Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)
(рівень вищої освіти)

Розробка автоматизованої системи обліку та аналізу поточної успішності студентів
(тема)

Виконав:
здобувач 2 року навчання,
групи КІТПВм-24-2

Євген ОБРИВКО

Спеціальності 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка

Тип програми Освітньо-професійна

Освітня програма Комп'ютерно-інтегровані технологічні процеси і виробництва

Керівник доц. Наталія ДЕМСЬКА

Допускається до захисту
Зав. кафедри КІТАР _____ Ігор НЕВЛЮДОВ
(підпис) (прізвище, ініціали)

2025р.

Я, Обривко Євген Віталійович, як здобувач вищої освіти ХНУРЕ, розумію і підтримую політику закладу з академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Я не використовував штучний інтелект для підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

15 грудня 2025 р. Євген ОБРИВКО

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет АКТ
Кафедра КІТАР
Рівень вищої освіти другий (магістерський)
Спеціальність 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка
Тип програми Освітньо-професійна
Освітня програма Комп'ютерно-інтегровані технологічні процеси і виробництва

ЗАТВЕРДЖУЮ:
Зав. кафедри КІТАР _____
(підпис)
« ___»_____2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Здобувачеві Обривко Євгену Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка автоматизованої системи обліку та аналізу поточної успішності студентів

Затверджена наказом по університету від 10.11.2025 р. № 1029 Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 20.01.2025 р.
3. Вихідні дані до роботи 3.1 Використане програмне забезпечення: серверна мова програмування PHP; система управління БД MySQL; HTML та CSS для створення розмітки та дизайну сторінок, JavaScript (AJAX); в процесі прогнозування оцінок використовується математичний алгоритм лінійної регресії; 3.2 Для написання коду використовується редактор VisualStudio Code; 3.3 Для запуску локального сервера, виконання скриптів PHP та адміністрування БД використовується пакет програмного забезпечення MAMP, що включає такі компоненти, як PHP, Apache, MySQL, phpMyAdmin.
4. Перелік питань, що потрібно опрацювати в роботі 4.1 Вступ; 4.2 Аналіз існуючих систем управління навчанням; 4.3 Постановка задач, які має вирішувати система; 4.4 Вибір та обґрунтування програмних засобів для реалізації поставлених задач; 4.5 Вибір засобів для реалізації прогнозування успішності; 4.6 Створення БД, опис зв'язків та таблиць даних; 4.7 Реалізація програмної логіки роботи системи; 4.8 Розрахунок парметрів стійкості системи; 4.9 Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

Графічний матеріал у вигляді презентації – 12 арк. ф. А 4

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Приміт-ка
1	Аналіз існуючих систем управління навчанням	01.09.25-16.09.25	виконано
2	Постановка задач дослідження та вибір програмних засобів для їх реалізації	17.09.25-02.10.25	виконано
3	Створення бази даних, опис зв'язків та таблиць даних	03.10.25-27.10.25	виконано
4	Реалізація програмної логіки роботи системи	28.10.25-15.11.25	виконано
5	Тестування системи, оптимізація та підготовка висновків	16.11.25-14.12.25	виконано
6	Оформлення пояснювальної записки	15.12.25	виконано
7	Подання роботи для проходження нормоконтролю	15.12.25	виконано
8	Подання роботи для перевірки на академічну доброчесність	17.12.25	виконано
9	Подання роботи на рецензію	19.12.25	виконано
10	Подання роботи на підпис зав. кафедри	20.12.25	виконано
11	Подання кваліфікаційної роботи в ЕК	21.12.25	виконано

Дата видачі завдання 01.09.2025 р.

Здобувач _____Євген ОБРИВКО

Керівник роботи _____ доц. Наталія ДЕМСЬКА

РЕФЕРАТ

Пояснювальна записка: 160 с., 8 табл., 6 рис., 3 дод., 21 джерело.

СИСТЕМА ОБЛІКУ, АНАЛІЗ, КОНТРОЛЬ УСПІШНОСТІ, АВТОМАТИЗАЦІЯ, ЯКІСТЬ ЗНАНЬ, ДИСТАНЦІЙНЕ НАВЧАННЯ, УЧНІ, СТУДЕНТИ, ТЕСТУВАННЯ ЗНАНЬ.

Об’єктом дослідження є процеси автоматизованої перевірки та оцінювання якості знань, а також методи прогнозування майбутньої успішності студентів та ведення навчальної статистики.

Предметом дослідження є веб-орієнтована інформаційна система перевірки та обліку успішності навчання, її структура, алгоритм роботи та методи і засоби реалізації компонентів.

Мета кваліфікаційної роботи полягає у розробці автоматизованої системи обліку та аналізу поточної успішності студентів. Розроблювана система має забезпечувати швидку та зручну взаємодію між викладачами та студентами в процесі перевірки та оцінювання знань, надавати додаткові можливості ведення статистики та прогнозування рівня майбутньої успішності, що у майбутньому позитивно вплине на об’єктивність оцінювання та підвищуватиме інтерес до досягнення навчальних результатів.

Окрім того, результати виконання роботи можна віднести до цілі сталого розвитку 4 «Якісна освіта», зокрема до пунктів 4.4 «Підвищити якість вищої освіти та забезпечити її тісний зв’язок з наукою, сприяти формуванню в країні міст освіти та науки» та 4.7 «Створити у школах сучасні умови навчання, включаючи інклюзивне, на основі інноваційних підходів».

