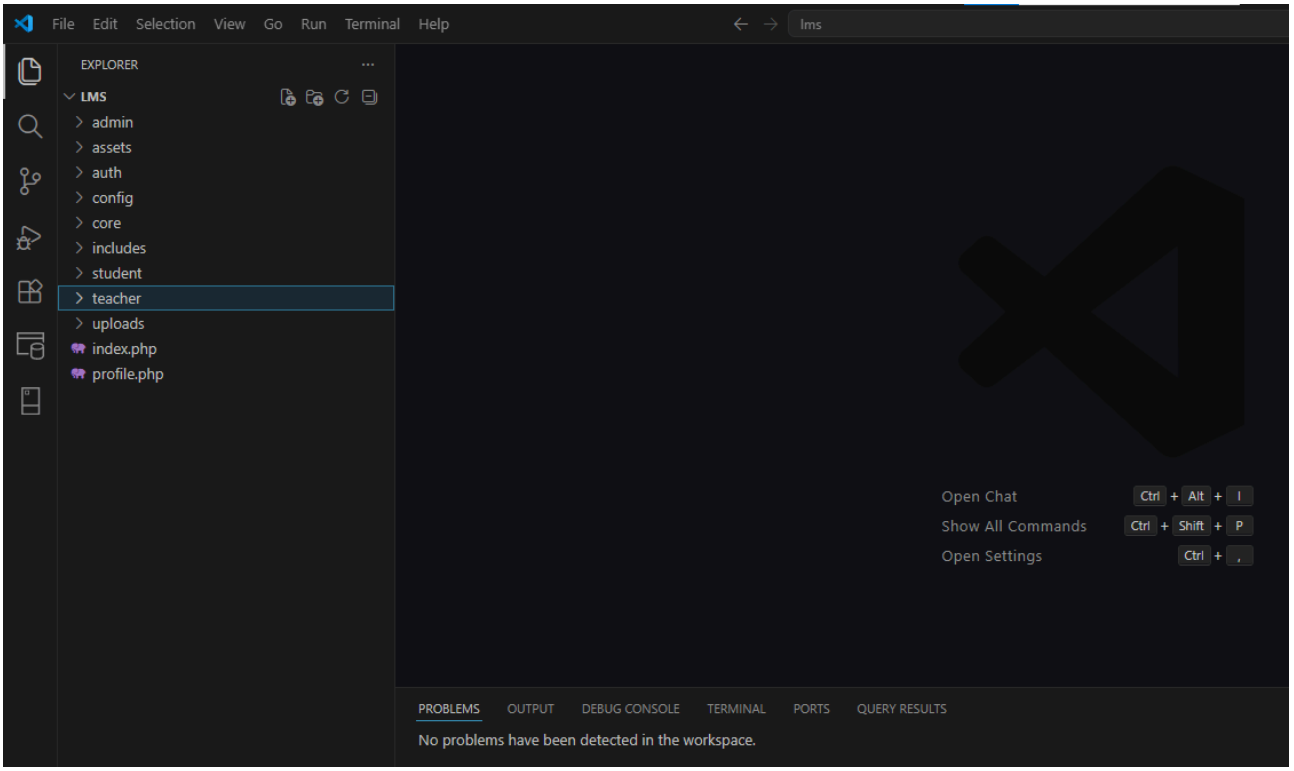


Рис. 2.2 Робоче середовище Visual Studio Code під час розробки LMS

Для керування базою даних було використано Microsoft SQL Server Management Studio (SSMS) - з його допомогою створюються таблиці, виконуються SQL-запити, редагуються дані та здійснюється адмініструванням. В SSMS зручно стежити за структурою, контролювати зв’язки між таблицями і оптимізувати

запити, коли це потрібно. Інтерфейс SSMS з відкритою базою даних LMS наведено на рисунку 2.3.

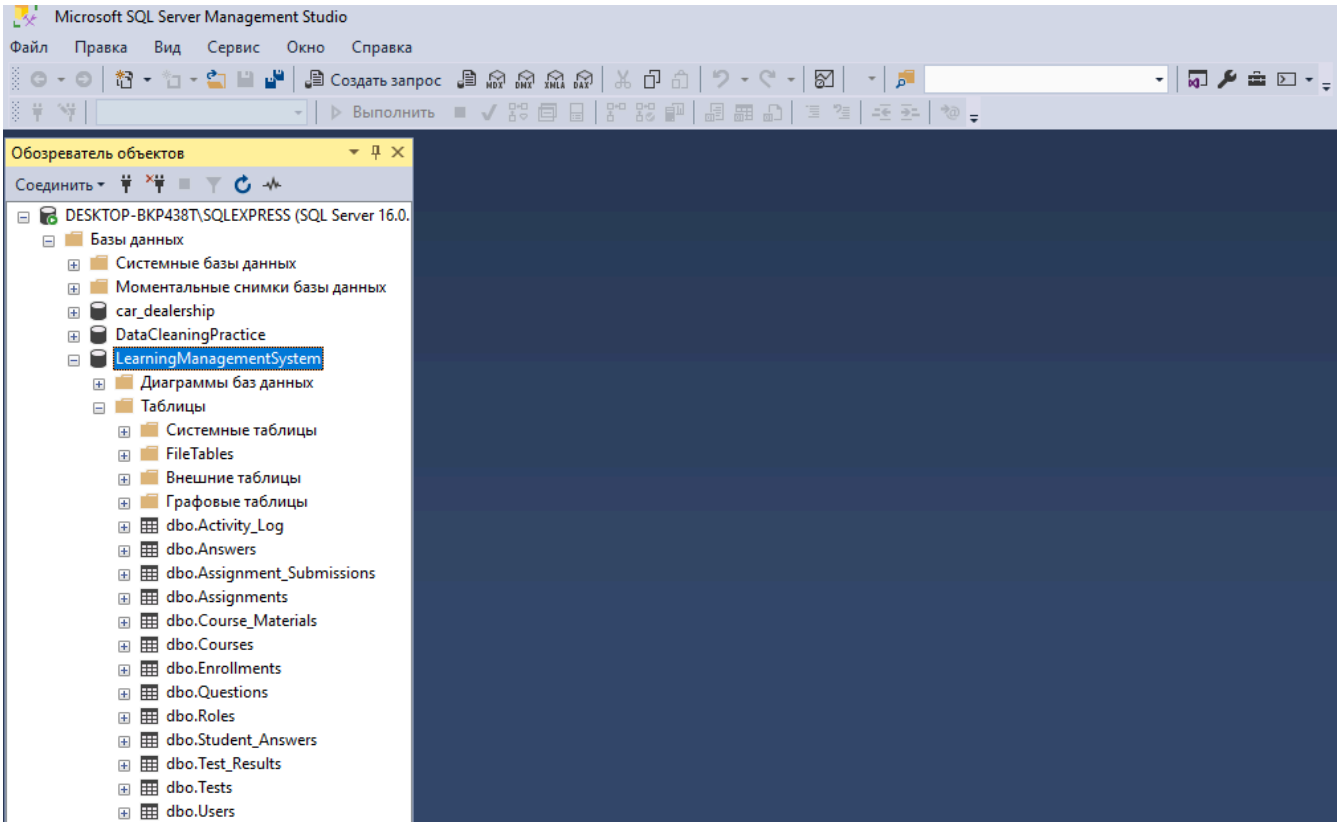


Рис. 2.3 Інтерфейс SQL Server Management Studio з відкритою базою даних LMS

Отже, комбінація Laragon, Visual Studio Code і SQL Server Management Studio дозволяє пройти весь цикл розробки веб-додатку: від коду і тестування до роботи з базою даних. Завдяки цим інструментам вдалося зібрати стабільну, керовану й масштабовану систему, яка точно підходить для освітньої роботи.

2.2 Вибір технологій та архітектурних рішень

Щоб система керування курсами працювала впевнено, не вимагала зайвих зусиль при встановленні і могла розширюватися далі, було сформовано відповідний стек технологій. Він охоплює інструменти для онлайн-тестування,

роботи з файлами, відстеження дедлайнів і можливістю аналізу виконання завдань.

Основна мова для серверної частини - PHP. Вона дає можливість закласти всю логіку обробки запитів, авторизацію користувачів, зв’язок із базами даних і формування динамічних сторінок. Драйвери sqlsrv забезпечують сумісність PHP із Microsoft SQL Server [11], а це важливо для побудови навчальних систем, які працюють із великими масивами структурованих даних. Для опанування базових принципів PHP використовувалися навчальні матеріали [12].

Клієнтська частина системи реалізована за допомогою стандартних веб-технологій:

- HTML5 – для структури сторінок;
- CSS3 – для стилізації інтерфейсу та адаптивності;
- JavaScript – для інтерактивних елементів, таймера тестування, перевірки форм та роботи з графіками.

Щоб візуалізувати результати тестів та загальний прогрес, використовується Chart.js [13] - ця бібліотека допомагає швидко організувати інтерактивні діаграми і графіки. Дані стають наочнішими, а студентам і викладачам зручніше аналізувати результати.

Усі дані зберігаються в Microsoft SQL Server. Це дає:

- надійний захист і збереження всіх даних по курсах, завданням, тестам та результатам;
- підтримку складних SQL-запитів;
- простір для розширення системи;
- швидку роботу з великими обсягами інформації.

Архітектуру проєкту побудовано модульно. Кожен модуль (керування курсами, матеріалами, завданнями, тестуванням, дедлайнами, аналітикою) працює незалежно, а взаємодіють вони через спільну базу даних. Це дозволяє спокійно додавати новий функціонал або змінювати існуючий, не чіпаючи всю структуру системи.

2.3 Реалізація безпеки системи

Безпека для веб-орієнтованих навчальних систем - питання серйозне, адже тут зберігаються і персональні дані користувачів, і результати тестування, і файли із завданнями, і багато іншої конфіденційної інформації. У цій системі було реалізовано комплекс рішень, щоб захистити дані, розмежувати права доступу та уникнути збоїв між різними модулями. Схему розмежування доступу між ролями наведено на рисунку 2.4.

Спершу було розпочато із автентифікації: паролі користувачів не зберігаються у відкритому вигляді, а одразу хешуються функцією password_hash(). Навіть якщо хтось отримає доступ до бази - витягти справжні паролі не вийде. Перевірка під час входу здійснюється через password_verify(), яка співставляє пароль, введений користувачем, із хешованим значенням у базі.

Ще один важливий момент - розподіл доступу відповідно до ролей. У системі є три варіанти: студент, викладач і адміністратор. Для кожної ролі - свій набір дозволів. Наприклад, студент бачить навчальні матеріали та проходить тести, викладач створює нові завдання і перевіряє роботи, а адміністратор керує користувачами та має доступ до інформації про виході студентів із курсів.

Рис. 2.4 Схема розмежування доступу між ролями користувачів

Щоб ніхто не міг зламати базу через SQL-ін’єкції, усі SQL-запити виконуються у вигляді підготовлених інструкцій із параметрами через драйвер sqlsrv [11]. Так система впевнено захищає себе від некоректних або шкідливих запитів і відповідає рекомендаціям Microsoft SQL Server [10].

Ретельно перевіряються й усі дані, які приходять від користувачів. Перевірка працює як у браузері (через JavaScript), так і на сервері (PHP), тож некоректно введена або небезпечна інформація не потривожить роботу системи.

Перед тим як завантажити чи відкрити студентський файл, система щоразу перевіряє права доступу. Користувач отримує доступ лише до тих файлів, які належать до курсів, у яких він записаний.

Усе, що стосується виходу студентів із курсів, фіксується у таблиці Activity_Log. Адміністратор може переглядати ці дані та стежити за активністю,

проте це не розгорнутий журнал дій, а скоріше окрема функція контролю за відвідуванням курсів.

Завдяки цим підходам забезпечується захист персональної інформації, чіткий розподіл доступу, стійкість проти SQL-ін’єкцій та стабільна робота навіть при великій кількості одночасних користувачів.

2.4 Клієнтська частина та інтерфейс системи

Клієнтська частина LMS - це ключовий елемент взаємодії користувача з усіма можливостями системи. Інтерфейс було розроблено простим, інтуїтивним і без зайвих кроків для виконання основних дій. Весь акцент - на зручності для студентів і викладачів та швидкому доступі до матеріалів, завдань, тестів і аналітики. Дашборд користувача наведено на рисунку 2.5.

Всі основні елементи розділені по модулях. Курси, матеріали, завдання, тести, дедлайни і аналітика - це окремі сторінки, де все розкладено по своїх місцях, без зайвої плутанини. Тому шукати потрібну інформацію або переходити між розділами просто.

Після входу користувач відразу потрапляє на дашборд. Тут усе залежить від ролі: студенти бачать свої останні оцінки, календар дедлайнів і можуть долучитися до нового курсу. Викладачам відображається графік середньої успішності студентів і календар по роботах, які вони самі створили. Основна ідея - вся найважливіша інформація перед очима без зайвих кліків.

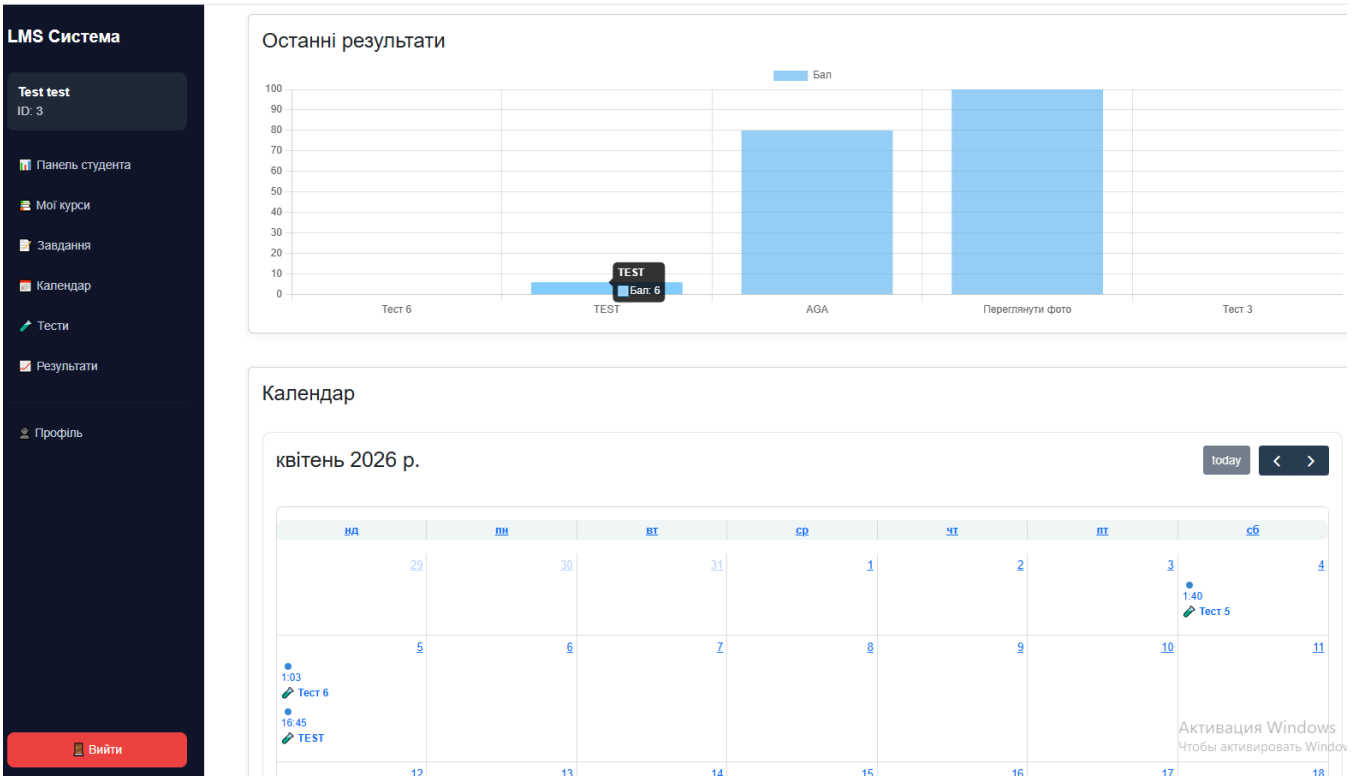


Рис. 2.5 Дашборд користувача в системі

Сторінка кожного курсу має панель навігації з розділами "Матеріали", "Завдання", "Тести" та "Аналітика". Так усе в навчальному процесі об’єднано в одному місці - не потрібно постійно переключатися на інші сторінки.

У модулі завдань студенти бачать, що і коли здати, можуть завантажити або повторно відправити свою роботу до дедлайну. Викладачі переглядають подані файли, ставлять оцінки, залишають коментарі. Все побудовано так, щоб не треба було виконувати зайвих дій для перевірки чи завантаження.

У тестовому модулі є таймер, який показує, скільки часу залишилося. По завершенню студент моментально отримує результат, викладачі можуть подивитись статистику результатів тесту. Інтерфейс відображення результатів тесту наведено на рисунку 2.6.

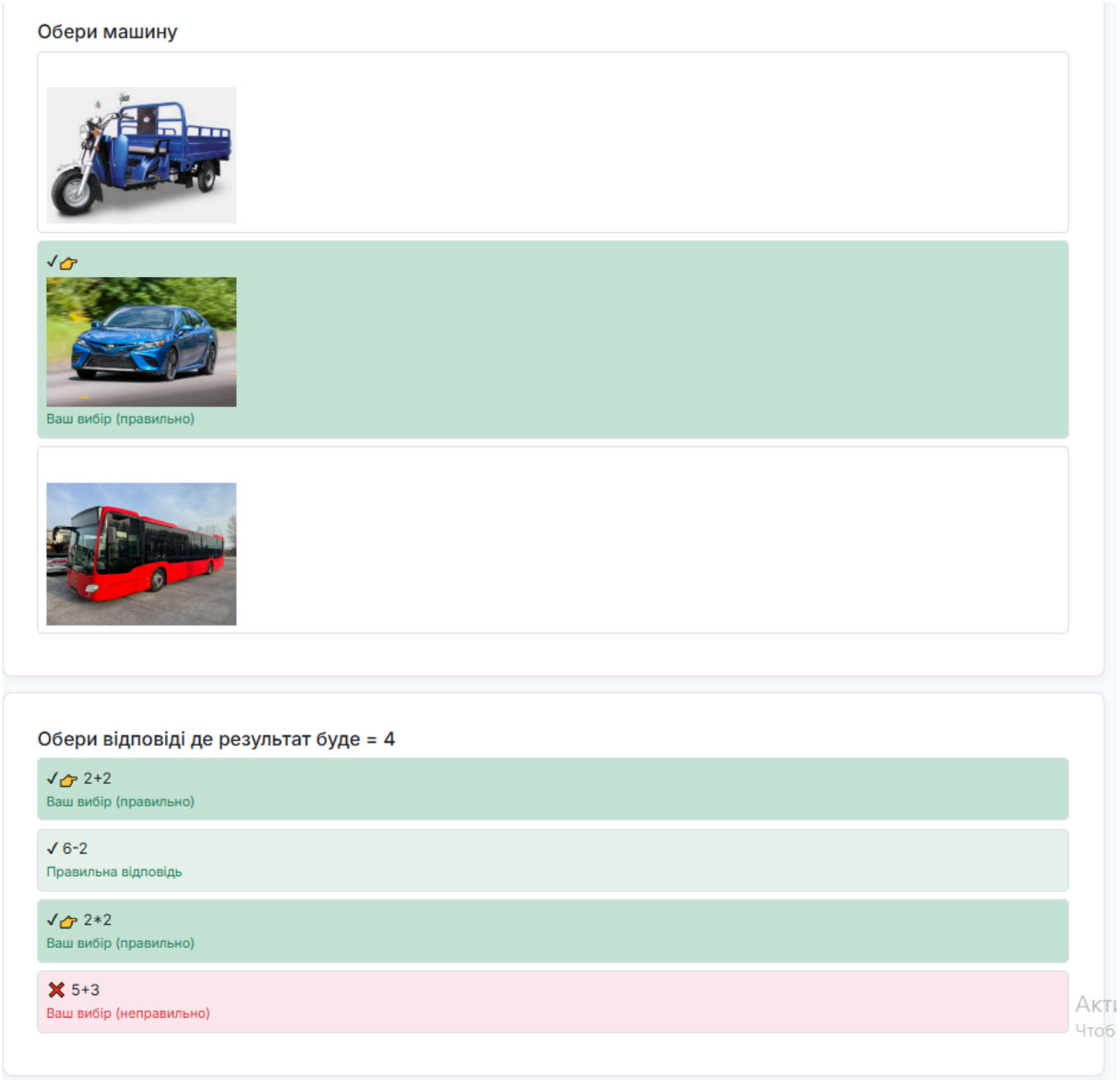


Рис. 2.6 Інтерфейс результату теста правильні/неправильні відповіді

За аналітику відповідає окремий модуль - тут прогрес студентів можна подивитись у вигляді графіків та діаграм на базі Chart.js [13]. Все працює інтерактивно й наочно: видно середні бали, скільки завдань виконано, результати тестів і загальну динаміку. Аналітичну панель викладача наведено на рисунку 2.7.

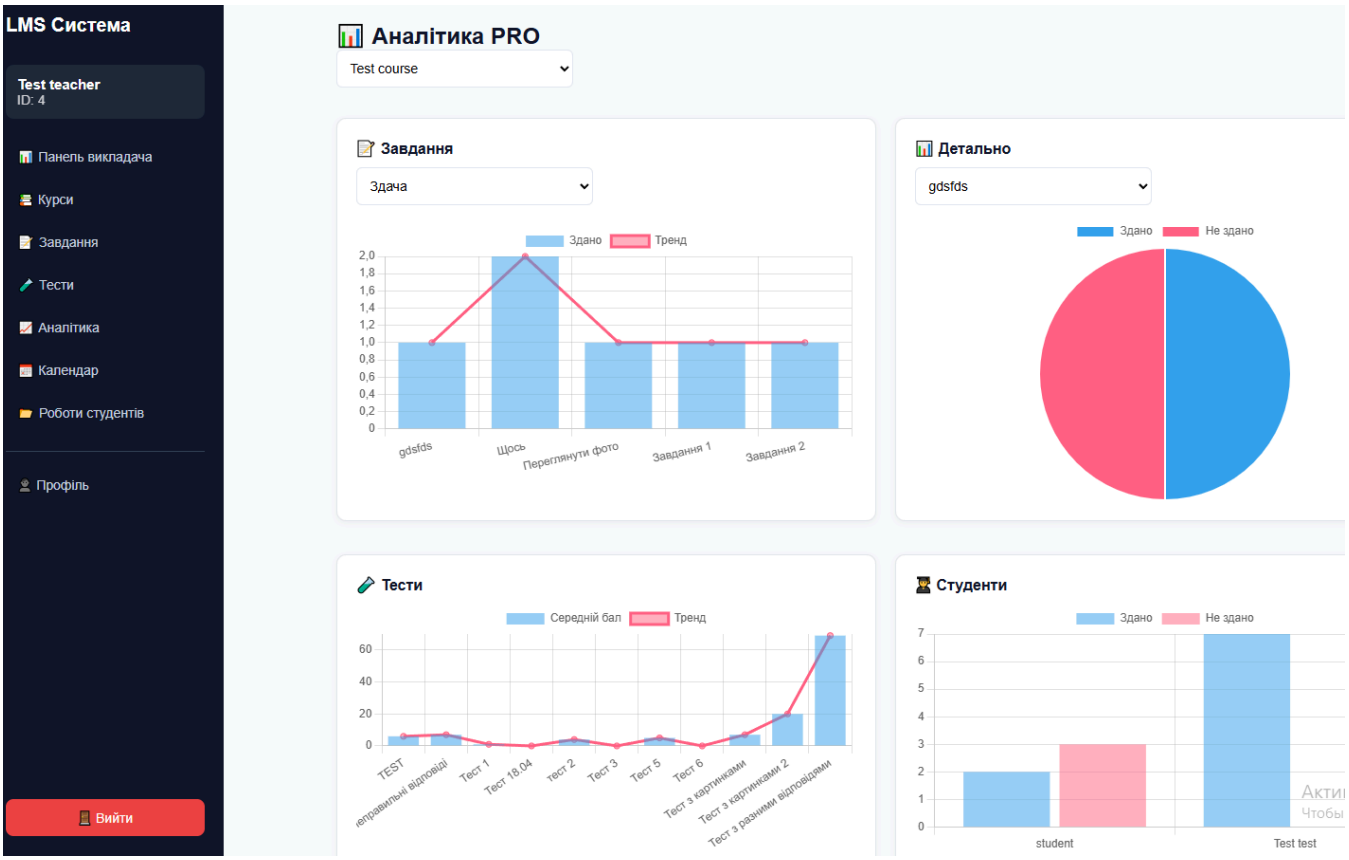


Рис. 2.7 Аналітична панель викладача

Стилі оформлені на CSS3, плюс для інтерактивності підключено JavaScript. Вийшов зрозумілий і логічний інтерфейс, у якому швидко перемикатись між модулями просто, незалежно від того, студент ти чи викладач.

По суті, клієнтська частина LMS дає інтуїтивне керування навчанням: все під рукою, працює швидко та не змушує довго розбиратись у деталях.

2.5 Робота з базою даних

База даних лежить у центрі цієї системи: саме тут зберігається все - від інформації про користувачів і курси до навчальних матеріалів, завдань, тестів і результатів. Для серверної частини було обрано Microsoft SQL Server, що забезпечує надійність, високу продуктивність і дозволяє працювати з непростими запитами. Структуру бази даних системи наведено на рисунку 2.8.

Структура бази даних побудована на реляційній моделі: таблиці між собою пов’язані. Головна таблиця - Users. Тут зберігаються всі дані про студентів, викладачів та адміністратора. Роль кожного визначає, що йому дозволено в системі й до яких модулів він має доступ.

Інформація про курси знаходиться в таблиці Courses. А таблиця Enrollments відображає відносини "багато-до-багатьох": завдяки їй студенти можуть підписатися на кілька курсів, а викладачі - вести різні дисципліни одразу.

Матеріали курсу містяться в таблиці Course_Materials: тут і назва, й опис, і шлях до файлу. Завдання зберігаються у Assignments, а виконані студентами роботи у Assigment_Submissions - із зазначенням файлу, дати здачі та оцінки.

Модуль тестування реалізовано через таблиці Tests, Questions, Answers та Results.

Тести реалізовані через кілька таблиць:

- Tests включає загальну інформацію про тест,
- Questions - перелік запитань для кожного тесту,
- Answers - студентські відповіді,
- Results - підсумкові бали.

Така структура дозволяє зберегти як саму "архітектуру" тестування, так і всі деталі по кожній відповіді.

Події виходу студентів із курсів фіксуються через таблицю Activity_Log. Адміністратор бачить цю інформацію й завдяки їй стежить за активністю користувачів у дисциплінах.

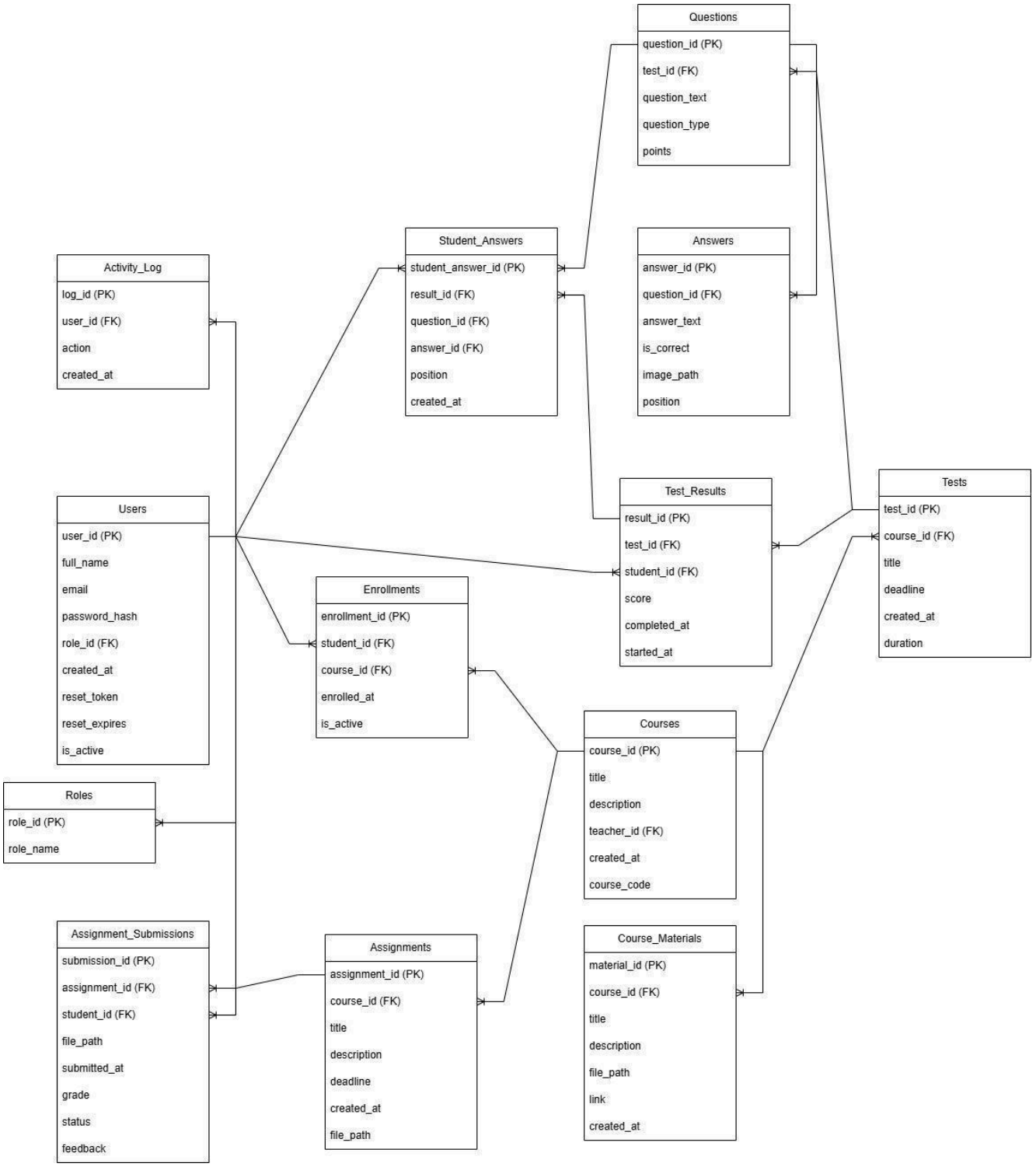


Рис. 2.8 Структура бази даних системи (ER-діаграма)

Зв’язки між таблицями продумані так, щоб дані були цілісними, а всі модулі системи працювали злагоджено. Реляційна модель дає можливість швидко отримувати потрібну вибірку, вести статистику, обробляти результати тестів і підтримувати синхронність даних між різними частинами системи.

Для взаємодії PHP із SQL Server використовуються драйвери sqlsrv [11], що забезпечують стабільну роботу з параметризованими запитами, транзакціями та великим обсягом даних.

2.6 Архітектура програмної системи

Архітектура цієї програмної системи модульна: функції розбиті на окремі логічні частини. Це помітно полегшує розробку, тестування і подальше розширення - кожен модуль можна змінювати незалежно від інших.

Система побудована на трьох основних рівнях: клієнтський, серверний і база даних. Загальну архітектуру програмної системи навелено на рисунку 2.9.

Клієнтський рівень - це весь інтерфейс і робота користувача з LMS. На серверному реалізована бізнес-логіка: обробка запитів, перевірка прав, зв’язок із базою даних. Рівень бази даних зберігає інформацію про користувачів, курси, матеріали, завдання, тести та результати.

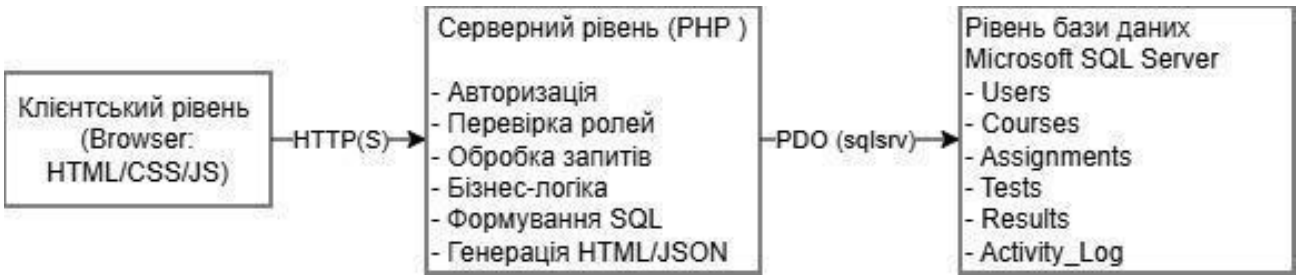


Рис. 2.9 Загальна архітектура програмної системи

Клієнтська частина надсилає запити до серверу, який побудований на PHP. Сервер приймає ці дані, перевіряє ролі користувачів, формує SQL-запити і відправляє їх у базу. Результати повертаються на клієнт HTML-сторінками або JSON (наприклад, для генерації графіків у аналітичному модулі).

Модульна структура системи включає такі компоненти:

- Модуль користувачів - реєстрація, авторизація, визначення ролей.

- Модуль курсів - створення курсів, приєднання студентів, перегляд матеріалів.
- Модуль завдань - створення, здача робіт, оцінювання.
- Модуль тестування - створення тестів, проходження, збереження відповідей та результатів.
- Модуль аналітики - побудова графіків успішності на базі зібраних даних.
- Модуль дедлайнів - відображення важливих дат у календарі студентів і викладачів.
- Модуль Activity_Log - фіксація випадків виходу студентів із курсу.

Усі модулі мають доступ до спільної бази даних, але кожен працює за своїми правилами та має власні серверні обробники. Такий підхід дає змогу додавати нові можливості, не зачіпаючи основну структуру.

Завдяки цьому архітектура забезпечує стабільність, цілісність даних і дозволяє масштабувати систему. Чіткість поділу рівнів і модулів допомагає тримати все зрозумілим, керованим і готовим до змін у майбутньому.

3 РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

3.1 Проєктування структури системи

Проєктування структури системи - це той самий момент у розробці, який визначає, як будуть взаємодіяти між собою всі основні деталі LMS, щоб усе працювало разом без збоїв. Уже з цього етапу закладається підтримка різних ролей користувачів, робота з навчальними матеріалами, завданнями, тестами, аналітикою та нормальна взаємодія з базою даних. Загальну схему взаємодії компонентів наведено на рисунку 3.1.

Вся система побудована за трирівневою схемою: клієнтський рівень, серверна логіка й база даних. Клієнтський рівень - це інтерфейс, через який користувач працює з системою. На сервері відбувається обробка запитів, перевірка доступу та виконання бізнес-логіки. А база даних тримає всю інформацію про користувачів, курси, завдання, тести й результати.

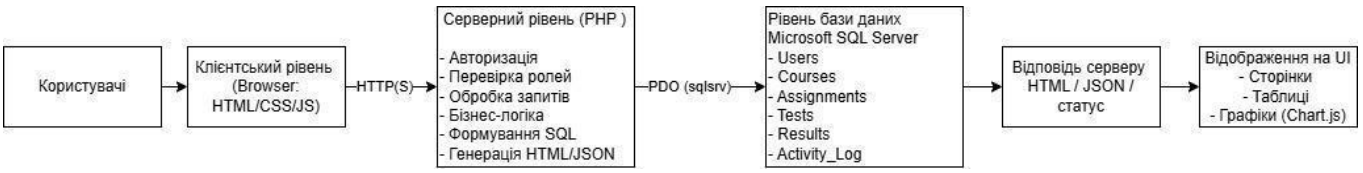


Рис. 3.1 Загальна схема взаємодії компонентів системи

Усі модулі підключені до однієї схеми взаємодії: користувач надсилає запит через інтерфейс, сервер реагує відповідно до його ролі та потреб модуля, далі йде робота з базою даних. Повертається все на клієнт у вигляді HTML-сторінок чи JSON-відповідей - їх уже можна використовувати для графіків чи інтерактивних деталей. Загальний алгоритм роботи системи наведено на рисунку 3.2.

Головні модулі системи - робота з користувачами, управління курсами, модуль завдань, модуль тестування, аналітика та дедлайни. Кожен має свої задачі,

але вся інформація лишається цілісною, бо всі обмінюються даними через одну спільну базу.

З такою архітектурою легко додавати щось нове - нові модулі чи фічі - і при цьому система стабільно працює навіть якщо користувачів стає більше. Модульна побудова суттєво спрощує обслуговування й оновлення: якщо треба щось змінити в одному з компонентів, це не зачіпає решту.

Рис. 3.2 Загальний алгоритм роботи системи

Діаграма варіантів використання показує, які ролі є в системі (адміністратор, викладач, студент) і як користувачі взаємодіють із основними функціями LMS. З її

допомогою можна побачити, що доступно кожній ролі та як вони працюють із модулями системи. Діаграму варіантів використання наведено на рисунку 3.3.

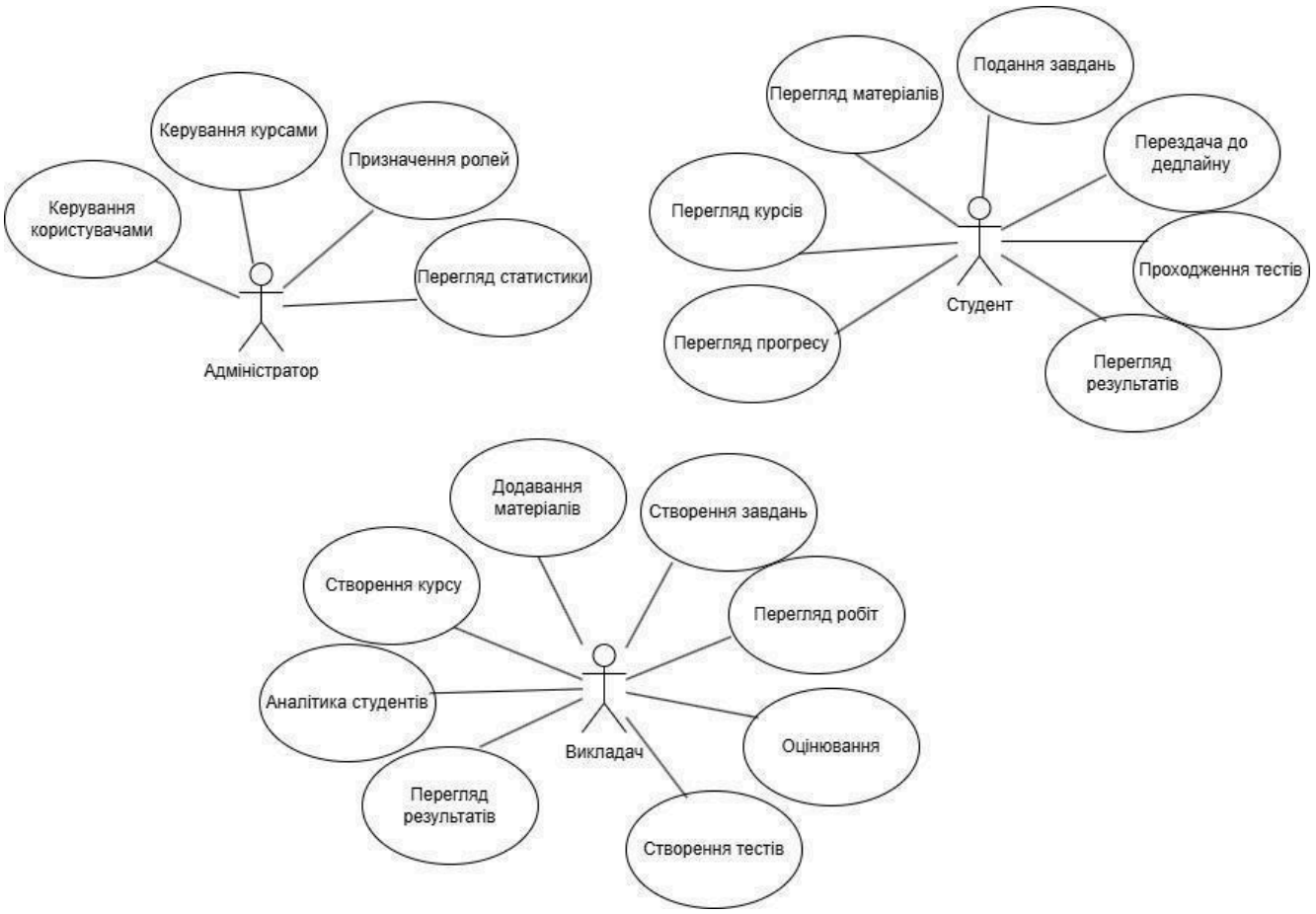


Рис. 3.3 Діаграма варіантів використання LMS

3.2 Реалізація клієнтської частини

Клієнтська частина системи відповідає за те, як виглядає інтерфейс і як користувачі взаємодіють з основними модулями LMS. Вона створена на основі HTML, CSS і JavaScript, що допомогло зробити зрозумілу структуру сторінок і додати потрібну інтерактивність. Тут головне - простота, логічна навігація і мінімум рухів, щоб виконувати базові дії. Дашборд користувача після авторизації наведено на рисунку 3.4.