ABSTRACT

Explanatory note: 160 p., 8 tables, 6 figures, 3 appendices, 21 sources.

ACCOUNTING SYSTEM, ANALYSIS, PERFORMANCE CONTROL, AUTOMATION, KNOWLEDGE QUALITY, DISTANCE LEARNING, PUPILS, STUDENTS, KNOWLEDGE TESTING.

The object of the research is the processes of automated verification and assessment of the quality of knowledge, as well as methods of predicting future student success and maintaining educational statistics.

The subject of the research is a web-based information system for verifying and accounting for learning success, its structure, work algorithm, and methods and means of implementing components.

The purpose of the qualification work is to develop an automated system for accounting and analyzing current student success. The developed system should provide fast and convenient interaction between teachers and students in the process of verifying and assessing knowledge, provide additional opportunities for maintaining statistics and predicting the level of future success, which in the future will positively affect the objectivity of assessment and increase interest in achieving educational results.

In addition, the results of the work can be attributed to Sustainable Development Goal 4 "Quality Education", in particular to points 4.4 "Improve the quality of higher education and ensure its close connection with science, promote the formation of cities of education and science in the country" and 4.7 "Create modern learning conditions in schools, including inclusive ones, based on innovative approaches."

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних;

СУБД – Система управління базами даних;

AJAX – (Asynchronous JavaScript and XML) – технологія асинхронного обміну даними між веб-сторінкою та базою даних.

CPU (Central Processor Unit) – центральний процесор;

CSS (Cascading Style Sheets) – спеціальна мова стилів сторінок;

ER-модель (Entity-Relationship model) – модель залежності сутність-зв’язок;

HTML (HyperText Markup Language) – стандартизована мова розмітки документів;

LMS (Learning Management Systems) – система управління навчанням;

Moodle (Modular Object-Oriented Dynamic Learning Environment) – система управління навчальним середовищем;

MTBF (Mean Time Between Failures) – показник надійності обладнання;

PHP (Hypertext Preprocessor) – скриптова мова програмування;

PSU (Power Supply Unit) – блок живлення;

RAM (Random Access Memory) – оперативна пам’ять;

SQL (Structured Query Language) – мова запитів до бази даних;

SSD (Solid State Drive) – твердотільний накопичувач;

VS Code (Visual Studio Code) – середовище розробки та редактор коду.

ВСТУП

У нинішньому часі навчальні заклади стикаються все з більшою кількістю викликів. Сучасні вимоги до навчання спонукають їх до створення все більш комфортних та прогресивних умов навчання, а різноманітні фактори та події у світі підштовхують до все більшого залучення технологій дистанційного навчання. Особливо актуальним це є в контексті війни чи пандемії, коли звичні методи навчання не є безпечними як для викладачів, так і для здобувачів освіти [1]. У зв’язку з цим виникає потреба у створенні зручних та комфортних умов та інструментів для онлайн-навчання. Зокрема варто приділити особливу увагу методам контролю, перевірки, обліку та прогнозування навчальних результатів. Важливо, щоб при використанні сучасних технологій та методів дистанційного навчання, якість отримання та перевірки знань лише зростала та ставала більш інноваційною, автоматизованою та легкодоступною.

Оцінювання виконує низку важливих функцій, що забезпечують ефективність освітнього процесу та роблять його не лише інструментом вимірювання, а й потужним засобом розвитку, підтримки та управління освітнім процесом, зокрема: діагностувальна функція дає змогу визначити рівень сформованості компетентностей учнів, виявити причини труднощів і прогалин у знаннях; коригувальна функція допомагає спрямувати навчання у правильному напрямі; прогностична функція дає можливість прогнозувати освітні потреби, планувати індивідуальну траєкторію розвитку, обирати напрями поглибленого вивчення предметів; управлінська функція забезпечує прийняття обґрунтованих рішень на рівні викладача та освітньої установи.

Метою даної кваліфікаційної роботи є розробка автоматизованої системи обліку та аналізу поточної успішності студентів. Дана система має забезпечити швидку та зручну взаємодію між викладачами та студентами в процесі перевірки та оцінювання знань.

Об’єктом дослідження є процеси автоматизованої перевірки та оцінювання якості знань, а також методи прогнозування майбутньої успішності студентів та ведення навчальної статистики.

Предметом дослідження є веб-орієнтована інформаційна система перевірки та обліку успішності навчання, її структура, алгоритм роботи та методи і засоби реалізації компонентів.

Методологія дослідження базується на єдності системного, особистісно-орієнтованого та технологічного підходів, що дало змогу розробити студентоцентровану, алгоритмізовану, орієнтовану на структуру навчальної дії методику конструювання індивідуальних завдань оцінювання освітніх досягнень здобувачів освіти.

Для досягнення поставленої мети необхідно виконати наступні завдання:

– провести аналіз сучасних існуючих систем управління навчанням;

– дослідити структуру LMS-систем, визначити переваги та недоліки різних варіантів реалізації;

– зробити постановку задач дослідження та вибір програмних засобів для їх реалізації;

– розробити та описати базу даних, її зв'язки, сутності та атрибути;

– реалізувати програмну логіку роботи системи, використовуючи середовище VS Code;

– розрахувати показники стійкості системи.

При розробці системи варто врахувати наступні ключові аспекти: розподілення профілів та «ролей» користувачів в системі; створення і редагувати тестуваннь, а також перевірка їх результатів; автоматизовані процеси перевірки знань (тестування); інструменти обліку та перегляду навчальної статистики; прогнозування навчальних результатів на основі минулих оцінок; простота, зручність та інтуїтивна зрозумілість інтерфейсу.