Після входу користувач відразу потрапляє на дашборд із актуальною інформацією для його ролі. У студента відображаються останні оцінки, календар дедлайнів і можливість приєднатися до курсу за кодом. Викладач бачить графік

середньої успішності студентів і календар дедлайнів своїх завдань. Завдяки цьому користувачі отримують основне відразу, без зайвих переходів між сторінками.

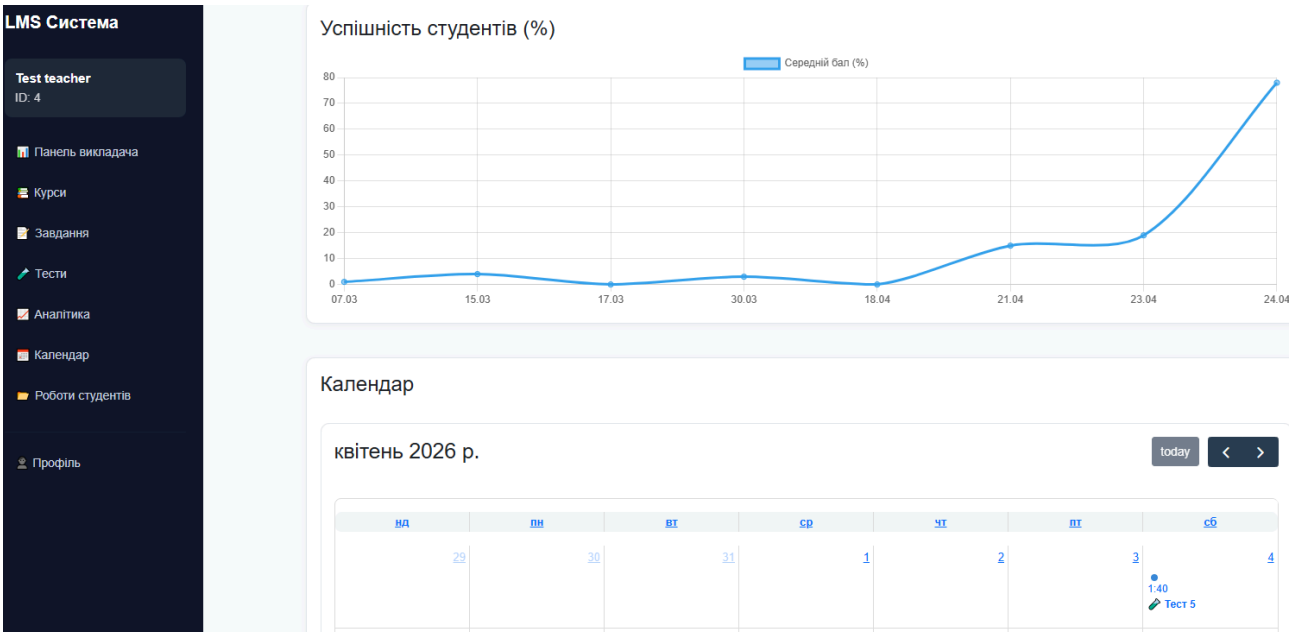


Рис. 3.4 Дашборд користувача після авторизації

На сторінці курсу є панель навігації, через яку можна перейти до матеріалів, завдань, тестів або аналітики. Кожен розділ має свою структуру і набір елементів. У модулі матеріалів можна переглядати файли, які викладач додав до курсу. Модуль завдань дає студентам опис роботи, дедлайн і можливість завантажити виконане завдання, а викладачам - переглядати подані роботи та виставляти оцінки. Інтерфейс сторінки курсу з панеллю навігації наведено на рисунку 3.5.

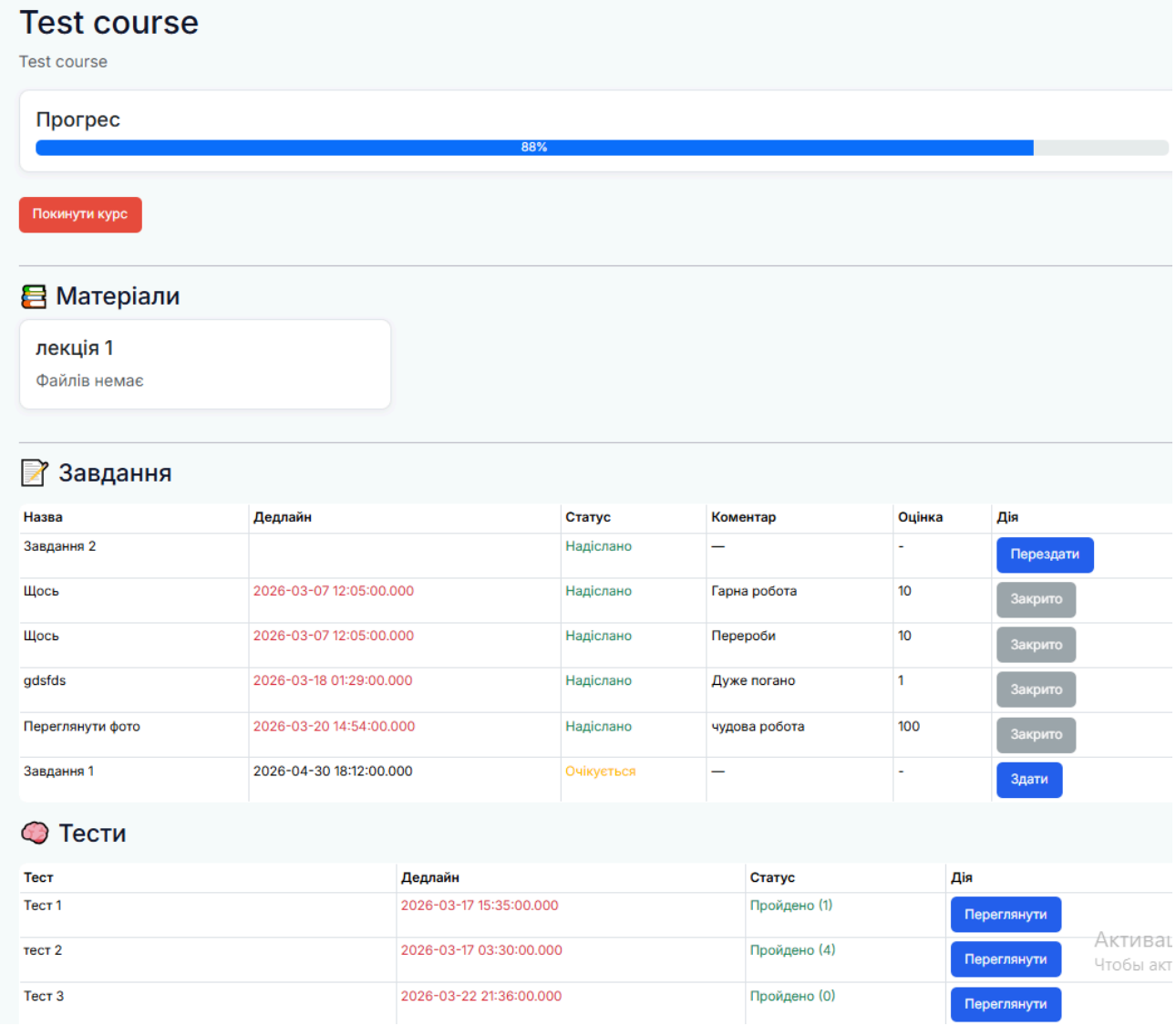


Рис. 3.5 Інтерфейс сторінки курсу з панеллю навігації

Модуль тестування має таймер, який рахує час до завершення тесту. Після здачі тесту студент бачить свій результат, а викладач має доступ до статистики результату тесту. Такий підхід допомагає уникнути помилок і полегшує роботу всім учасникам процесу. Інтерфейс проходження тесту наведено на рисунку 3.6.

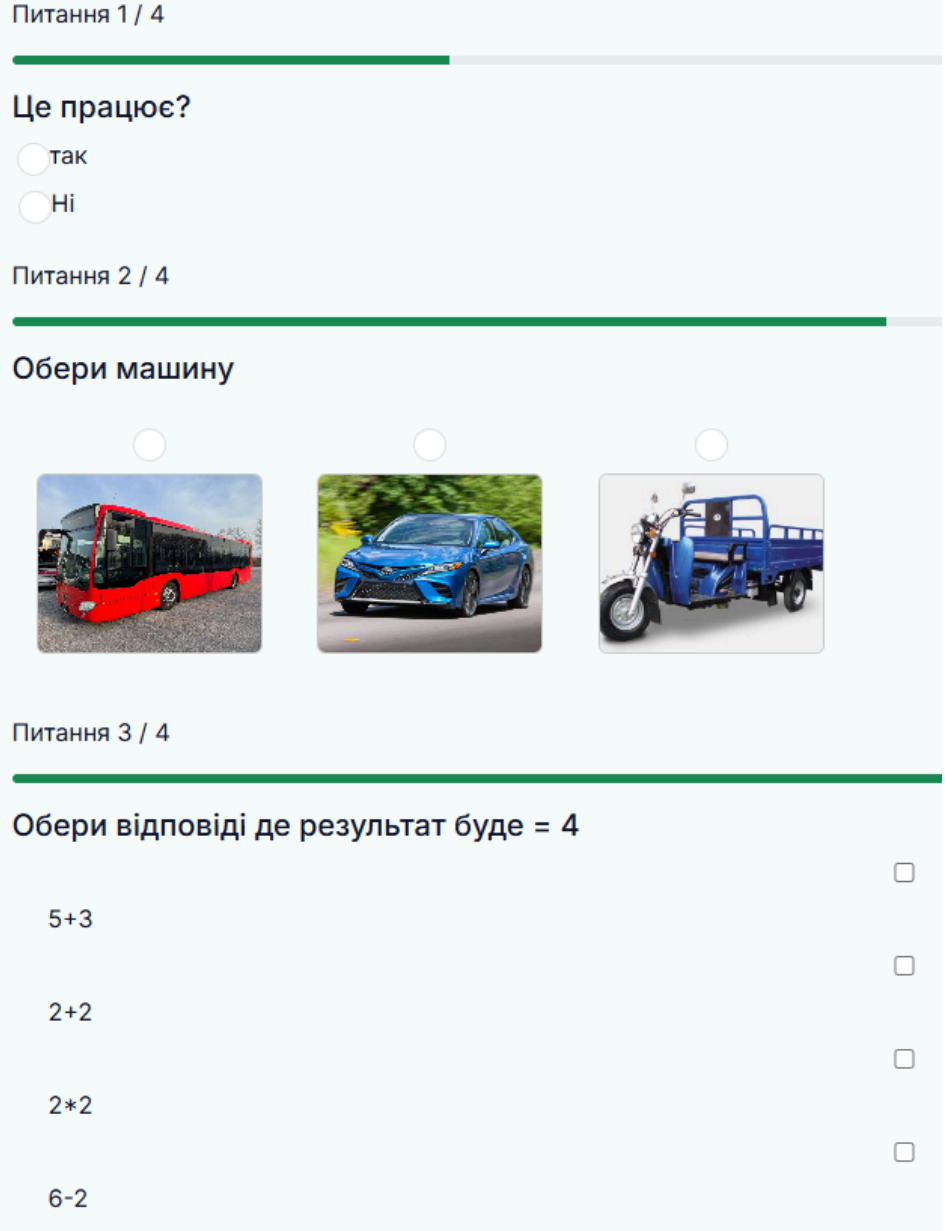


Рис. 3.6 Інтерфейс проходження тесту

В аналітиці можна переглядати статистику у вигляді графіків. Для їх створення використовується бібліотека Chart.js, тож інформація подається наочно, а оцінити зміну результатів можна буквально за кілька секунд. Студент бачить свої дані, викладач - статус всієї групи. Аналітичну панель наведено на рисунку 3.7.

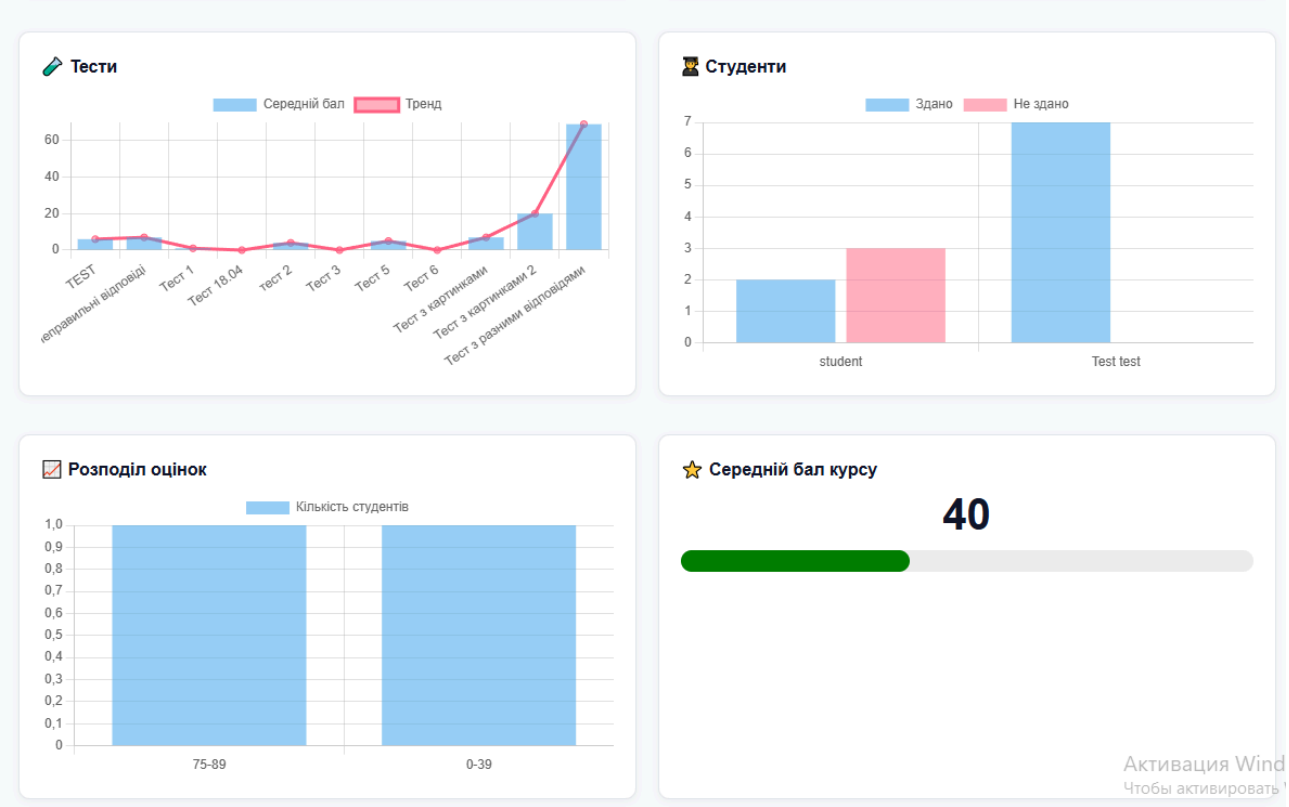


Рис. 3.7 Аналітична панель з графіками успішності

За стилізацію інтерфейсу відповідає CSS3. Основна ідея - простота розташування елементів, щоб було легко орієнтуватися й швидко знаходити все потрібне. JavaScript відповідає за інтерактивні деталі: таймери в тестах, календарі дедлайнів, побудову графіків.

У підсумку, клієнтська частина дає зручний доступ до можливостей LMS, підтримує основні модулі й забезпечує зрозумілий інтерфейс як для студентів, так і для викладачів. Вона є важливою складовою, оскільки визначає комфорт користувача та ефективність роботи з навчальними матеріали.

3.3 Реалізація серверної частини

Сервер LMS побудований на PHP і тягне на собі всю основну логіку системи. Тут проходить обробка користувацьких запитів, взаємодія з базою даних і розподіл доступу між різними ролями. Саме сюди "прилітають" всі рішення щодо прав доступу, створення SQL-запитів, обробки даних із форм, збереження

результатів тестів і завдань, а також підготовки інформації, яку потім отримує клієнт. Загальну обробку запиту на сервері наведено на рисунку 3.8.

Як тільки клієнт надсилає запит, сервер перевіряє, чи має користувач відповідні повноваження для цієї дії. Завдяки цьому студенти, викладачі та адміністратор отримують кожен свій рівень доступу. Наприклад, студент може лише переглядати матеріали й здавати завдання, а викладач - створювати й перевіряти їх, а також складати тести. Це не просто організація - завдяки такій схемі система чітко розмежовує функції й блокує всі спроби несанкціонованого доступу.

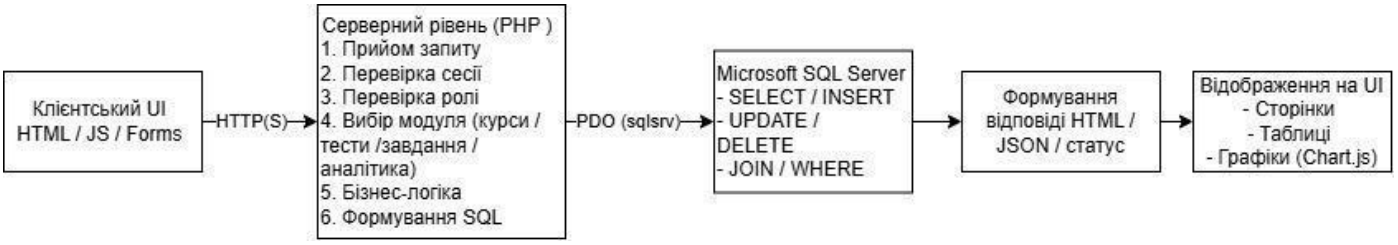


Рис. 3.8 Загальна схема обробку запиту на сервері

Для роботи з базою даних використовується драйвер sqlsrv. Він дає змогу формувати параметризовані запити та захиститися від SQL-ін’єкцій. Серверна логіка відповідає за всі запити, які стосуються курсів, матеріалів, завдань, тестів, результатів і статистики - а також додає, редагує чи видаляє ці дані. Усі операції тут йдуть через підготовлені інструкції, що суттєво підтягує безпеку та стабільність роботи LMS.

Завдання - окрема тема. Коли студент завантажує файл, сервер перевіряє його, зберігає у потрібній папці й додає запис у таблицю Assignment_Submissions. А коли викладач перевіряє роботу, сервер оновлює оцінку та коментар, формуючи повну історію виконання для кожного завдання. Логіку обробки подання завдання наведено на рисунку 3.9.

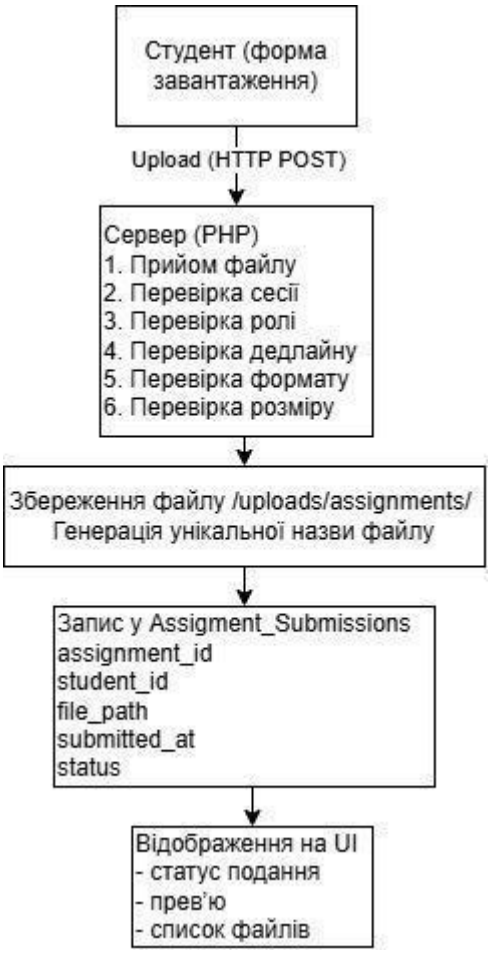


Рис. 3.9 Логіка обробки подання завдання студентом

Модуль тестування теж працює повністю через сервер. Сервер приймає відповіді на всі питання, записує їх у базу, а після завершення обраховує підсумковий результат. Всі підрахунки залишаються на стороні сервера - так користувачі не можуть їх підмінити чи щось "підкрутити".

Щодо аналітики: її теж підтримує серверна логіка. PHP-скрипти роблять вибірки з бази, рахують середні результати і кількість виконаних робіт, а потім передають дані на клієнт у JSON для будування графіків. Тобто вся робота з даними та їх обробка відбувається на сервері, а візуалізація - вже на клієнті.

Усе це зводиться до простої істини: сервер - це центральний вузол LMS. Він керує логікою, тягне на собі всю роботу з базою даних, розподіляє доступ і поєднує клієнтський інтерфейс із реальними даними. Від його роботи залежить, наскільки стабільно триматимуться всі модулі системи.

3.4 Реалізація модуля тестування

Модуль тестування займає важливе місце в системі - через нього перевіряються знання студентів у межах навчальних курсів. Весь процес охоплює кілька етапів: викладач створює тест, додаючи потрібні запитання й варіанти відповідей, студент проходить тест, а його відповіді зберігаються після завершення спроби, а результат підраховується автоматично. Обмін між клієнтською частиною, сервером і базою даних відбувається постійно. Інтерфейс проходження тесту наведено на рисунку 3.10.

Все влаштовано так, щоб модулем було легко користуватись і щоб підрахунок результатів не викликав помилок. Коли викладач формує тест, він просто додає запитання й відповіді, а система зберігає всю цю інформацію окремо в базі. Це дозволяє відтворити тест у точному вигляді пізніше.

Студент, працюючи з тестом, бачить усі запитання на одній сторінці, а для контролю часу використовується таймер. Якщо заданий час вичерпано, система автоматично завершує спробу і зберігає все, що студент встиг відповісти. Це допомагає уникати шахрайства й ставить всіх у рівні умови.

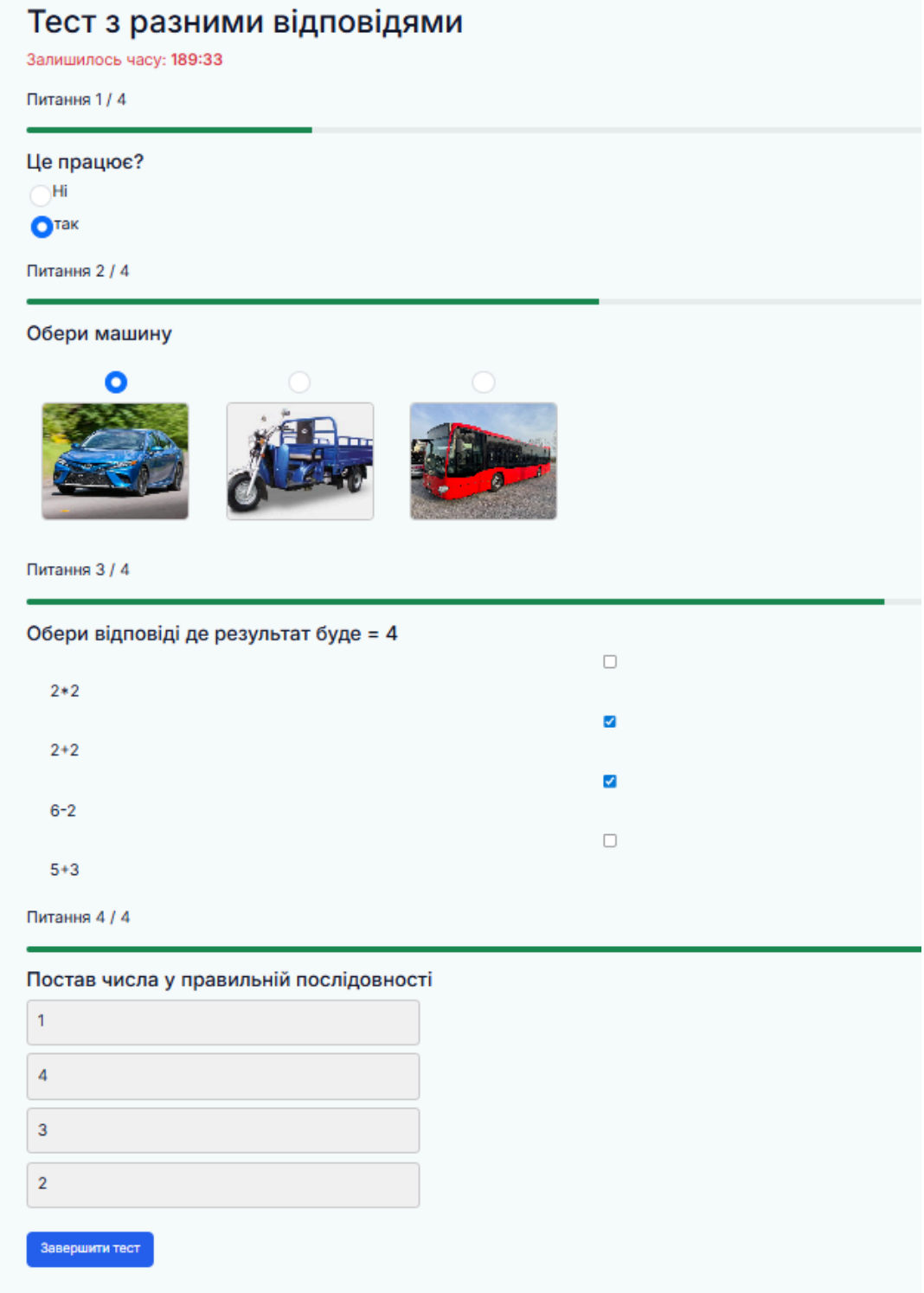


Рис. 3.10 Інтерфейс проходження тесту з таймером

Відповіді студента надходять на сервер і зберігаються у базі даних. Після завершення тесту сервер проводить перевірку: відповіді студента порівнюються із правильними варіантами, які зберігаються у відповідній таблиці. Схему обробки відповідей під час проходження тесту наведено на рисунку 3.11.

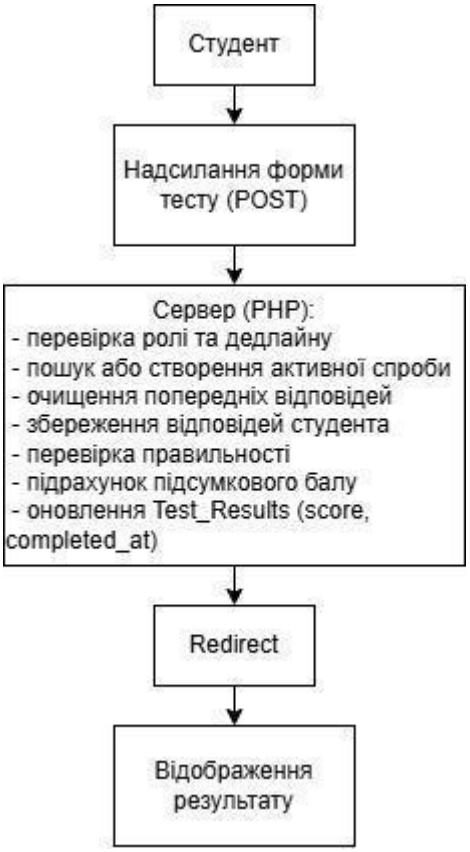


Рис. 3.11 Схема обробки відповідей під час проходження тесту

Як тільки система підрахувала бали, студент бачить свій підсумковий результат. Підсумки тесту також потрапляють до таблиці Test_Results - це дає змогу викладачу переглядати статистику й бачити успіхи студентів у розрізі курсів. А модуль аналітики вже на основі цих даних будує графіки з динамікою результатів всередині курсу.

Тестовий модуль інтегровано з усіма іншими частинами системи. Сервер перевіряє права доступу, тестова структура й результати зберігаються у базі даних, а клієнтська частина відповідає за зручний інтерфейс і комунікацію з користувачем. Завдяки цьому модуль працює стабільно, а всі результати обробляються коректно.

Фактично, тут покрито весь цикл роботи з тестами - від їх створення до показу підсумкових балів. Саме тому цей модуль такий важливий для LMS: він дозволяє чесно оцінювати успішність студентів і збирати об’єктивну статистику.

3.5 Реалізація модуля завдань

Модуль завдань займає ключове місце в системі: саме через нього студенти й викладачі взаємодіють під час виконання практичних робіт. Тут викладач створює завдання, студенти надсилають свої відповіді, а потім усе перевіряється й оцінюється. Весь цей процес працює через взаємодію клієнтської частини, серверної логіки та бази даних. Інтерфейс творення завдання викладачем наведено на рисунку 3.12.

Структуру модуля було зроблно таким чином, щоб усе працювало просто і швидко: для основних дій не треба зайвий раз клацати. Викладач указує назву завдання, короткий опис, дату здачі і якщо потрібно - додає файл. Уся ця інформація потрапляє у таблицю Assignments, а студенти потім бачать свої завдання в межах курсу.

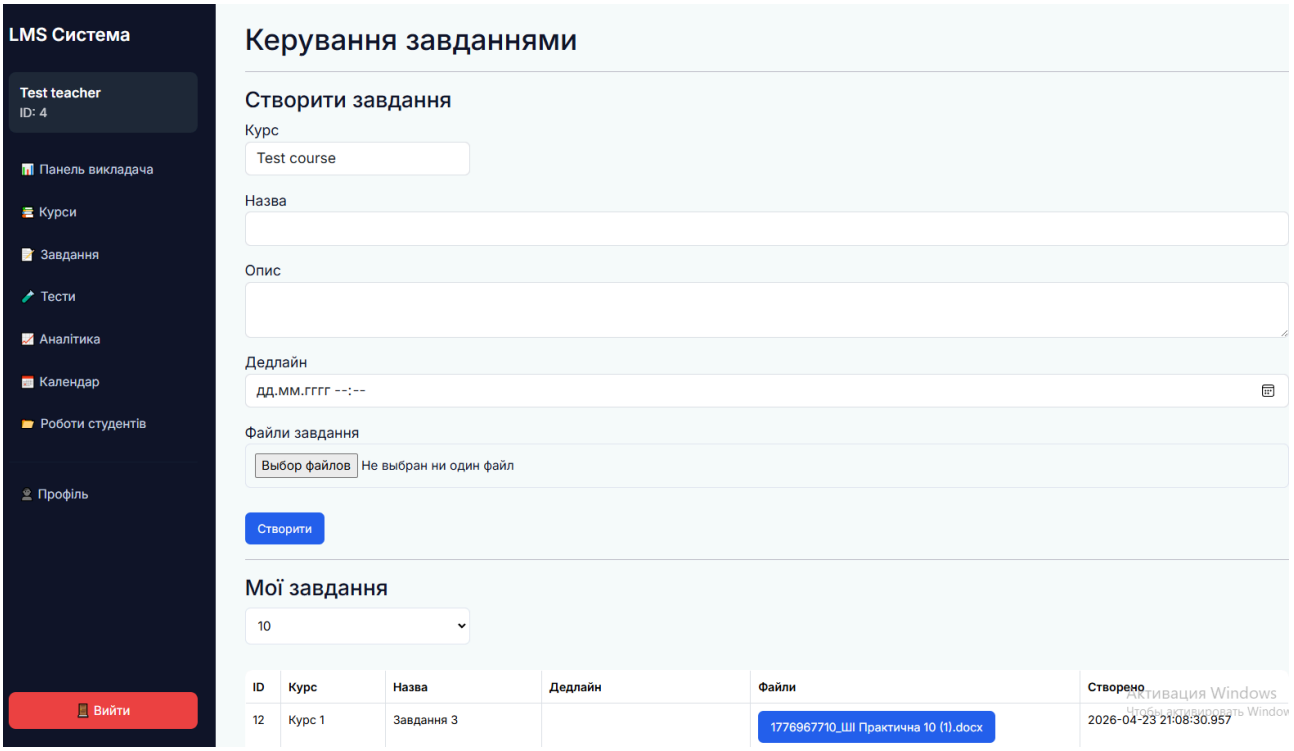


Рис. 3.12 Інтерфейс творення завдання викладачем

Студент бачить список завдань на сторінці курсу. В описі кожного відображено дедлайн і є кнопка для завантаження готової роботи. Коли студент подає файл, сервер перевіряє правильність даних, зберігає документ у потрібному каталозі й створює запис у таблиці Assignment_Submissions. Якщо термін подачі ще не минув, студент може перездати роботу, а система зберігає нову версію файлу як окремий запис. Інтерфейс подання завдання студентом наведено на рисунку 3.13.

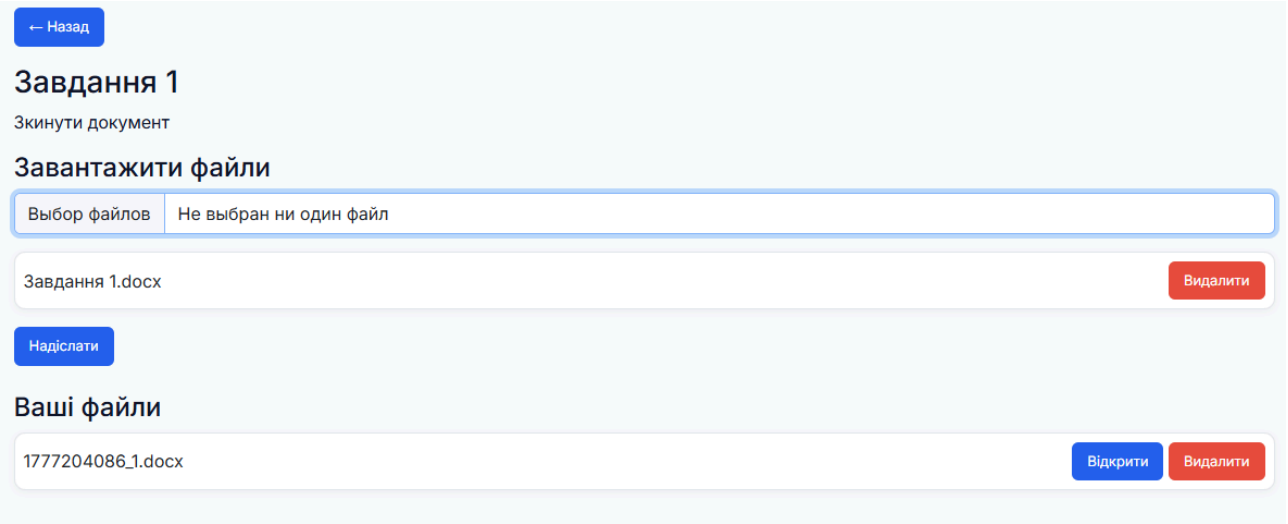


Рис. 3.13 Інтерфейс подання завдання студентом

Після відправки роботи викладач відкриває список отриманих файлів. Для кожного студента видно дату подачі, завантажений файл і поле для оцінки. Після перевірки можна залишити коментар і зафіксувати оцінку у таблиці Assigment_Submissions. Так фіксується факт подання роботи та її оцінювання. Інтерфейс перевірки поданих робіт наведено на рисунку 3.14.

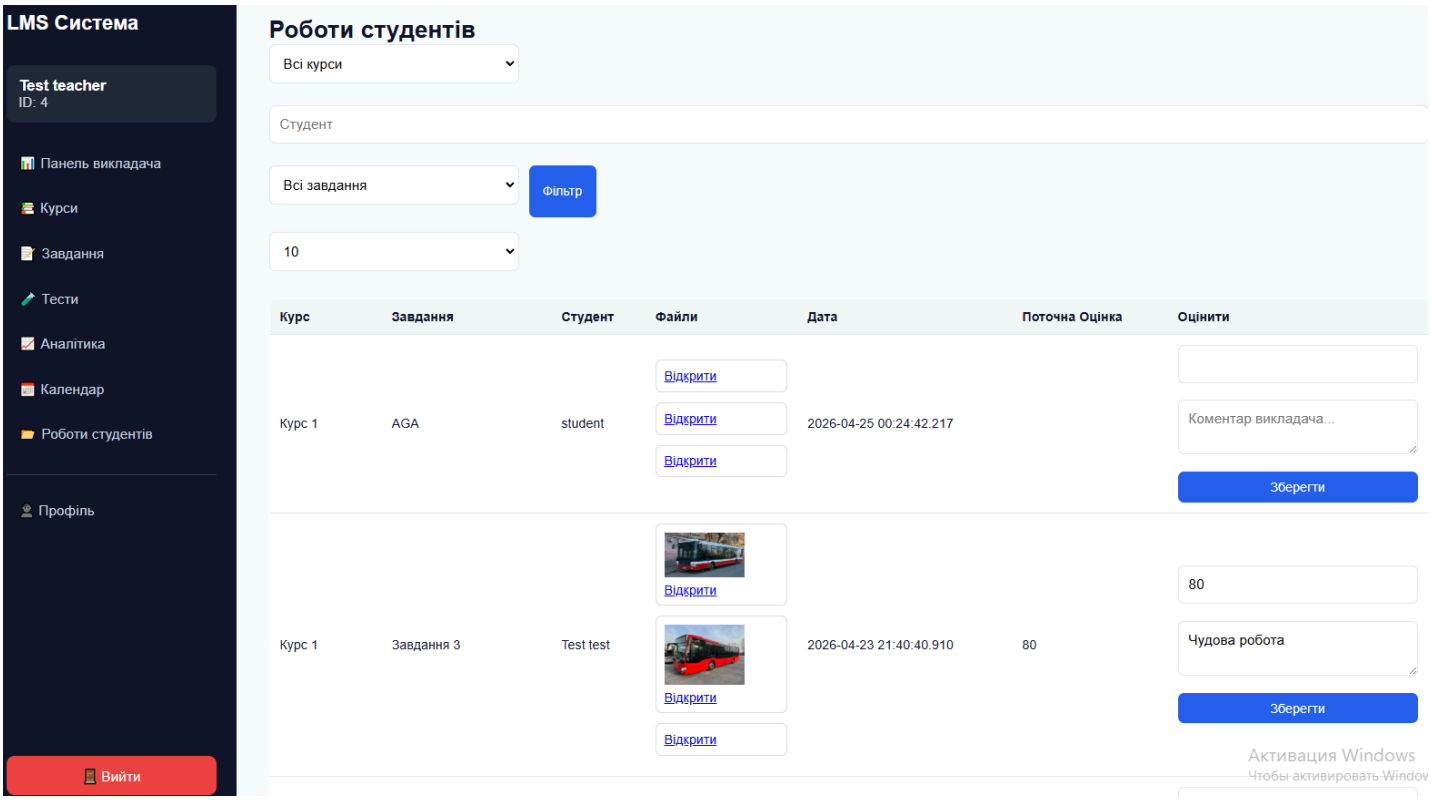


Рис. 3.14 Інтерфейс перевірки поданих робіт

Модуль завдань тісно пов’язаний з іншими частинами системи. Оцінки використовують у модулі аналітики для побудови графіків успішності, а дані про дедлайни автоматично потрапляють у календар дашборду - так студенти й викладачі не пропустять важливих дат.

Серверна частина контролює доступи, працює з файлами, записує результати та готує відповіді для клієнтської частини. Такий підхід дозволяє тримати дані в порядку й гарантує стабільну роботу модуля у всій системі.

Модуль завдань охоплює основні процеси роботи з практикою: від створення завдання до оцінювання кожної роботи. Це невід’ємна частина LMS, що допомагає організувати навчання просто й зрозуміло як для студентів, так і для викладачів.

3.6 Реалізація модуля аналітики

Модуль аналітики - одна з ключових частин системи. Він допомагає побачити реальну картину навчальних результатів студентів і дає викладачам можливість швидко оцінити динаміку успішності групи. Вся робота модуля ґрунтується на взаємодії серверної логіки, яка готує статистику, і клієнтської частини, котра відповідає за її відображення. Аналітична панель студента наведена на рисунку 3.15.

Аналітика доступна як студентам, так і викладачам, але кожен бачить свою інформацію. Студенти отримують лише свої дані - оцінки за завдання, результати тестів, зміну балів з часом. Для викладача відкривається загальна статистика по групі, яка дає уявлення про те, як засвоюється матеріал, і дозволяє помічати проблеми на ранньому етапі.

Графіки створюються за допомогою бібліотеки Chart.js [13]: дані подаються в лінійних або стовпчикових графіках, можна використовувати й інші типи. Весь масив інформації формує сервер у форматі JSON й уже готовий передається на клієнт, де відразу виводиться.