Робота виконана згідно [1-2]. Результати досліджень за темою роботи опубліковані в [3-4].

1 АНАЛІЗ ІСНУЮЧИХ СИСТЕМ УПРАВЛІННЯ НАВЧАННЯМ

1.1 Інтеграція автоматизованих технологій у навчальні процеси

У нинішній час остітні технології дуже стрімко розвиваються і для цього розвитку є характерною саме цифрова трансформація освіти.

Все частіше спостерігається перехід від звичних паперових форм ведення документації до автоматизованих систем. Наразі це не проста тенденція, а нагальна потреба для підвищення об'єктивності оцінювання, ефективності, та прозорості навчального процесу.

Звичні методи контролю знань, які базуються на ручній перевірці успішності та виконанні робіт у зошитах, журналах, на паперових листах мають безліч суттєвих недоліків:

– великі витрати часу;

– суб'єктивність оцінювання;

– складність аналізу та ведення статистики;

– проблеми з доступом [5].

Викладачі витрачають дуже багато кількість часу створення завдань та різних варіантів для них, перевірку цих завдань та саму оцінку виконання. При перевірці яких-небудь творчих чи індивідуальних завдань може виникнути проблема необ'єктивності оцінювання подібних завдань з боку викладача.

При використанні традиційних методів контролю та оцінки знань немає можливості збирати статистичні дані, прогнозувати чи аналізувати динаміку навчальних досягнень окремого студента чи групи студентів за який-небудь період часу чи за конкретною темою.

Також, однією з проблем є те, що студенти не мають повного доступу до своїх оцінок чи навчальних результатів, не можуть бачити загалом кортину своєї успішності, аби отримати додаткову мотивацію та, наприклад,

сконцентрувати свою увагу на тій чи іншій темі.

Якщо автоматизувати перелічені вище процеси, це допоможе вирішити багато питань і проблем притаманних традиційним методам перевірки та контролю успішності навчання [5].

Таким чином, розробка автоматизованої системи обліку та аналізу поточної успішності студентів є актуальною задачею, що дозволить оптимізувати навчальний процес, дасть додаткові можливості для прогнозування успішності навчання та дозволить скоротити час на перевірку та створення тестів.

1.2 Сучасні методи контролю та оцінки успішності

Для ефективного моніторингу знань в автоматизованих системах можна використовувати різні методи контролю. Такі методи умовно поділяють на підсумкові та формуючі.

Метод формуючого оцінювання (Formative Assessment) ставить за мету не скільки просте виставлення фінальної оцінки, а скільки надання студентові та викладачеві зворотного зв'язку в процесі навчання, що дозволяє вчасно звертати на важливі фактори чи теми в процесі навчання і коригувати навчальний процес. В автоматизованій системі це можна реалізувати, наприклад, за допомогою коротких тестів після кожної пройденої теми чи заняття, або за допомогою інтерактивних завданнь, чи опитувань для перевірки розуміння студентами найголовніших моментів, яким слід привернути особливу увагу [6].

Другим методом контролю є метод підсумкового оцінювання (Summative Assessment). Цей метод використовується здебільшого для оцінки результатів навчання після проходження певного блоку тем, курсу чи семестру. Сюди можна віднести контрольні, самостійні роботи та інші іспити.

В автоматизованих системах найпоширенішим інструментом в

контексті обох типів оцінювання є комп'ютерна форма з тестуваннями. Такий інструмент дозволяє використовувати різноманітні формати питань, що перевірятимуться системою автоматично:

– класичним форматом опитування, найпростішим в реалізації є вибір питання з вибором однієї правильної відповіді (Multiple Choice);

– більш ускладненим варіантом опитування, що перевіряє більш глибоке знання теми та потребує більше часу на надання відповіді є вибір кількох правильних відповідей (Multiple Response);

– введення короткої відповіді (Fill-in-the-Blank) є варіантом опитування, де студент в якості відповіді на питання вводить слово, число чи символ, а система порівнює введену студентом відповідь з еталонним значенням, або вчитель перевіряє правильність відповіді вручну;

– встановлення відповідності (Matching) є типом опитування, в якому студент має співставити елементи у відповідності один одному;

– питання на порядок (Ordering) є типом опитування, в якому студент має розташовати елементи у правильній послідовності [6].

Окрім цього, сучасні методи контролю та оцінки успішності також прагнуть автоматизувати запис навчальних результатів в електронний журнал оцінок. Для цього система може надавати вчителю зручний інструмент для оцінювання навчальних робіт, чи записувати оцінки у журнал автоматично (якщо результати тестувань перевіряються системою). А студентам, в свою чергу, важливо надати можливість переглядати результати своєї успішності навчання. Система має надавати вільний доступ до своєї навчальної статистики, студенти повинні мати змогу бачити свої оцінки, сортувати їх за темами, навчальними дисциплінами, навчальними модулями, тощо.

1.3 Аналіз існуючих систем управління навчанням (LMS)

Задачі, що поставлені для виконання даної роботи, частково або повністю вирішуються класом програмних продуктів, що називаються Learning Management Systems (LMS), або іншими словами «системи управління навчанням». Вони представяєть з себе комплексні платформи для організації дистанційного, або змішаного навчання. Ці платформи призначені для адміністрування, управління, відстеження та звітності шкільного, університетського чи корпоративного навчання [7].