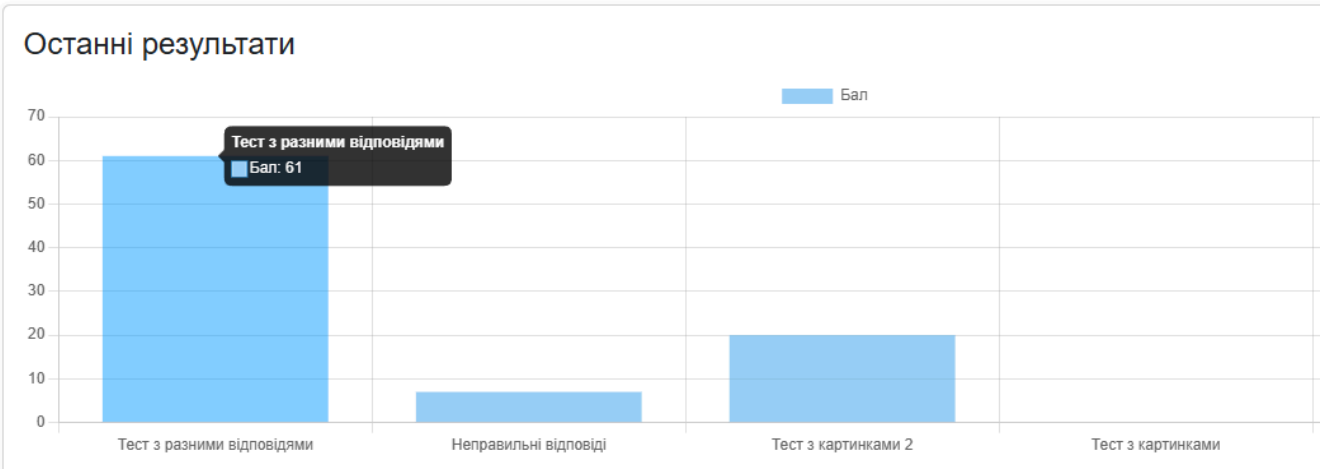


Рис. 3.15 Аналітична панель студента

У студентському розділі видно графік останніх оцінок - цього достатньо, щоб швидко зрозуміти свої успіхи та зміни в динаміці. Тут враховані й результати тестів, і оцінки за завдання, отже усе виглядає комплексно. Після появи нової оцінки графік оновлюється самостійно.

Викладачам доступний ще один вигляд - середні оцінки по студентах за кожен день. Це допомагає відстежувати загальну активність групи, сміливо помічати, коли спостерігається спад або навпаки, підйом. Усі ці дані тягнуться з таблиць Submissions та Results, що забезпечує точність підрахунків. Аналітичну панель викладача наведено на рисунку 3.16.

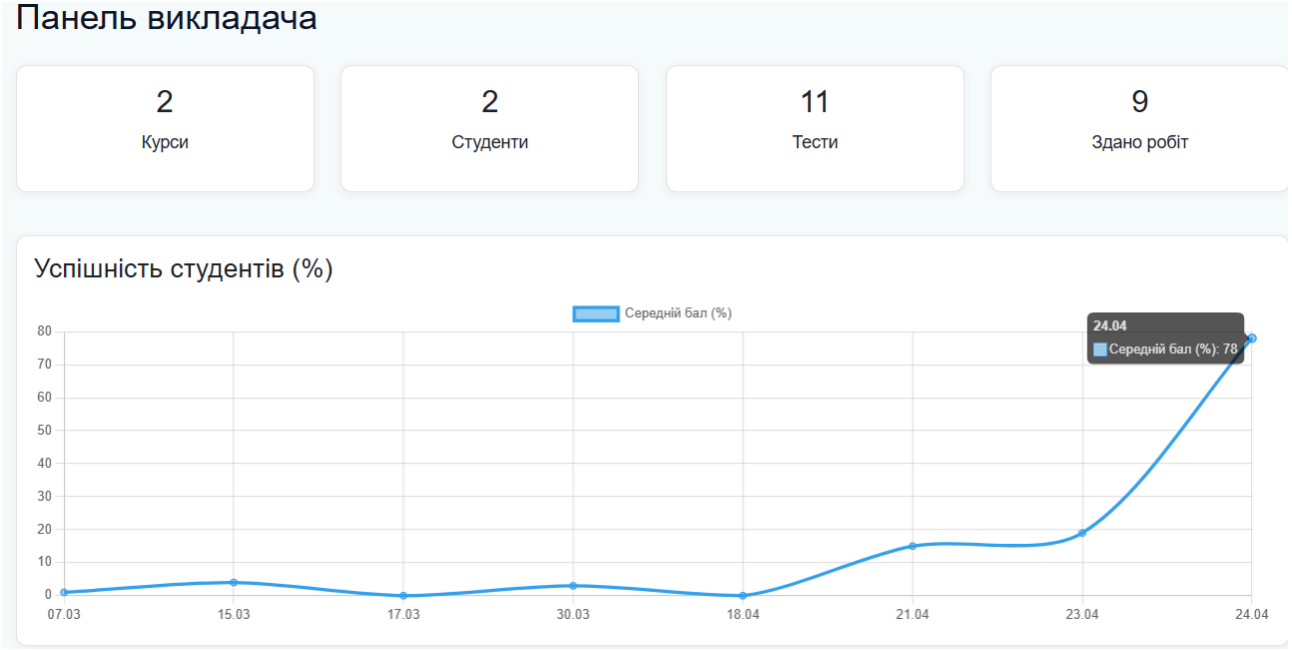


Рис. 3.16 Аналітична панель викладача з графіком середньої успішності

Серверна частина аналізує базу, підраховує середні значення, кількість завдань і тестів, та інші деталі. Потім усе це структуровано відправляється на клієнт. Такий поділ дає змогу утримати стабільну роботу навіть при великій кількості даних.

Аналітичний модуль пов’язаний з дашбордом - студент і викладач бачать важливі показники одразу після входу в систему. Це дозволяє швидко зорієнтуватися і краще планувати навчання.

Зрештою, аналітичний модуль відповідає за зрозумілу візуалізацію статистики, підтримує прийняття рішень у навчальному процесі та без перебільшення є незамінною частиною LMS. Для студентів це можливість оцінити свої результати, для викладачів - контролювати динаміку в групі і вчасно впливати на ситуацію.

3.7 Реалізація роботи з базою даних

База даних - це по суті, серце всієї системи: кожен модуль LMS, чи то робота з користувачами, курсами, матеріалами, завданнями, тестами або аналітикою, працює з даними, які зберігаються в Microsoft SQL Server. Серверна частина спілкується з базою через драйвер sqlsrv [11]. Він дозволяє використовувати параметризовані SQL-запити, що не тільки полегшує роботу із великими обсягами інформації, а й робить її безпечнішою. Схему взаємодії PHP-логіки з таблицями Assignments та Assignment_Submissions наведено на рисунку 3.17.

Базу даних побудовано на реляційній моделі: тут багато пов’язаних між собою таблиць. Головна - Users, саме в ній міститься інформація про студентів, викладачів та адміністратора. Courses - це опис навчальних курсів, а Enrollments - зв’язок між користувачами та курсами. Для матеріалів, завдань, тестів та результатів передбачено окремі таблиці, завдяки чому повний навчальний процес легко відстежується й підтримується.

Обмін із базою відбувається через підготовлені SQL-запити. Це дає змогу захистити систему від SQL-ін’єкцій та гарантує коректність обробки даних. Наприклад, коли користувач авторизується, сервер перевіряє, чи існує такий користувач, і визначає його роль. Якщо мова про курси - система отримує перелік доступних курсів, а також підтягує матеріали, завдання та тести, пов’язані з кожним із них.

Коли студент здає завдання, сервер створює новий запис у таблиці Assignment_Submissions - там зберігається шлях до файлу, дата подачі та ID користувача. Якщо викладач перевіряє цю роботу, у запис додають оцінку та

коментар. В результаті зберігається вся історія виконання і стає видно, як і коли оцінювали завдання.

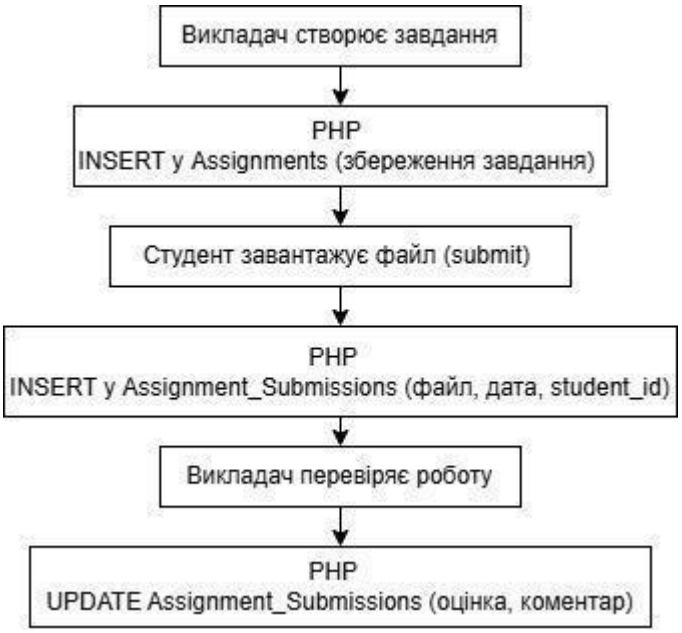


Рис. 3.17 Схема взаємодії PHP-логіки з таблицями Assignments та Assignment_Submissions

Тестування теж активно використовує дані. Таблиця Tests містить загальну інформацію про тест, Questions - перелік запитань, Results - підсумкові бали студентів. Відповіді зберігаються в таблиці Student_Answers - що дозволяє переглянути вибрані варіанти результат спроби без плутанини.

У модулі аналітики дані з Assignment_Submissions і Results допомагають будувати статистику. Сервер вибирає потрібну інформацію, рахує середні значення і формує структуру, яку клієнтська частина отримує у форматі JSON - її вже використовують для побудови графіків.

Окрема таблиця Activity_Log зберігає інформацію про вихід студента з курсу. Це бачить адміністратор, тож він завжди може проконтролювати активність учасників у рамках дисципліни.

Такий підхід до роботи з базою даних тримає всю інформацію цілісною, забезпечує тісну взаємодію між модулями й дає системі стабільність. Використання Microsoft SQL Server із параметризованими запитами зробило

надійну структуру даних та можливість подальшого масштабування системи, що є важливим для LMS і дозволяє без проблем обробляти дані на кожному етапі навчання.

3.8 Тестування системи

Тестування - це фінальний етап розробки, на якому перевіряється, чи правильно працюють усі модулі LMS, чи система стабільна, і чи відповідає логіка тому, що було задумано у проєкті. Під час тестування було перевірено роботу клієнтської й серверної частини, взаємодію з базою даних і наскільки добре виконується основний сценарій для користувачів. Результат авторизації користувача наведено на рисунку 3.18.

Модулі тестувалися поступово в процесі розробки, тому більшість багів вдалося зловити та виправити ще на ранніх стадіях. Особлива увага приділялася перевірці роботи з базою даних, адже вся система тримається на базі даних - тут зберігається інформація про користувачів, курси, завдання, тести та результати.

Для модуля авторизації було перевірено, як відбувається вхід для різних типів користувачів, що робиться у випадку неправильних даних і чи відбувається перенаправлення на потрібні сторінки після входу. Всі ці пункти підтвердили: система правильно розділяє доступ до своїх модулів.

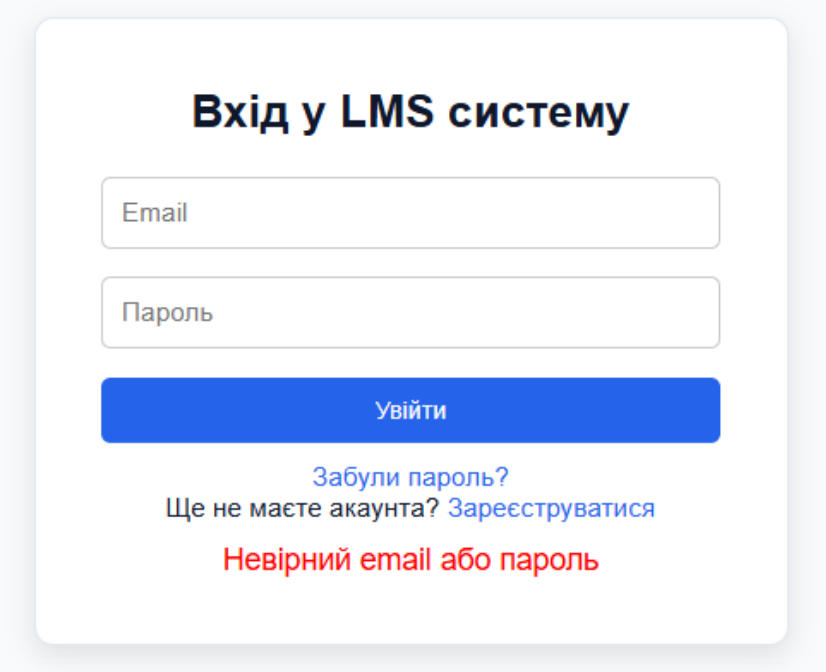


Рис. 3.18 Результат авторизації користувача

Модуль курсів було протестовано на те, чи коректно відображається список, чи може студент приєднатися до курсу і чи працює доступ до матеріалів, завдань і тестів. Виявилося, що зв’язки між користувачами та курсами фіксуються, і чужий курс просто так не переглянеш - доступ є лише до своїх.

У блоці завдань було протестовано створення завдань викладачем, відправлення відповідей студентом, повторне завантаження файлів до дедлайну і виставлення оцінок. Окремо було перевірено, наскільки надійно фіксуються файли та записи у таблиці Assignment_Submissions. Усі дії працювали як задумано. Результат тестування подання завдання студентом наведено на рисунку 3.19.

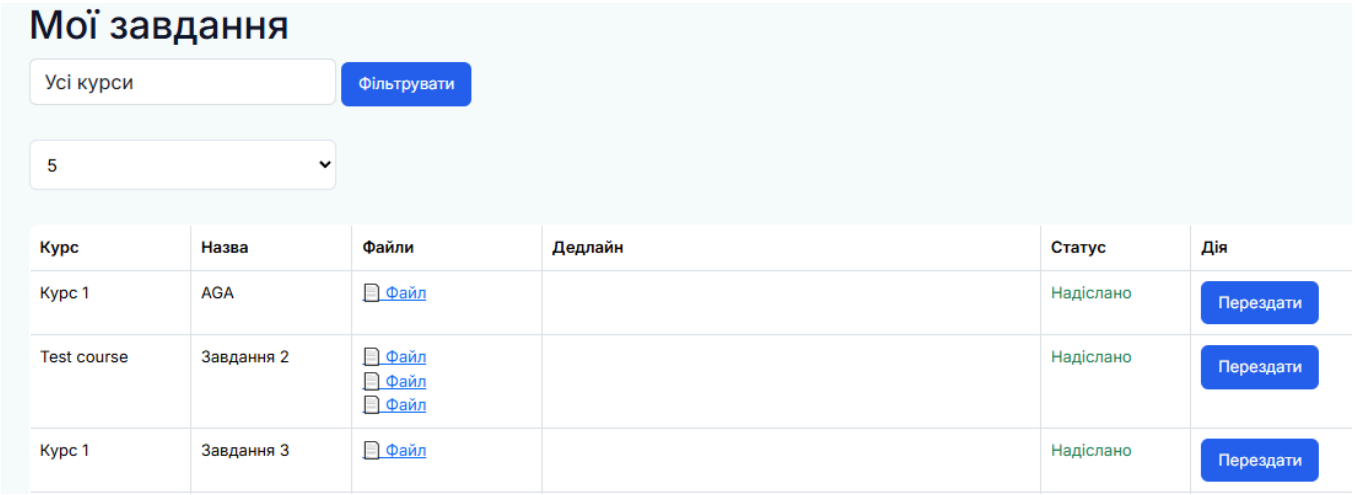


Рис. 3.19 Результат тестування подання завдання студентом

Модуль тестування перевірявся по кількох пунктах: як відображаються запитання, як працює таймер, чи зберігаються відповіді автоматично і як формується фінальний результат. Було розглянуто як ручне завершення тесту, так і автоматичне після завершення часу. Всі відповіді безпомилково зберігалися у базі, а підсумок формувався згідно з відповідями. Результат тестування проходження тесту наведено на рисунку 3.20.

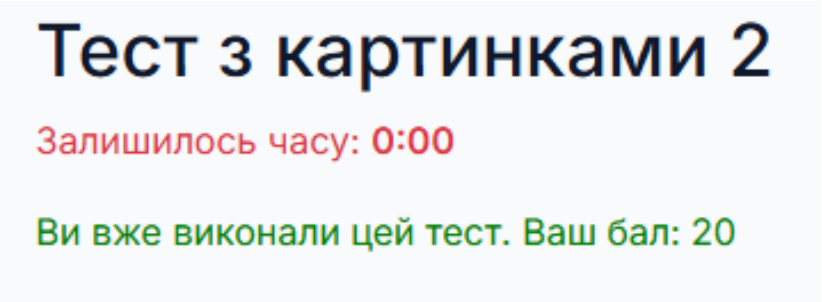


Рис. 3.20 Результат тестування проходження тесту

В модуль аналітики було перевірено, чи коректно формуються статистичні дані та чи будуються графіки. Було встановлено, що дані передаються у JSON, і графіки оновлюються після нових оцінок. Все це показало, що клієнт взаємодіє із сервером стабільно.

Окремо було протестовано роботу з базою даних: як виконуються параметризовані запити, як обробляються помилки, наскільки правильно

працюють зв’язки між таблицями й чи не збивається система при великій кількості операцій. Основні сценарії проходили без збоїв.

Окрім стандартних перевірок, були знайдені й дрібні неточності, які ще треба підправити. Наприклад, у модулі онлайн-тестування таймер додавав до заданої тривалості лишніх 180 хвилин, через що механізм відліку працював неправильно. Також під час перевірки завдань на впорядкування інтерфейс міг позначати правильну відповідь як неправильну - хоча бал розраховувався і зберігався вірно. Обидві ці проблеми стосуються клієнтської частини й можуть бути вирішені у майбутніх оновленнях, не чіпаючи базову логіку роботи.

Загалом тестування показало: всі модулі працюють справно, клієнт і сервер узгоджено спілкуються, а логіка відповідає вимогам проєкту. Система готова до використання та подальшого розвитку.

ВИСНОВКИ

У кваліфікаційній роботі було реалізовано веб-орієнтовану інформаційну систему для керування навчальними курсами. Вона об’єднує онлайн-тестування, контроль дедлайнів, роботу з файлами та інтерактивну аналітику прогресу студентів. Дослідження дозволило краще зрозуміти, що сучасна LMS має бути не просто місцем для зберігання курсів, а інструментом із функціями, без яких цифрову освіту сьогодні вже не уявити.

Було розглянуто сучасні системи управління навчанням, зокрема Moodle, Google Classroom і Prometheus. Порівняння їхнього функціоналу дало змогу побачити, які можливості реалізовано вдало, а де є певні обмеження. Це допомогло сформувати перелік вимог до власної системи та визначити, на яких елементах варто зосередити увагу - насамперед на аналітиці, роботі з файлами та контролю дедлайнів.

На основі попереднього аналізу були визначені вимоги та створено загальну архітектуру веб-орієнтованої системи. Було побудовано логічну модель бази даних, окреслено основні сутності та зв’язки між ними. Також продумано взаємодію між клієнтською частиною, сервером і сховищем даних, щоб забезпечити узгоджену роботу всіх елементів і можливість подальшого розвитку системи.

У процесі реалізації створено основні модулі: управління курсами, роботу з файлами, тестування, механізм дедлайнів та аналітика успішності студентів. Інтерфейс орієнтований на зручність користувачів, а серверна частина на PHP відповідає за обробку запитів, перевірку доступу та взаємодію з Microsoft SQL Server. Реалізовано збереження відповідей у тестах, можливість повторного завантаження файлів до дедлайну та побудову графіків успішності за допомогою Chart.js.

Після завершення програмної частини було проведено тестування. Перевірено роботу кожного модуля, стабільність виконання дій, коректність

взаємодії з базою даних і відповідність функціоналу заданим вимогам. Результати підтвердили, що система працює стабільно, правильно обробляє дані та формує статистику без збоїв.

У підсумку було реалізовано веб-систему, що поєднує управління курсами, тестування, роботу з файлами та аналітику. Система пройшла повний цикл розробки від аналізу та проєктування до програмної реалізації й тестування. Перспективи подальшого розвитку системи полягають у розширені функціоналу шляхом додавання нових типів тестових завдань, удосконалення засобів аналітики та розробки мобільного застосунку, що сприятиме підвищенню зручності, гнучкості та ефективності використання системи в освітньому процесі.

ПЕРЕЛІК ПОСИЛАНЬ

1. РОЗРОБКА ТА ВИКОРИСТАННЯ СИСТЕМИ ДИСТАНЦІЙНОГО НАВЧАННЯ «ELEARNING» У ЗАКЛАДАХ ВИЩОЇ ОСВІТИ / І. І. Сугоняк та ін. *Information Technologies and Learning Tools*. 2022. Т. 87, № 1. С. 288–305. URL: https://doi.org/10.33407/itlt.v87i1.4231 (дата звернення: 27.04.2026).
2. Moodle Documentation. Moodle Project. URL: https://docs.moodle.org/ (дата звернення: 27.04.2026).
3. Classroom Help. *Google Help*. URL: https://support.google.com/edu/classroom/ (дата звернення: 27.04.2026).
4. Prometheus - Будуй кар'єру курс за курсом. *Prometheus*. URL: https://prometheus.org.ua/ (дата звернення: 27.04.2026).
5. Кривонос О. СИСТЕМИ УПРАВЛІННЯ НАВЧАННЯМ У ЦИФРОВІЙ ОСВІТІ: ПОРІВНЯЛЬНИЙ АНАЛІЗ ПЛАТФОРМ І ПЕДАГОГІЧНІ ПЕРСПЕКТИВИ . *Суспільство та національні інтереси*. 2025. № 9(17). URL: https://eprints.zu.edu.ua/45179/ (дата звернення: 27.04.2026).
6. USING LCMS MOODLE FOR ORGANIZING DISTANCE AND MIXED LEARNING / O. Halytskyi та ін. *Academic Notes Series Pedagogical Science*. 2023. Т. 1, №210. URL: https://pednauk.cusu.edu.ua/index.php/pednauk/article/view/1595 (дата звернення: 27.04.2026).
7. PHP: MS SQL Server PDO Driver - Manual. *PHP*. URL: https://www.php.net/manual/en/ref.pdo-sqlsrv.php (дата звернення: 27.04.2026).
8. HTML Living Standard. W3C. URL: https://html.spec.whatwg.org/ (дата звернення: 27.04.2026).
9. JavaScript Guide - JavaScript | MDN. *MDN Web Docs*. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide (дата звернення: 27.04.2026).

10. Microsoft SQL Documentation - SQL Server. *Microsoft Learn: Build with answers in reach*. URL: https://learn.microsoft.com/sql/ (дата звернення: 27.04.2026).
11. Microsoft Drivers for PHP for SQL Server - PHP drivers for SQL Server. *Microsoft Learn: Build with answers in reach*. URL: https://learn.microsoft.com/sql/connect/php/ (дата звернення: 27.04.2026).
12. Школа програмування. Уроки PHP для початківців / #1 - Введення в PHP з нуля. Що таке PHP та як з ним працювати?, 2024. *YouTube*. URL: https://www.youtube.com/watch?v=HJZbJMFxqGg (дата звернення: 27.04.2026).
13. Chart.js | Chart.js. *Chart.js | Open source HTML5 Charts for your website*. URL: https://www.chartjs.org/docs/latest/ (дата звернення: 27.04.2026).
14. Школа програмування. Уроки CSS3 з нуля / #1 - Що таке CSS? Уроки для початківців українською, 2023. *YouTube*. URL: https://www.youtube.com/watch?v=mz0AVh0VQ74 (дата звернення: 27.04.2026).
15. Школа програмування. Курс HTML з нуля / #1 - Що таке HTML5? Уроки для початківців українською, 2023. *YouTube*. URL: https://www.youtube.com/watch?v=ZY6uf5985PI (дата звернення: 27.04.2026).
16. Школа програмування. Уроки JavaScript з нуля / #1 – Основи мови JS для початківців. Що до чого?, 2023. *YouTube*. URL: https://www.youtube.com/watch?v=RS4LF577fj4 (дата звернення: 27.04.2026).

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ТЕХНОЛОГІЙ ЦИФРОВОГО РОЗВИТКУ

Інформаційна система управління навчальними курсами з онлайн-тестуванням, контролем дедлайнів та інтерактивною аналітикою прогресу на основі PHP, HTML, CSS та JavaScript

Виконав студент 4 курсу
групи ТЦР- 44
Єфімов Максим Олександрович
Керівник роботи
К.т.н. Аронов Андрій Олексійович

Київ – 2026

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** – удосконалення процесу навчання шляхом створення веб-системи з підтримкою управління навчальними курсами, онлайн-тестування, контролю дедлайнів та роботи з файлами.
- **Об'єкт дослідження** – організація навчання у веб-орієнтованих інформаційних системах.
- **Предмет дослідження** – підходи, моделі та програмні рішення для розробки LMS із підтримкою тестування, дедлайнів, завдань і аналітики результатів.

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Аналіз LMS та формування вимог – дослідити існуючі платформи та визначити функціональні й нефункціональні вимоги до системи.
2. Проєктування архітектури та бази даних – створити архітектуру веб-системи та логічну модель БД для курсів, завдань, тестів і аналітики.
3. Розроблення клієнтської й серверної частин – реалізувати інтерфейс користувача та серверну логіку на PHP з підтримкою ролей і доступів.
4. Реалізація основних модулів – створити модулі курсів, завдань, онлайн-тестування з таймером, контролю дедлайнів та аналітики прогресу.
5. Тестування системи – перевірити коректність роботи всіх модулів, взаємодію клієнта й сервера та відповідність вимогам.

АНАЛІЗ АНАЛОГІВ

Критерій	Moodle	Google Classroom	Prometheus	Розроблена система
Простота інтерфейсу	– складний	+ дуже простий	+ простий	+ простий
Гнучкість тестування	+ багато типів питань, таймер	– обмежено	– базово	+ таймер, типи питань
Дедлайни	– негнучкі	+ гнучкі	– базові	+ гнучкі
Передача файлів	– залежить від налаштувань	+ можна	– немає	+ можна до дедлайнів
Робота з файлами	– базово	+ Drive	– залежить від курсу	+ збереження файлу після виходу з курсу
Аналітика	– потребує плагінів	– мінімальна	– мінімальна	+ базова інтерактивна аналітика
Розширюваність	+ плагіни	– обмежена	– обмежена	+ модульна
Поріг входу	– високий	+ низький	+ низький	+ низький

ВИМОГИ ДО СИСТЕМИ

Функціональні вимоги:

1. Ролі та доступ – створення акаунтів, авторизація та розмежування прав для студентів, викладачів і адміністратора.
2. Курси викладача – створення курсів, додавання матеріалів, редагування структури та перегляд списку студентів.
3. Матеріали курсу – завантаження й перегляд файлів, документів та зображень у межах курсу.
4. Завдання студентів – подання робіт, перегляд власних файлів і можливість оновлення до дедлайну.
5. Перевірка робіт – оцінювання, коментарі та доступ викладача до студентських файлів навіть після виходу студента з курсу.
6. Дедлайни – відображення строків виконання завдань і тестів у календарі або списку.
7. Тестування – створення тестів різних типів, таймер, збереження відповідей і автоматичне формування результатів.
8. Аналітика – перегляд результатів тестів і завдань, а також загальних показників успішності студентів.
9. Адміністрування – керування користувачами та перегляд подій, пов'язаних із їхньою активністю в курсах.

Нефункціональні вимоги:

1. Безпека доступу – авторизація за логіном і паролем, 3 ролі (студент, викладач, адміністратор), заборона доступу до чужих даних.
2. Надійність даних – усі курси, завдання, тести, результати та файли зберігаються в SQL Server без втрати даних при стандартних операціях.
3. Продуктивність – основні сторінки завантажуються за 2-3 секунди при одночасній роботі до 20-50 користувачів.
4. Кросбраузерність – коректна робота в актуальних версіях Chrome, Firefox та Edge.
5. Модульність – архітектура дозволяє додавати нові модулі (курси, завдання, тести, аналітика) без зміни існуючої логіки та структури БД.
6. Юзабіліті – основні дії (перегляд курсу, здача завдання, тестування) виконуються за 3-4 кліки.

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Back-end

серверна логіка

база даних

Front-end

стилізація

структура інтерфейсу

Інструменти розробки

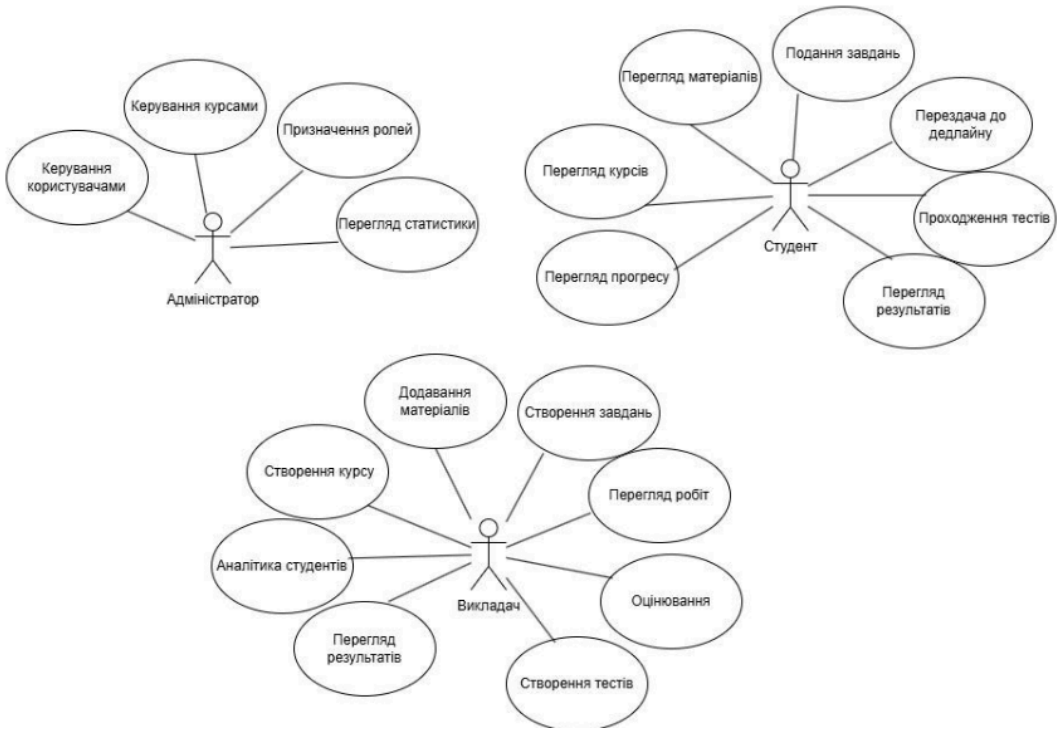
візуалізація статистики

динамічна взаємодія

локальний веб-сервер

редактор коду

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



ДІАГРАМА КОМПОНЕНТІВ

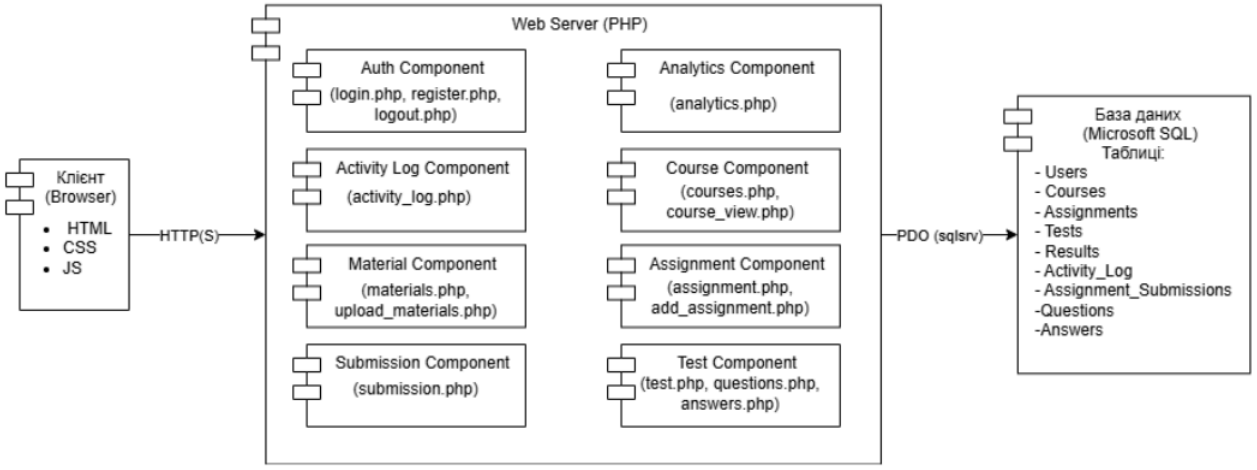


СХЕМА СТРУКТУРИ БАЗИ ДАНИХ

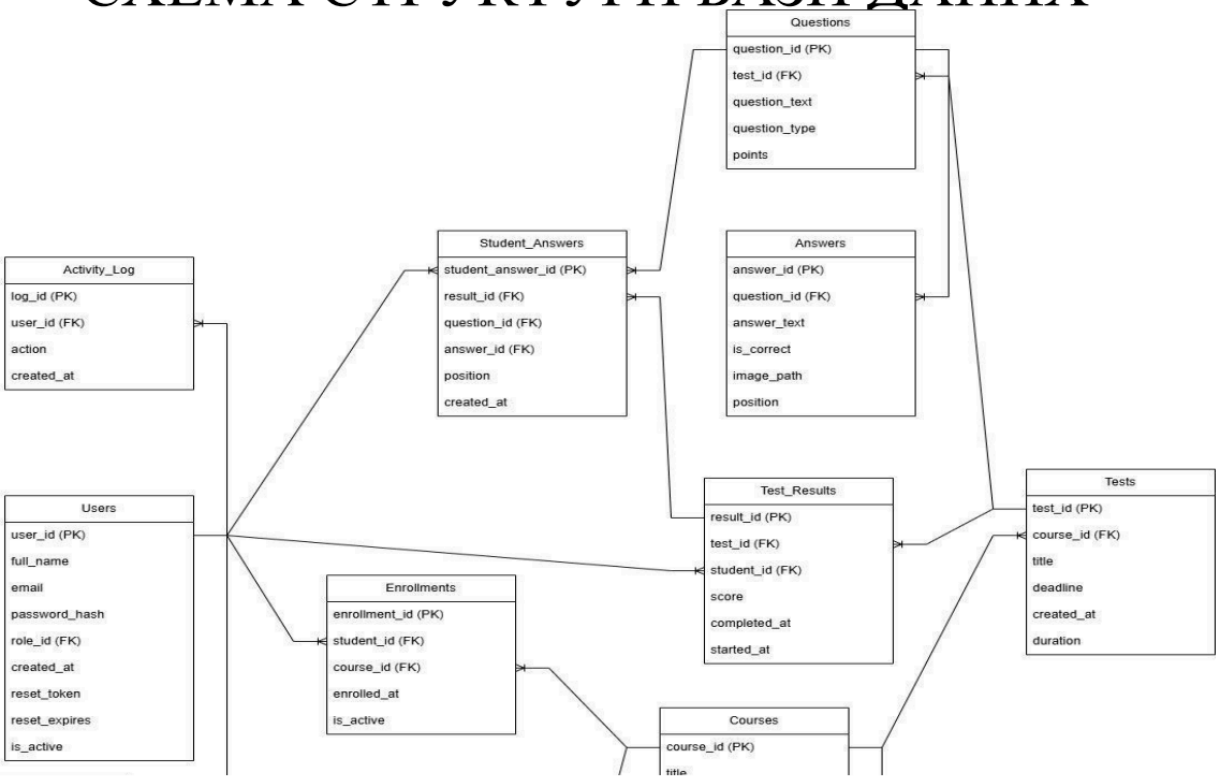
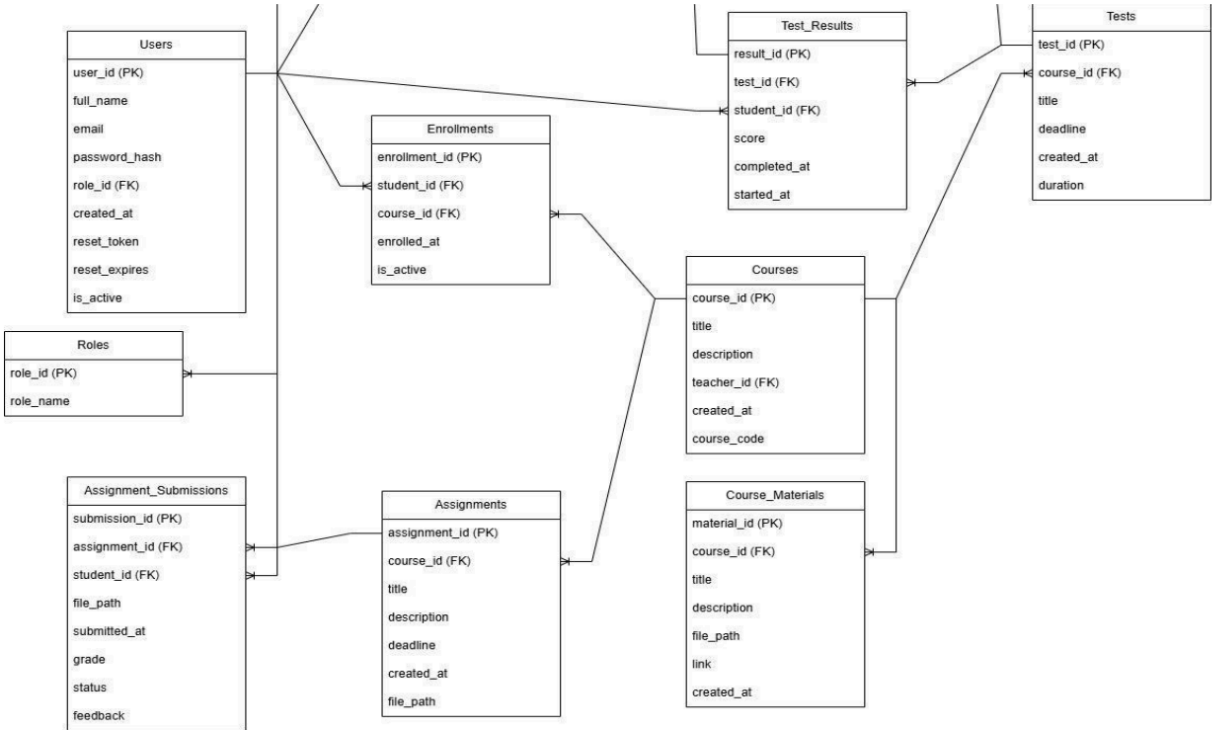
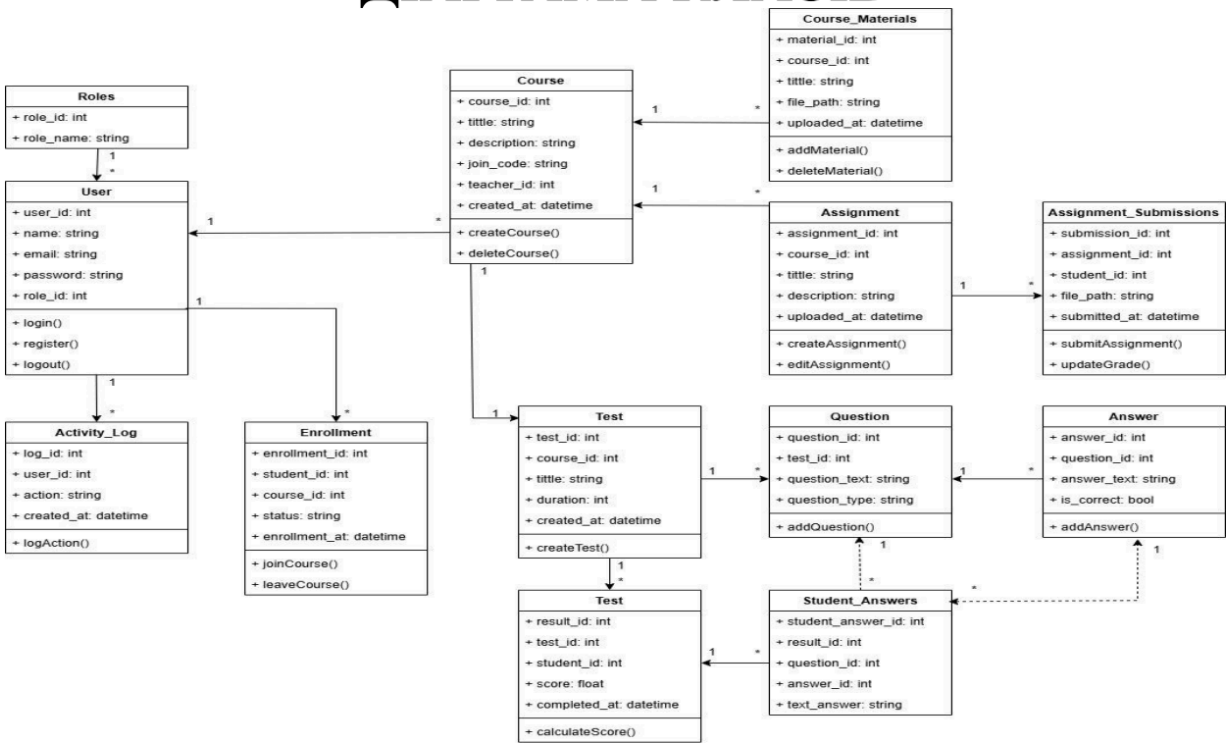