Ключовими перевагами використання систем управління навчанням є:

– централізоване навчання, усі матеріали, курси та дані клристувачів зберігаються в одному місці;

– заощадження матеріальних та часових витрат;

– гнучкість доступу до інформаці, учасники навчального процесу можуть навчатись будь-де віддалено, використовуючи різні пристрої;

– простота оновлення та додаванння контенту, при оновленні файлів на сервері усі користувачі одночасно побачать зміни;

– аналітика навчання, результатів, помилок, витрат часу, тощо.

Далі розглянуто елементи систем управління навчанням, їх призначення та особливості.

1.3.1 Компоненти LMS-систем та варіанти реалізації

Система управління навчанням є складним в реалізації комплексом програм, архітектура такої системи має забезпечувати зручне та гнучке, використання для всіх учасників освітнього процесу. Аби подібна система могла ефективно функціонувати, при її розробці необхідно правильно обрати та інтегрувати ключові компоненти [8].

На рис. 1.1 показано елементи системи управління навчанням (LMS).

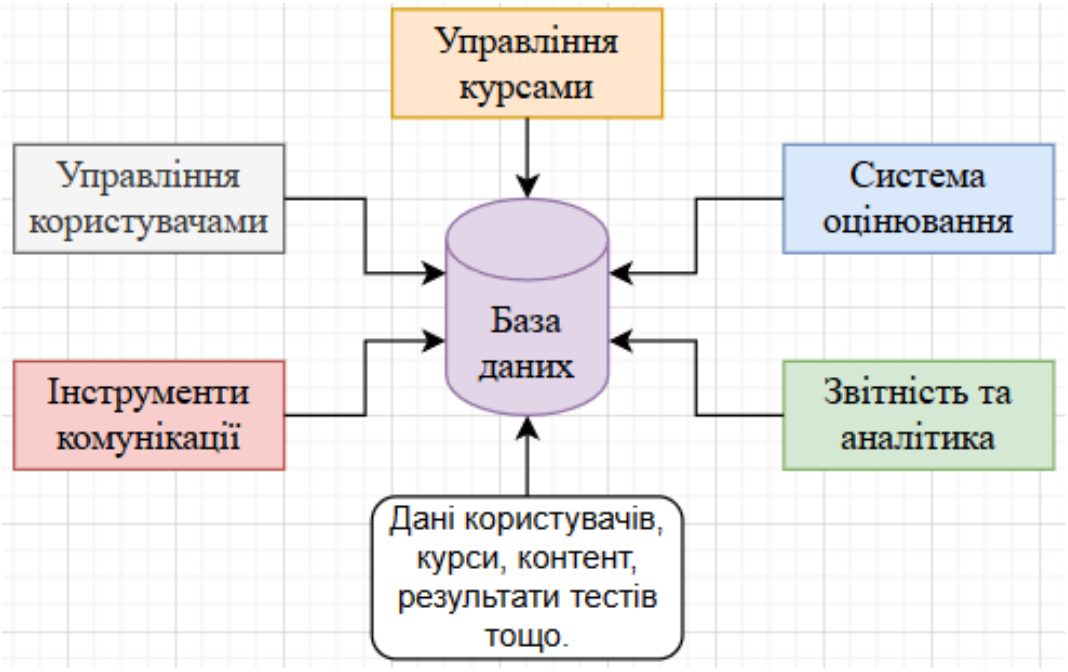

Рисунок 1.1 – Елементи системи управління навчанням (LMS) [8]

Елементи, що показані (рис. 1.1) демонструють стандартний «набір» складових більшості сучасних LMS.

База даних є центральною складової цієї системи, там зберігаються дані користувачів, курси, контент, тестові питання, оцінки тощо.

Елемент «Управління курсами» надає викладачам (авторам контенту) можливість створювати курси, запрошувати туди студентів а також створювати та додавати матеріали, контент, оголошення тощо.

Елемент «Система оцінювання» відповідає за перевірку успішності студентів у ході навчання. Сюди відносяться інструменти для створення тестувань, форм для опитувань та відстежування результатів навчання.

Інструменти комунікації включають різноманітні навчальні форуми, чати, або системи оповіщень учасників навчального процесу.

До звітності та аналітики можна віднести елементи статистики навчання, різноманітні звіти та дашборди.

Блок «Управління користувачами» відповідає за процес реєстрації чи авторизації в системі, розподілення ролей користувачів та їхні профілі. Зазвичай у LMS є дві ролі користувачів (студент та викладач). Студент має

доступ до інформації створеної чи доданої викладачем, наприклад тести, оцінки, учбові матеріали, тощо. Важливо зазначити, що студенти (учні) мають доступ лише до того контенту, доступ до якого їм надав викладач, або адміністратор системи. Викладач, відповідно має можливості створювати, додавати чи редагувати контент, виставляти оцінки, переглядати успішність учнів, підпорядкованим його навчальній дисципліні чи курсу.

1.4 Аналіз сучасних LMS-рішень

Далі розглянуто декілька розповсюжених та найуспішніших на сьогоднішній день прикладів реалізації LMS-систем, виділено їх переваги та недоліки в контексті автоматизації навчального процесу, виділено основні фактори, що вимагають сучасних рішень.

1.4.1 Moodle (Modular Object-Oriented Dynamic Learning Environment)

Moodle є однією з найпоширеніших у світі LMS з відкритим вихідним кодом. Вона широко використовується в школах, університетах та у різних компаніях чи організаціях, зокрема у секторі корпоративного навчання [9].

На рисунку 1.2 представлено вигляд LMS Moodle.