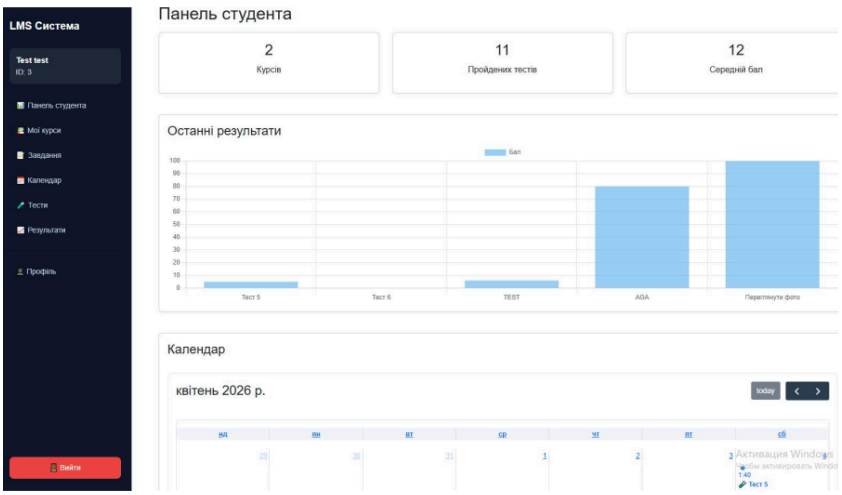
СХЕМА СТРУКТУРИ БАЗИ ДАНИХ



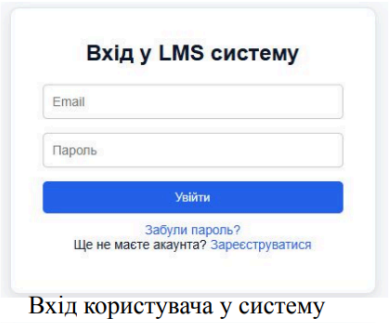
ДІАГРАМА КЛАСІВ



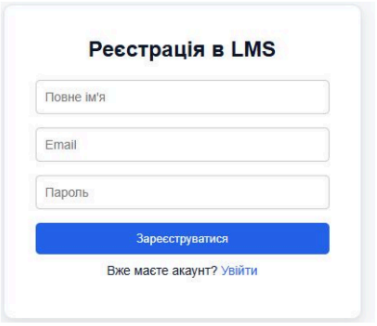
ЕКРАННІ ФОРМИ ІНТЕРФЕЙСУ



Головна сторінка студента

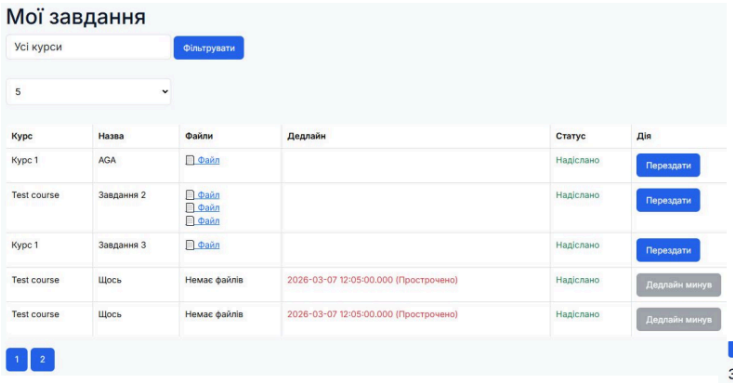


Вхід користувача у систему



Реєстрація користувача у системі

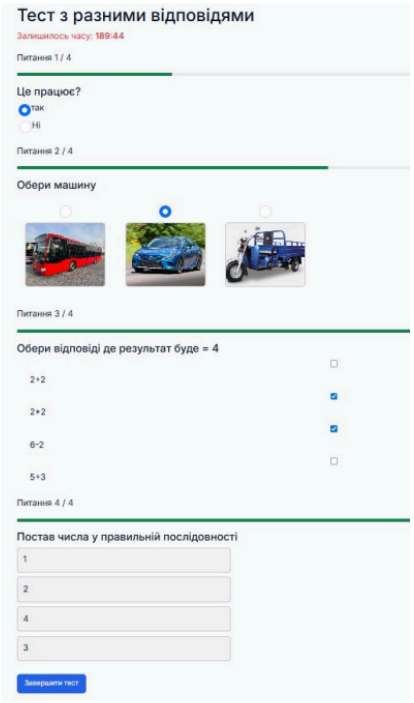
ЕКРАННІ ФОРМИ ІНТЕРФЕЙСУ



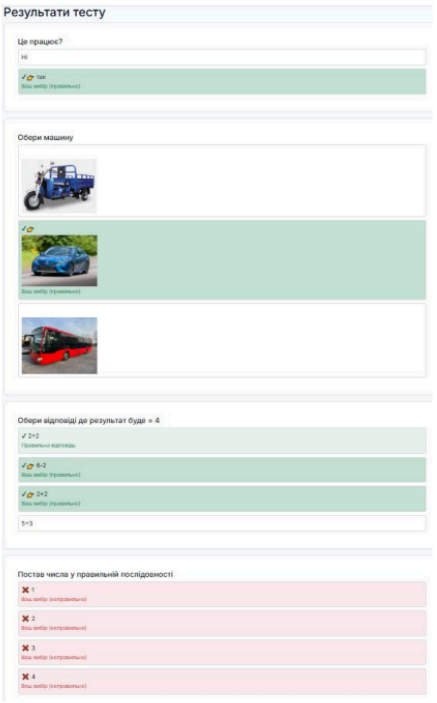
Сторінка завдань для студента

Сторінка виконання завдання студентом

ЕКРАННІ ФОРМИ ІНТЕРФЕЙСУ

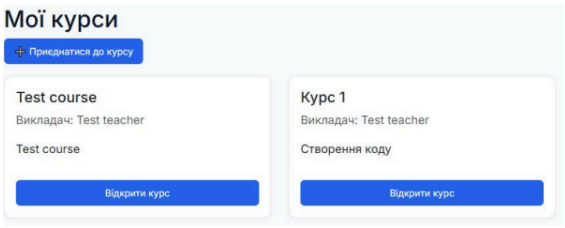


Сторінка проходження студентом тесту

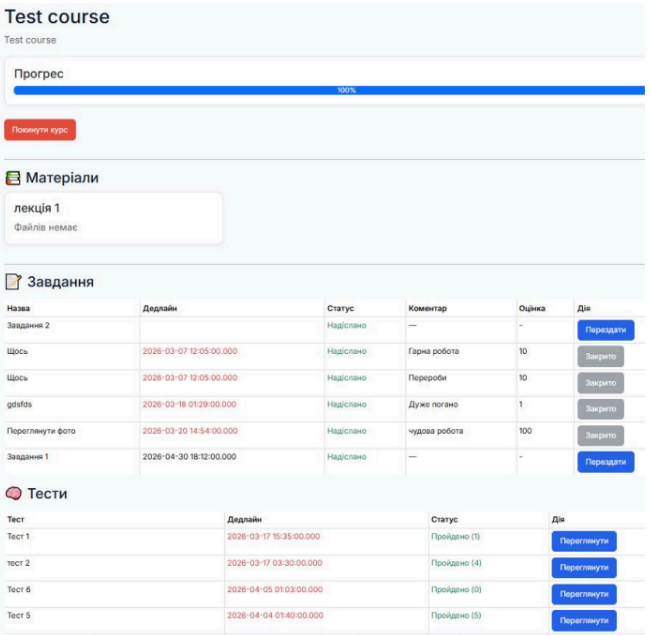


Сторінка результату теста студента

ЕКРАННІ ФОРМИ ІНТЕРФЕЙСУ



Сторінка курсів студента



Сторінка курсу студента зі всіма матеріалами

АПРОБАЦІЯ

Тези доповідей:

1. Єфімов М.О., Аронов А.О., Розробка веб-орієнтованої LMS-системи для дистанційного навчання. // VII Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 23 квітня 2026 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез - К.: ДУІКТ, 2026, С.91-93.
2. Єфімов М.О., Аронов А.О., Особливості розробки модуля онлайн-тестування у веб-орієнтованій системі навчання. // Всеукраїнська науково-технічна конференція «Технології цифрового розвитку: програмні системи та децентралізація», 1-3 травня 2026 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез - (подано до друку)
3. Єфімов М.О., Аронов А.О., Інтелектуальний аналіз у веб-орієнтованій системі навчання. // VI Всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15 травня 2026 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез - (подано до друку)

ВИСНОВКИ

1. Проведено аналіз предметної галузі та існуючих LMS, що дозволило визначити недоліки популярних платформ (Moodle, Google Classroom, Prometheus) та сформувати вимоги до власної системи.
2. Сформовано архітектуру веб-системи, яка включає модулі курсів, завдань, тестування, дедлайнів, роботи з файлами та аналітики.
3. Розроблено структуру бази даних, що забезпечує зберігання користувачів, курсів, завдань, тестів, відповідей, дедлайнів і статистики.
4. Реалізовано модуль управління курсами, який дозволяє створювати, редагувати та організовувати навчальні матеріали.
5. Створено модуль завдань із підтримкою завантаження та передачі файлів до дедлайну, включно зі збереженням історії подань.
6. Розроблено модуль онлайн-тестування з таймером і збереженням результатів.
7. Створено аналітичний модуль, який відображає статистику успішності студентів у вигляді графіків.
8. Проведено тестування системи, яке підтвердило коректність роботи функціоналу, стабільність і відповідність вимогам.

ДОДАТОК Б. ЛІСТИНГИ ОСНОВНИХ МОДУЛІВ

Файл auth/login.php

```
<?php
session_start();
require_once '../config/database.php';

$message = "";

if ($_SERVER["REQUEST_METHOD"] === "POST") {

   $email = trim($_POST["email"]);
   $password = $_POST["password"];

   try {

      $stmt = $pdo->prepare("SELECT * FROM Users WHERE email
= ?");
      $stmt->execute([$email]);

      $user = $stmt->fetch(PDO::FETCH_ASSOC);

      if ($user && password_verify($password,
$user["password_hash"])) {

         if (isset($user["is_active"]) && $user["is_active"] == 0) {
            $message = "Ваш акаунт заблокований";
         } else {

            $_SESSION["user_id"] = $user["user_id"];
            $_SESSION["full_name"] = $user["full_name"];
            $_SESSION["role_id"] = $user["role_id"];

            if ($user["role_id"] == 1) {
               header("Location: ../student/dashboard.php");
            }
            elseif ($user["role_id"] == 2) {
               header("Location: ../teacher/dashboard.php");
            }
            elseif ($user["role_id"] == 3) {
               header("Location: ../admin/dashboard.php");
            }

            exit();
         }

      } else {
         $message = "Невірний email або пароль";
      }

   } catch (PDOException $e) {
      $message = "Помилка: " . $e->getMessage();
   }
}
?>
```

Файл core/auth.php

```
<?php

function checkAuth(){

   if(!isset($_SESSION["user_id"])){

      header("Location: /lms/auth/login.php");
      exit();

   }
```

Файл teacher/courses.php

```
<?php
session_start();
require_once "../config/database.php";

if(!isset($_SESSION["user_id"]) || $_SESSION["role_id"] != 2){
   header("Location: ../auth/login.php");
   exit();
}

$message = "";

$per_page = $_GET["per_page"] ?? 5;
$page = $_GET["page"] ?? 1;

$per_page = (int)$per_page;
$page = (int)$page;

if(!in_array($per_page,[5,10,25])) $per_page = 5;
if($page < 1) $page = 1;

$offset = ($page - 1) * $per_page;

if($_SERVER["REQUEST_METHOD"] == "POST"){

$title = $_POST["title"];
$description = $_POST["description"];
$teacher_id = $_SESSION["user_id"];


$course_code = strtoupper(substr(md5(uniqid()),0,6));

try{

$stmt = $pdo->prepare("
INSERT INTO Courses (title, description, teacher_id, course_code)
VALUES (?, ?, ?, ?)
");

$stmt->execute([$title, $description, $teacher_id, $course_code]);

$message = "Курс створено успішно. Код курсу: ".$course_code;

}catch(PDOException $e){

$message = $e->getMessage();

}

}


$stmt = $pdo->prepare("
SELECT COUNT(*) FROM Courses WHERE teacher_id = ?
");
$stmt->execute([$_SESSION["user_id"]]);

$total = $stmt->fetchColumn();
$total_pages = ceil($total / $per_page);


$stmt = $pdo->prepare("
SELECT * FROM Courses
WHERE teacher_id = ?
ORDER BY course_id DESC
OFFSET $offset ROWS FETCH NEXT $per_page ROWS ONLY
");

$stmt->execute([$_SESSION["user_id"]]);
```

```
$courses = $stmt->fetchAll(PDO::FETCH_ASSOC);;

?>
```

Файл teacher/tests.php

```
<?php
require_once "../core/middleware.php";
require_once "../config/database.php";

requireRole(2);

$teacher_id = $_SESSION["user_id"];
$message = "";


$per_page = $_GET["per_page"] ?? 10;
$page = $_GET["page"] ?? 1;

$per_page = (int)$per_page;
$page = (int)$page;

if(!in_array($per_page,[5,10,25])) $per_page = 10;
if($page < 1) $page = 1;

$offset = ($page - 1) * $per_page;


if(isset($_GET["delete"])){

$test_id = $_GET["delete"];

$stmt = $pdo->prepare("
DELETE FROM Student_Answers
WHERE answer_id IN (
   SELECT answer_id FROM Answers
   WHERE question_id IN (
      SELECT question_id FROM Questions WHERE test_id=?
   )
)
");
$stmt->execute([$test_id]);


$stmt = $pdo->prepare("
DELETE FROM Answers
WHERE question_id IN (
SELECT question_id FROM Questions WHERE test_id=?
)
");
$stmt->execute([$test_id]);


$stmt = $pdo->prepare("
DELETE FROM Questions WHERE test_id=?");
$stmt->execute([$test_id]);


$stmt = $pdo->prepare("
DELETE FROM Test_Results WHERE test_id=?");
$stmt->execute([$test_id]);


$stmt = $pdo->prepare("
DELETE FROM Tests WHERE test_id=?");
$stmt->execute([$test_id]);

$message = "Test deleted";

}


if(isset($_POST["create_test"])){

$title = $_POST["title"];
$course_id = $_POST["course_id"];
$deadline = $_POST["deadline"];
$duration = $_POST["duration"];

if($duration <= 0){

$message = "Duration must be greater than 0";

}else{

if($deadline){
$deadline = date("Y-m-d H:i:s", strtotime($deadline));
}else{
$deadline = null;
}

$stmt = $pdo->prepare("
INSERT INTO Tests (course_id,title,deadline,duration)
VALUES (?,?,?,?)
");

$stmt->execute([$course_id,$title,$deadline,$duration]);

$message = "Test created";

}

}


if(isset($_GET["edit"])){

$edit_id = $_GET["edit"];

$stmt = $pdo->prepare("
SELECT * FROM Tests WHERE test_id=?
");

$stmt->execute([$edit_id]);

$edit_test = $stmt->fetch(PDO::FETCH_ASSOC);

}


if(isset($_POST["update_test"])){

$test_id = $_POST["test_id"];
$title = $_POST["title"];
$deadline = $_POST["deadline"];
$duration = $_POST["duration"];

if($deadline){
$deadline = date("Y-m-d H:i:s", strtotime($deadline));
}else{
$deadline = null;
}

$stmt = $pdo->prepare("
UPDATE Tests
SET title=?, deadline=?, duration=?
WHERE test_id=?
");

$stmt->execute([$title,$deadline,$duration,$test_id]);

$message = "Test updated";

}


$stmt = $pdo->prepare("
SELECT * FROM Courses
WHERE teacher_id = ?
");

$stmt->execute([$teacher_id]);

$courses = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

```
$stmt = $pdo->prepare("
SELECT COUNT(*)
FROM Tests t
JOIN Courses c ON t.course_id = c.course_id
WHERE c.teacher_id = ?
");
$stmt->execute([$teacher_id]);

$total = $stmt->fetchColumn();
$total_pages = ceil($total / $per_page);

$stmt = $pdo->prepare("
SELECT t.*, c.title course_title
FROM Tests t
JOIN Courses c ON t.course_id = c.course_id
WHERE c.teacher_id = ?
ORDER BY t.test_id DESC
OFFSET $offset ROWS FETCH NEXT $per_page ROWS ONLY
");
$stmt->execute([$teacher_id]);

$tests = $stmt->fetchAll(PDO::FETCH_ASSOC);

?>
```

Файл teacher/questions.php

```
<?php

require_once "../core/middleware.php";
require_once "../config/database.php";

requireRole(2);

$test_id = $_GET["test_id"];

$message = "";

$stmt = $pdo->prepare("SELECT title FROM Tests WHERE
test_id=?");
$stmt->execute([$test_id]);
$test = $stmt->fetch(PDO::FETCH_ASSOC);

if(isset($_GET["delete"])){

   $qid = $_GET["delete"];

   $pdo->prepare("DELETE FROM Answers WHERE
question_id=?")->execute([$qid]);
   $pdo->prepare("DELETE FROM Questions WHERE
question_id=?")->execute([$qid]);

   $message="Питання видалено";
}

if(isset($_POST["update_question"])){

   $qid = $_POST["question_id"];
   $text = $_POST["question_text"];
   $type = $_POST["question_type"];
   $points = $_POST["points"];

   $pdo->prepare("
      UPDATE Questions
      SET question_text=?, question_type=?, points=?
      WHERE question_id=?
   ")->execute([$text,$type,$points,$qid]);

   $message="Питання оновлено";
}

if(isset($_POST["add_question"])){

   $question = $_POST["question"];
   $type = $_POST["question_type"];
   $points = $_POST["points"];

   $stmt = $pdo->prepare("
      INSERT INTO Questions
(test_id,question_text,question_type,points)
      VALUES (?,?,?,?)
   ");

   $stmt->execute([$test_id,$question,$type,$points]);

   $message = "Питання додано";
}

$stmt = $pdo->prepare("SELECT * FROM Questions WHERE test_id
= ?");
$stmt->execute([$test_id]);
$questions = $stmt->fetchAll(PDO::FETCH_ASSOC);

?>
```

Файл student/take_test.php

```
<?php
require_once "../core/middleware.php";
require_once "../config/database.php";

requireRole(1);

$student_id = $_SESSION["user_id"];
$test_id = $_GET["test_id"] ?? 0;

$message = "";


$stmt = $pdo->prepare("SELECT * FROM Tests WHERE test_id=?");
$stmt->execute([$test_id]);
$test = $stmt->fetch(PDO::FETCH_ASSOC);

if (!$test) {
   die("Тест не знайдено");
}


if ($test["deadline"] && strtotime($test["deadline"]) < time()) {
   die("Термін виконання тесту минув.");
}

$stmt = $pdo->prepare("
SELECT TOP 1 * FROM Test_Results
WHERE test_id=? AND student_id=? AND completed_at IS NOT
NULL
ORDER BY result_id DESC
");
$stmt->execute([$test_id, $student_id]);
$completed = $stmt->fetch(PDO::FETCH_ASSOC);

if ($completed) {
   $message = "Ви вже виконали цей тест. Ваш бал: " .
$completed["score"];
}


$stmt = $pdo->prepare("
SELECT TOP 1 * FROM Test_Results
WHERE test_id=? AND student_id=? AND completed_at IS NULL
ORDER BY result_id DESC
");
$stmt->execute([$test_id, $student_id]);
$attempt = $stmt->fetch(PDO::FETCH_ASSOC);

if (!$attempt && !$completed) {

   $stmt = $pdo->prepare("
      INSERT INTO Test_Results (test_id,student_id,score,started_at)
      VALUES (?,?,0,GETDATE())
   ");
   $stmt->execute([$test_id, $student_id]);
```