До переваг Moodle можна віднести:

– модуль тестування, який підтримує різноманітні типи питань;

– гнучкий журнал оцінок, який дозволяє створювати складні формули розрахунку підсумкової оцінки, вагові коефіцієнти для різних завдань;

– велика кількість плагінів та модулів;

– Moodle є Open Source проєктом і є безкоштовним.

До недоліків Moodle можна віднести:

– складність адміністрування, що вимагає кваліфікованого персоналу для встановлення, налаштування та підтримки серверів;

– недостатньо інтуїтивний інтерфейс, який для студентів та викладачів може здаватись складним та перевантаженим.

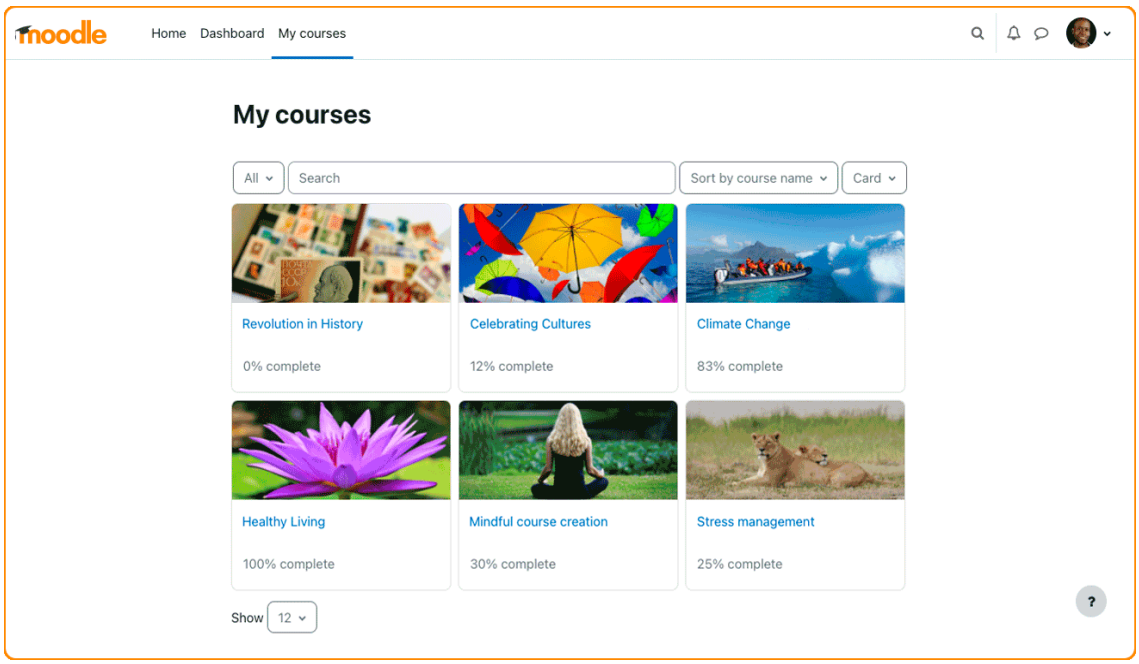

Рисунок 1.2 – Зовнішній вигляд LMS Moodle [9]

1.4.2 Google Classroom (Google Forms)

Google Classroom є безкоштовним хмарним сервісом сервісом, що входить до екосистеми Google Workspace for Education, який сам по собі не є повноцінною LMS, а скоріше його можна віднести до однієї зі складових частин Google Workspace, яка є пов'язуючим ланцюгом інших програм (Google Drive, Google Docs, Google Forms, Google Sheets та Google Slides).

Google Forms дуже розповсюджений у багатьох навчальних закладах різних рівнів. Таку популярність він отримав завдяки можливості швидко розповсюджувати та аналізувати інформацію, створювати опитування, тести, анкети та форми для збору інформації [10].

Головною особливістю Google Forms є зручний конструктор опитувань з великою кількістю налаштувань, де кожне питання можна налаштовувати окремо, також присутня можливість завантаження файлів.

На рисунку 1.3 представлено вікно створення питань для нового тестування у Google Forms.

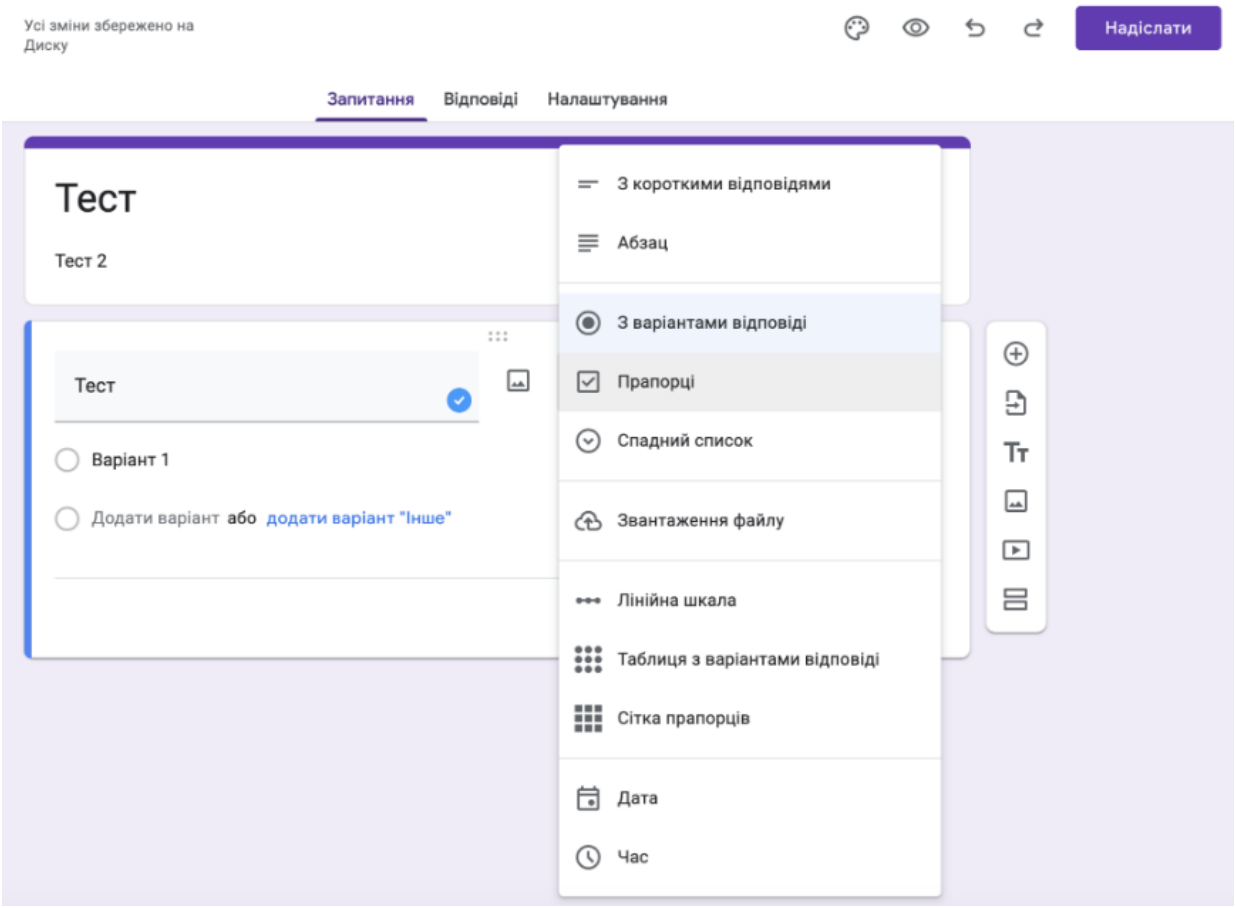

Рисунок 1.3 – Вікно створення питань для тесту у Google Forms [10]

До переваг Google Forms можна віднести:

– безкоштовне використання сервісу для закладів освіти чи особистого використання;

– немає обмежень щодо кількості створених форм для опитувань чи самих питань та відповідей на них;

– простий та зрозумілий у використанні інтерфейс (конструктор форм);

– не вимагає власного сервера.

До недоліків Google Forms можна віднести:

– обмеженість функціонал тестувань і обмежені налаштування при їх створенні (наприклад: час проходження опитування, варіативність форми відповідей);

– «примітивний» журнал оцінок, що по-суті є просто списком оцінок, без будь-якої аналітики чи можливостей розрахунку та прогнозування успішності;

– залежність від екосистеми та інших програм компанії Google.

Аналіз показав, що існуючі рішення систем управління навчанням часто є компромісними. Повноцінні LMS, такі як Moodle є надто складними в реалізації та потребують багато ресурсів для підтримки та налаштування. Більш прстим варіантом реалізації є системи типу Google Classroom, проте вони не надають достатньої гнучкості в налаштуванні тестів та аналітиці і прогнозуванні успішності, а також залежні від інших програм компанії Google.

Спеціалізовані платформи хоч і є локалізованими під систему світи конкретної держави та конкретну мову навчання, проте мають свої обмеження в налащтуваннях, недостатню гнучкість функціоналу та не у повній мірі «доступними».

1.5 Висновки до розділу

Проаналізувавши існуючі системи управління навчанням, сучасні методи контролю та оцінки успішності, а також системи управління навчанням LMS, можна дійти висновку про відсутність простого та водночас функціонального інструменту, який би дозволяв викладачам оперативно створювати тестування та керувати ними, а студентам проходити їх та відстежувати свою успішність в єдиному, інтуїтивно зрозумілому середовищі.

Оскільки LMS-системи є сучасними та розвинутими платформами для змішаного та онлайн-навчання, було прийнято рішення, узявши за основу їх переваги, створити простий, доступний та ефективний інструмент контролю успішності навчання.

Основний акцент при розробці варто зробити двох основних процесах, а саме тестуваннях та оцінюванні знань. Вчителі повинні мати зручні інструменти для створення, редагування та налаштування тестувань, функціонал «конструктора тестів» має включати налаштування балів, часу на

проходження, виду питання (відкрите, з одніє відповідю, з кількома відповідями, тощо). Учні в свою чергу повинні мати повний доступ до результатів свого навчання «електронного щоденника» і його функціонал, зокрема, повинен включати елементи аналітики і статистики (сортування оцінок, середній бал, загальна оцінка, прогнозування, тощо) [11].

Все це дозволить створити зручне навчальне середовище, що автоматизуватиме основні процеси перевірки успішності, яке підвищуватиме результативність та доступність навчання, дозолить скоротити час на перевірку завдань, а також надасть викладачам та учням можливості для аналізу та прогнозування перебігу навчального процесу на різних його етапах.