```
    $stmt = $pdo->prepare("
        SELECT TOP 1 * FROM Test_Results
        WHERE test_id=? AND student_id=? AND completed_at IS
NULL
        ORDER BY result_id DESC
    ");
    $stmt->execute([$test_id, $student_id]);
    $attempt = $stmt->fetch(PDO::FETCH_ASSOC);
}


function normalize_sqlsrv_datetime($value)
{
    if ($value instanceof DateTime) {
        $value->setTimezone(new
DateTimeZone(date_default_timezone_get()));
        return $value->getTimestamp();
    }

    if (!$value) {
        return false;
    }

    $str = preg_replace("/\.\d+$/", "", (string)$value);

    $dt = new DateTime($str, new DateTimeZone("UTC"));

    $dt->setTimezone(new
DateTimeZone(date_default_timezone_get()));

    return $dt->getTimestamp();
}


if ($attempt) {

    $started_at = normalize_sqlsrv_datetime($attempt["started_at"]);

    if ($started_at === false) {
        $started_at = time();
    }

    $duration_minutes = intval($test["duration"]);
    if ($duration_minutes <= 0) {
        $duration_minutes = 2;
    }

    $duration = $duration_minutes * 60;

    $time_passed = time() - $started_at;
    $time_left = $duration - $time_passed;

    if ($time_left < 0) {
        $time_left = 0;
    }

    if ($time_left === 0) {
        die("Час вичерпано!");
    }

} else {
    $time_left = 0;
}


if (isset($_POST["submit_test"]) && $attempt) {

    $stmt = $pdo->prepare("DELETE FROM Student_Answers
WHERE result_id=?");
    $stmt->execute([$attempt["result_id"]]);

    $score = 0;

    $stmt = $pdo->prepare("SELECT * FROM Questions WHERE
test_id=?");
    $stmt->execute([$test_id]);
    $questions = $stmt->fetchAll(PDO::FETCH_ASSOC);

    foreach ($questions as $q) {

        $qid = $q["question_id"];
        $type = $q["question_type"];
        $points = $q["points"] ?? 1;

        $stmt = $pdo->prepare("SELECT * FROM Answers WHERE
question_id=?");
        $stmt->execute([$qid]);
        $answers = $stmt->fetchAll(PDO::FETCH_ASSOC);

        if ($type == "single") {

            $answer_id = $_POST["answers"][$qid] ?? null;

            if ($answer_id) {

                $stmt = $pdo->prepare("
                    INSERT INTO Student_Answers (result_id, question_id,
answer_id)
                    VALUES (?,?,?)
                ");
                $stmt->execute([$attempt["result_id"], $qid, $answer_id]);

                $stmt = $pdo->prepare("SELECT is_correct FROM Answers
WHERE answer_id=?");
                $stmt->execute([$answer_id]);
                $a = $stmt->fetch();

                if ($a && $a["is_correct"]) {
                    $score += $points;
                }
            }
        }

        if ($type == "multiple") {

            $correct = [];

            foreach ($answers as $a) {
                if ($a["is_correct"]) $correct[] = $a["answer_id"];
            }

            $selected = $_POST["answers"][$qid] ?? [];
            if (!is_array($selected)) $selected = [$selected];

            foreach ($selected as $sel) {
                $stmt = $pdo->prepare("
                    INSERT INTO Student_Answers (result_id, question_id,
answer_id)
                    VALUES (?,?,?)
                ");
                $stmt->execute([$attempt["result_id"], $qid, $sel]);
            }

            $per_point = count($correct) ? ($points / count($correct)) : 0;

            foreach ($selected as $sel) {
                if (in_array($sel, $correct)) {
                    $score += $per_point;
                }
            }
        }

        if ($type == "image") {

            $correct = [];

            foreach ($answers as $a) {
                if ($a["is_correct"]) $correct[] = $a["answer_id"];
            }

            $selected = $_POST["answers"][$qid] ?? null;

            if ($selected) {

                $stmt = $pdo->prepare("
                    INSERT INTO Student_Answers (result_id, question_id,
answer_id)
```

```
                VALUES (?,?,?)
            ");
            $stmt->execute([$attempt["result_id"], $qid, $selected]);

            if (in_array($selected, $correct)) {
                $score += $points;
            }
        }
    }

    if ($type == "ordering") {

        $correct = [];

        foreach ($answers as $a) {
            $correct[$a["answer_id"]] = $a["position"];
        }

        $student = $_POST["ordering"][$qid] ?? [];

        $correct_count = 0;

        foreach ($student as $aid => $pos) {

            $stmt = $pdo->prepare("
                INSERT INTO Student_Answers (result_id, question_id,
answer_id, position)
                VALUES (?,?,?,?)
            ");
            $stmt->execute([$attempt["result_id"], $qid, $aid, $pos]);

            if (isset($correct[$aid]) && $correct[$aid] == $pos) {
                $correct_count++;
            }
        }

        $total = count($correct);

        if ($total > 0) {
            $score += ($points / $total) * $correct_count;
        }
    }
}

$score = round($score);

$stmt = $pdo->prepare("
    UPDATE Test_Results
    SET score=?, completed_at=GETDATE()
    WHERE result_id=?
");
$stmt->execute([$score, $attempt["result_id"]]);

header("Location: take_test.php?test_id=" . $test_id);
exit();
}


$stmt = $pdo->prepare("SELECT * FROM Questions WHERE
test_id=?");
$stmt->execute([$test_id]);
$questions = $stmt->fetchAll(PDO::FETCH_ASSOC);

?>
```

Файл teacher/assignments.php

```
<?php

require_once "../core/middleware.php";
require_once "../config/database.php";

requireRole(2);

$teacher_id=$_SESSION["user_id"];

$message="";


$per_page = $_GET["per_page"] ?? 10;
$page = $_GET["page"] ?? 1;

$per_page = (int)$per_page;
$page = (int)$page;

if(!in_array($per_page,[5,10,25])) $per_page = 10;
if($page < 1) $page = 1;

$offset = ($page - 1) * $per_page;


if($_SERVER["REQUEST_METHOD"]==="POST"){

if(isset($_POST["ajax"])){

$title=$_POST["title"];
$description=$_POST["description"];
$deadline=$_POST["deadline"];
$course_id=$_POST["course_id"];


$uploadedFiles = [];

if(!empty($_FILES["files"]["name"])){

$target="../uploads/assignments/";

foreach($_FILES["files"]["name"] as $key => $name){

$tmp = $_FILES["files"]["tmp_name"][$key];

$filename = time()."_".$name;

move_uploaded_file($tmp, $target.$filename);

$uploadedFiles[] = $filename;

}

}

$file_path = !empty($uploadedFiles) ? json_encode($uploadedFiles) :
null;

if($deadline){
$deadline=date("Y-m-d H:i:s",strtotime($deadline));
}else{
$deadline=null;
}

$stmt=$pdo->prepare("
INSERT INTO Assignments
(course_id,title,description,deadline,file_path)
VALUES(?,?,?,?,?)
");

$stmt->execute([
$course_id,
$title,
$description,
$deadline,
$file_path
]);

echo json_encode(["status"=>"ok"]);
exit;

}

}


$stmt=$pdo->prepare("
SELECT * FROM Courses
WHERE teacher_id=?
```

```
");

$stmt->execute([$teacher_id]);

$courses=$stmt->fetchAll(PDO::FETCH_ASSOC);

$stmt = $pdo->prepare("
SELECT COUNT(*)
FROM Assignments a
JOIN Courses c ON a.course_id=c.course_id
WHERE c.teacher_id=?
");
$stmt->execute([$teacher_id]);

$total = $stmt->fetchColumn();
$total_pages = ceil($total / $per_page);


$stmt=$pdo->prepare("
SELECT a.*, c.title course_title
FROM Assignments a
JOIN Courses c ON a.course_id=c.course_id
WHERE c.teacher_id=?
ORDER BY a.created_at DESC
OFFSET $offset ROWS FETCH NEXT $per_page ROWS ONLY
");

$stmt->execute([$teacher_id]);

$assignments=$stmt->fetchAll(PDO::FETCH_ASSOC);

?>
```

Файл student/submit_assignment.php

```
<?php
require_once "../core/middleware.php";
require_once "../config/database.php";

requireRole(1);

$assignment_id=$_GET["assignment_id"];
$student_id=$_SESSION["user_id"];

$message="";

$stmt=$pdo->prepare("SELECT * FROM Assignments WHERE
assignment_id=?");
$stmt->execute([$assignment_id]);
$assignment=$stmt->fetch(PDO::FETCH_ASSOC);

if(!$assignment){
   die("Assignment not found");
}

$stmt=$pdo->prepare("
SELECT * FROM Assignment_Submissions
WHERE assignment_id=? AND student_id=?
");
$stmt->execute([$assignment_id,$student_id]);
$submission=$stmt->fetch(PDO::FETCH_ASSOC);

$deadline_passed=$assignment["deadline"] &&
strtotime($assignment["deadline"])<time();

if(isset($_GET["delete"]) && $submission){

   $file=$_GET["delete"];

   $files=json_decode($submission["file_path"],true);
   if(!is_array($files)) $files=[];

   $files=array_filter($files,function($f)use($file){
      return $f!=$file;
   });

   @unlink("../uploads/submissions/".$file);

   $new=json_encode(array_values($files));

   $stmt=$pdo->prepare("
   UPDATE Assignment_Submissions
   SET file_path=?
   WHERE submission_id=?
   ");
   $stmt->execute([$new,$submission["submission_id"]]);

   header("Location:
submit_assignment.php?assignment_id=".$assignment_id);
   exit;
}

if($_SERVER["REQUEST_METHOD"]=="POST"){

   $dir="../uploads/submissions/";
   if(!is_dir($dir)) mkdir($dir,0777,true);

   $uploaded = [];

   if($submission){
      $old_files = json_decode($submission["file_path"], true);
      if(is_array($old_files)){
         $uploaded = $old_files;
      }
   }

   if(isset($_FILES["files"])){
      foreach($_FILES["files"]["name"] as $i=>$name){

         if($_FILES["files"]["error"][$i]==0){


$filename=time()."_".preg_replace("/[^a-zA-Z0-9\.]/","",$name);

            move_uploaded_file(
               $_FILES["files"]["tmp_name"][$i],
               $dir.$filename
            );

            $uploaded[]=$filename;
         }
      }
   }

   $json=json_encode($uploaded);

   if($submission){

      $stmt=$pdo->prepare("
      UPDATE Assignment_Submissions
      SET file_path=?,submitted_at=GETDATE(),status='resubmitted'
      WHERE submission_id=?
      ");
      $stmt->execute([$json,$submission["submission_id"]]);

      $message="Перездали";

   }else{

      $stmt=$pdo->prepare("
      INSERT INTO Assignment_Submissions
      (assignment_id,student_id,file_path,status)
      VALUES(?,?,?,?)
      ");
      $stmt->execute([$assignment_id,$student_id,$json,"submitted"]);

      $message="Надіслано";
   }

   $stmt=$pdo->prepare("
   SELECT * FROM Assignment_Submissions
   WHERE assignment_id=? AND student_id=?
   ");
   $stmt->execute([$assignment_id,$student_id]);
   $submission=$stmt->fetch(PDO::FETCH_ASSOC);
```

```
}


?>
```

Файл teacher/submissions.php

```
<?php

require_once "../core/middleware.php";
require_once "../config/database.php";

requireRole(2);

$message="";


$course=$_GET["course"] ?? '';
$student=$_GET["student"] ?? '';
$title=$_GET["title"] ?? '';
$assignment = $_GET["assignment"] ?? '';


$per_page = $_GET["per_page"] ?? 10;
$page = $_GET["page"] ?? 1;

$per_page = (int)$per_page;
$page = (int)$page;

if(!in_array($per_page,[5,10,25])) $per_page = 10;
if($page < 1) $page = 1;

$offset = ($page - 1) * $per_page;


if(isset($_POST["grade"])){

$submission_id=$_POST["submission_id"];
$grade=min(100,max(0,$_POST["grade_value"]));
$feedback=$_POST["feedback"] ?? '';

$stmt=$pdo->prepare("
UPDATE Assignment_Submissions
SET grade=?, feedback=?, status='graded'
WHERE submission_id=?
");

$stmt->execute([$grade,$feedback,$submission_id]);

$message="Оцінку та відгук збережено";

}


$stmt=$pdo->prepare("SELECT course_id,title FROM Courses
WHERE teacher_id=?");
$stmt->execute([$_SESSION["user_id"]]);
$courses=$stmt->fetchAll();


$stmt=$pdo->prepare("
SELECT a.assignment_id, a.title, a.course_id
FROM Assignments a
JOIN Courses c ON a.course_id = c.course_id
WHERE c.teacher_id = ?
");
$stmt->execute([$_SESSION["user_id"]]);

$all_assignments=$stmt->fetchAll(PDO::FETCH_ASSOC);

$countQuery="
SELECT COUNT(*)
FROM Assignment_Submissions s
JOIN Users u ON s.student_id=u.user_id
JOIN Assignments a ON s.assignment_id=a.assignment_id
JOIN Courses c ON a.course_id=c.course_id
WHERE c.teacher_id=?
";

$countParams=[$_SESSION["user_id"]];

if($course){
$countQuery.=" AND c.course_id=?";
$countParams[]=$course;
}

if($student){
$countQuery.=" AND u.full_name LIKE ?";
$countParams[]="%$student%";
}

if($title){
$countQuery.=" AND a.title LIKE ?";
$countParams[]="%$title%";
}

if($assignment){
$countQuery.=" AND a.assignment_id=?";
$countParams[]=$assignment;
}

$stmt=$pdo->prepare($countQuery);
$stmt->execute($countParams);

$total = $stmt->fetchColumn();
$total_pages = ceil($total / $per_page);


$query="
SELECT
s.submission_id,
s.file_path,
s.grade,
s.feedback,
s.submitted_at,
u.full_name student,
a.title assignment,
c.title course,
c.course_id

FROM Assignment_Submissions s
JOIN Users u ON s.student_id=u.user_id
JOIN Assignments a ON s.assignment_id=a.assignment_id
JOIN Courses c ON a.course_id=c.course_id

WHERE c.teacher_id=?
";

$params=[$_SESSION["user_id"]];

if($course){
$query.=" AND c.course_id=?";
$params[]=$course;
}

if($student){
$query.=" AND u.full_name LIKE ?";
$params[]="%$student%";
}

if($title){
$query.=" AND a.title LIKE ?";
$params[]="%$title%";
}

if($assignment){
$query.=" AND a.assignment_id=?";
$params[]=$assignment;
}

$query.=" ORDER BY s.submitted_at DESC
OFFSET $offset ROWS FETCH NEXT $per_page ROWS ONLY";

$stmt=$pdo->prepare($query);
$stmt->execute($params);
```

```
$submissions=$stmt->fetchAll(PDO::FETCH_ASSOC);

?>
```

Файл teacher/analytics.php

```
<?php

require_once "../core/middleware.php";
require_once "../config/database.php";

requireRole(2);

$teacher_id = $_SESSION["user_id"];


$stmt = $pdo->prepare("SELECT course_id,title FROM Courses
WHERE teacher_id=?");
$stmt->execute([$teacher_id]);
$courses = $stmt->fetchAll(PDO::FETCH_ASSOC);

$course_id = $_GET["course_id"] ?? null;


$assignments = [];
$submitted = [];
$not_submitted = [];
$avg_grades = [];

$students_labels = [];
$students_done = [];
$students_not_done = [];

$tests_labels = [];
$tests_avg = [];

$risk_students = [];

$student_progress = [];

$total_students = 0;
$total_assignments = 0;

if($course_id){


$stmt = $pdo->prepare("SELECT COUNT(*) FROM Enrollments
WHERE course_id=?");
$stmt->execute([$course_id]);
$total_students = (int)$stmt->fetchColumn();


$stmt = $pdo->prepare("
SELECT a.assignment_id,a.title,
COUNT(s.submission_id) submitted,
AVG(s.grade) avg_grade
FROM Assignments a
LEFT JOIN Assignment_Submissions s ON
a.assignment_id=s.assignment_id
WHERE a.course_id=?
GROUP BY a.assignment_id,a.title
");
$stmt->execute([$course_id]);
$data = $stmt->fetchAll(PDO::FETCH_ASSOC);

$total_assignments = count($data);

foreach($data as $row){

$assignments[] = $row["title"];

$sub = (int)$row["submitted"];

$submitted[] = $sub;
$not_submitted[] = max(0, $total_students - $sub);
$avg_grades[] = $row["avg_grade"] ? round($row["avg_grade"],2) : 0;
}


$stmt = $pdo->prepare("
SELECT u.full_name,
COUNT(s.submission_id) done
FROM Users u
JOIN Enrollments e ON u.user_id=e.student_id
LEFT JOIN Assignment_Submissions s ON u.user_id=s.student_id
LEFT JOIN Assignments a ON s.assignment_id=a.assignment_id AND
a.course_id=?
WHERE e.course_id=?
GROUP BY u.full_name
");
$stmt->execute([$course_id,$course_id]);

$students = $stmt->fetchAll(PDO::FETCH_ASSOC);

foreach($students as $st){

$name = $st["full_name"];
$done = (int)$st["done"];

$students_labels[] = $name;
$students_done[] = $done;
$students_not_done[] = max(0, $total_assignments - $done);


if($done < ($total_assignments/2)){
$risk_students[] = $name;
}


$student_progress[$name] = [
"done"=>$done,
"total"=>$total_assignments
];

}


$stmt = $pdo->prepare("
SELECT t.title, AVG(r.score) avg_score
FROM Tests t
LEFT JOIN Test_Results r ON t.test_id=r.test_id
WHERE t.course_id=?
GROUP BY t.title
");
$stmt->execute([$course_id]);

$tests = $stmt->fetchAll(PDO::FETCH_ASSOC);

foreach($tests as $t){
$tests_labels[] = $t["title"];
$tests_avg[] = round($t["avg_score"],2);
}


$stmt = $pdo->prepare("
SELECT
   range_group,
   COUNT(*) as total
FROM (
   SELECT
      student_id,
      AVG(grade) as avg_grade,
      CASE
         WHEN AVG(grade) >= 90 THEN '90-100'
         WHEN AVG(grade) >= 75 THEN '75-89'
         WHEN AVG(grade) >= 60 THEN '60-74'
         WHEN AVG(grade) >= 40 THEN '40-59'
         ELSE '0-39'
      END as range_group,
      CASE
         WHEN AVG(grade) >= 90 THEN 1
         WHEN AVG(grade) >= 75 THEN 2
         WHEN AVG(grade) >= 60 THEN 3
         WHEN AVG(grade) >= 40 THEN 4
```

```
        ELSE 5
      END as sort_order
   FROM Assignment_Submissions s
   JOIN Assignments a ON s.assignment_id=a.assignment_id
   WHERE a.course_id=? AND grade IS NOT NULL
   GROUP BY student_id
) t
GROUP BY range_group, sort_order
ORDER BY sort_order
");

$stmt->execute([$course_id]);

$grade_dist = $stmt->fetchAll(PDO::FETCH_ASSOC);

$dist_labels = [];
$dist_values = [];

foreach($grade_dist as $d){
   $dist_labels[] = $d["range_group"];
   $dist_values[] = $d["total"];
}

$stmt = $pdo->prepare("
SELECT
   u.full_name,
   CASE
      WHEN AVG(s.grade) >= 90 THEN '90-100'
      WHEN AVG(s.grade) >= 75 THEN '75-89'
      WHEN AVG(s.grade) >= 60 THEN '60-74'
      WHEN AVG(s.grade) >= 40 THEN '40-59'
      ELSE '0-39'
   END as range_group
FROM Assignment_Submissions s
JOIN Assignments a ON s.assignment_id=a.assignment_id
JOIN Users u ON s.student_id = u.user_id
WHERE a.course_id=? AND s.grade IS NOT NULL
GROUP BY s.student_id, u.full_name
");

$stmt->execute([$course_id]);

$students_by_grade = [];

while($row = $stmt->fetch(PDO::FETCH_ASSOC)){
   $students_by_grade[$row["range_group"]][] = $row["full_name"];
}


$stmt = $pdo->prepare("
SELECT AVG(grade)
FROM Assignment_Submissions s
JOIN Assignments a ON s.assignment_id=a.assignment_id
WHERE a.course_id=? AND grade IS NOT NULL
");

$stmt->execute([$course_id]);
$course_avg = round($stmt->fetchColumn(),2);
$avg = $course_avg ?? 0;

}

?>
```

Файл config/database.php

```
<?php

$serverName = "DESKTOP-BKP438T\\SQLEXPRESS";
$database = "LearningManagementSystem";

try {
   $pdo = new
PDO("sqlsrv:Server=$serverName;Database=$database;TrustServerCer
tificate=true");
   $pdo->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

} catch (PDOException $e) {
   die("Помилка підключення: " . $e->getMessage());
}
```