2 ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ ТА ВИБІР ПРОГРАМНИХ ЗАСОБІВ ДЛЯ ЇХ РЕАЛІЗАЦІЇ

Розробка автоматизованої системи обліку та аналізу поточної успішності студентів вимагає використання сучасних технологічних та програмних рішень. Розглянувши існуючі системи управління навчанням та визначивши основні їх переваги та недоліки, а також проаналізувавши сучасні потреби в автоматизації та цифровізації навчальних процесів можна сформувати основні задачі дослідження, що будуть розглянуті далі.

2.1 Постановка задач, які має вирішувати система

Розроблювана автоматизована система обліку та аналізу поточної успішності студентів має вирішувати, насамперед задачу автоматизації процесу перевірки та оцінювання знань. Для цього, насамперед при розробці мають бути враховані наступні елементи системи:

– авторизація в системі та розподілення ролей користувачів (студент, викладач), також має бути окремі інструменти для адміністратора системи для управління користувачами;

– забезпечення інструментів створення, налаштування та проведення тестувань успішності;

– автоматична (або ручна, яко це необхідно) перевірка результатів тестувань та запис їх до журналу оцінок;

– зручний та вичерпний журнал оцінок, повний доступ до історії та статистики навчання;

– елементи прогнозування майбутніх результатів навчання для вчасного втручання, або коригування перебігу навчання чи додаткової мотивації [11].

Вчителі та студенти повинні мати окремі «ролі» в ситемі. Це необхідно

для забезпечення процесу авторизації та входу до системи, кожен учасник системи має свій «аккаунт» зі своїми даними, статистикою, тощо [11].

Викладачі повинні мати зручні та багатофункціональні інструменти для створення тестувань та перевірки їх результатів. Проте, вчителі можуть створювати тестування та бачити їх результатів лише для тих предметів, викладачами з яких вони є.

Інтерфейс викладача має включати інформацію про вчителя (ПІБ, предмет, тощо), окремий розділ для створення та налаштування тестувань, а також перегляду та ручної перевірки «відповідей» і оцінки результатів учнів учнів з тієї дисципліни, яку він викладає. Також має бути створено окремий розділ для перегляду успішності студентів за цією дисципліною, де зібрані усі результати та уся статистика за предметом викладача. Викладач може обрати необхідну групу чи клас учнів і переглянути результати окремого учня за конкретним тестом чи темою (оцінки, середній бал за темою чи предметом, прогнозування оцінок для майбутніх тестувань).

Студент так само повинен мати свій розділ з доступними лише йому тестуваннями і окремий розділ з оцінками та навчальною статистикою, як це реалізовано для вчителя, але студент може бачити виключно свою навчальну статистику та результати. Для цього студенти розподілені за навчальними класами чи групами, а вчителі при створенні тестування вказують для якого класу чи групи студентів це тестування призначене. Таким чином коли студент обирає та проходить доступне для нього тестування, то в залежності від тестування оцінка за нього автоматично підраховується і записується в журнал, або ж вчитель додатково самостійно перевіряє деякі питання (якщо це потрібно) і редагує оцінку. Після чого оцінка записується до журналу і підраховується напвчальна статистика для даного студента. Таким чином зберегаються та формуються правила доступу до контенту та функціоналу в системі.

Всі інформація про користувачів (логіни, паролі, персональні дані, тощо), а також інформація щодо створених тестувань (назва тесту, предмет,

тема, текст питання, тип питання, варіанти відповідей, тощо) і результати успішності (пройдені студентами тести, результати цих тестів, відповіді на питання, тощо) зберігаються в базі даних системи і оновлюються та стають доступними одразу після будь-яких змін чи оновлень.

Таким чином система виконуватиме задачу автоматизації головних навчальних процесів, а саме задачі автоматизованого контролю та обліку навчальних досягненью. Це призведе до скорочення часу на виконання цих процесів та надасть усім учасникам навчального процесу повну картину перебігу успішності та прогресу отримання знань.

2.2 Вибір та обґрунтування програмних засобів для реалізації поставлених задач

2.2.1 Програмні засоби для запуску системи та зберігання даних

Для розробки та запуску системи буде використано пакет програм для управління та запуску локального веб-серверу MAMP. Цей пакет програмного забезпечення призначений для локального тестування веб-додатків та сайтів без потреби хостингу. Основними перевагами даного пакету програм є простота встановлення та безкоштовність використання. Окрім того, є можливість вільно використовувати різні версії PHP, що позитивно впливає на сумісність коду з більш старими, або більш новими стандартами. Головною перевагою є наявність вбудованих інструментів створення, управління та адміністрування базами даних.

MAMP складається з наступних програм та засобів:

– операційна система, на якій працює пакет програм MAMP (у даному випадку Windows);

– Apache, веб-сервер обробки HTTP-запити та відкриває веб-сторінки;

– MySQL, система управління базами даних, де зберігаються дані веб-сайтів та здійснюється адміністрування БД;

– PHP, є мовою програмування для створення динамічних веб-сайтів.

MAMP є ізольованим середовищем, це означає, що після встановлення створюється окрема коренева папка веб-серверу (htdocs). Всі файли розроблюваного веб-додатку (HTML, PHP, різні зображення, тощо) у кореневій папці після запуску локального серверу, стають доступними для запуску та перегляду в веб-браузері. Коли користувач вводить у браузері адресу для запуску локального серверу чи відкриває початкову сторінку сайту (зазвичай це index.html), то веб-сервер Apache отримує цей запит і проводить пошук відповідних файлів у кореневій папці [12].

PHP (Hypertext Preprocessor) є скриптовою мовою програмування. Що використовується при розробці веб-додатків і є найкращим рішенням для створення «динамічних веб-сайтів». MAMP обробляє PHP-скрипти для виконання логічних операцій, обчислень чи взаємодії з БД.

Для роботи з даними і зберігання інформації у пакеті програм MAMP використовується база даних MySQL, керування якою відбувається через зручний веб-інтерфейс phpMyAdmin. Цей засіб дозволяє створювати бази даних, виконувати SQL-запити до БД та налаштовувати права доступу. SQL є мовою запитів до бази даних, за допомогою SQL-запитів можна виконувати будь-які операції з даними (додавання, видалення, сортування за різними атрибутами, тощо) [13]. Тому використання MySQL в якості бази даних та phpMyAdmin для адміністрування та управління БД є найкращим рішенням у випадку розроблюваної системи, оскільки вирішує одразу кілька задач, таких як створення, адміністрування та взаємодія з БД.

2.2.2 Програмні засоби для створення файлів та коду програми

Для написання програмного коду та створення файлів програми використано програмне середовище Visual Studio Code (VS Code). Він є сучасним редактором коду від компанії Microsoft, що користується великою популярністю серед розробників.

Перевагами VS Code можна назвати:

– величезна кількість розширень;

– підтримка майже будь-яких мов програмування;

– зручний інтерфейс та інструменти роботи з кодом;

– величезна кількість розширень;

– вбудований дебагер, що дозволяє покроковий запуск коду;

– підтримка різних платформ та операційних систем;

– вбудований термінал.

Вибір Visual Studio Code в якості основного середовища розробки обгрунтований його універсальністю, зручним інтерфейсом та наявністю всіх необхідних інструментів для ефективного виконання поставлених задач розробки.

Для створення розмітки, дизайну і зовнішнього вигляду веб-сторінок системи (інтерфейсу користувача) буде використано HTML та CSS.

HTML є стандартизованою мовою розмітки, вона є основою будь-якої веб-сторінки і визначає її зміст та структуру.

CSS або Cascading Style Sheets є мовою опису зовнішнього вигляду документу, зміст якого написаний мовою розмітки HTML.

Таким чином створюється зовнішній вигляд інтерфейсу користувача системи (у випадку даної системи – студента або викладача).

2.3 Прогнозування успішності за допомогою лінійної регресії

Як вже було зазначено раніше, розроблювана система повинна включати елементи навчальної статистики та прогнозування майбутньої успішності навчання. Для цього було б доречним реалізувати такий елемент статистики навчання, як прогнозування оцінки студента на наступний тест за предметом або наступний тест за конкретною темою з цього предмету. Це позитивно вплине на мотивацію студентів щодо коригування своєї успішності по ходу навчання, дозволить бачити тенденцію поточних оцінок та дасть змогу приблизно бачити загальну «картину» навчання і майбутню оцінку. Для вчителів в свою чергу це відкриває додаткові можливості

оцінити «слабкі» та «сильні» теми у навчанні студента та дасть більше розуміння та інформації щодо його реального рівня знань та майбутньої оцінки.

Щоб прогнозувати майбутні оцінки у даній системі буде використовуватись алгоритм лінійної регресії. Лінійна регресія є статистичним методом для моделювання взаємозв'язку між залежною змінною (у випадку розроблюваної системи – оцінкою) та однією чи кількома незалежними змінними (у випадку розроблюваної системи – тестами).

На рисунку 2.1 показано типовий графік лінійної регресії з однією незалежною змінною.

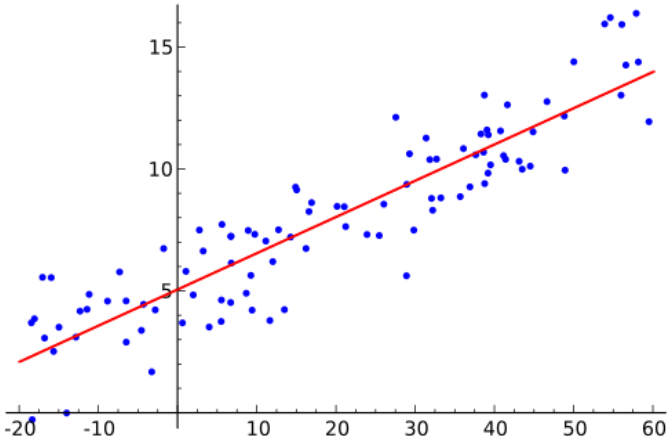

Рисунок 2.1 – Типовий графік лінійної регресії з однією незалежною змінною [14]

На графіку (рис. 2.1) вісь X (по горизонталі) відповідає за параметр який змінюється. Вісь Y (по вертикалі) відповідає за результат вимірювань, тобто це залежна змінна. Множина синіх точок на графіку лінійної регресії є результатами, вони показують залежність мі віссю X та віссю Y. Червона лінія на графіку називається лінією тренду, вона намагається пройти максимально близько до усіх точок одночасно, мінімізуючи відстань. Мета лінії тренду показати загальну закономірність результатів [14].

Математична формула лінійної регресії:

$$y = mx + b. \tag{4.1}$$

Де y – прогнозована оцінка (залежна змінна), x – номер наступного тесту (незалежна змінна), m – коефіцієнт нахилу, який показує динаміку оцінок (на скільки в середньому змінюється оцінка за один тест), b – це вільний член, або точка перетину з віссю Y (теоретична стартова оцінка) [15].

Переваги методу лінійної регресії для прогнозування результатів майбутніх тестувань:

– проста інтеграція в систему, реалізація за допомогою простих функцій у PHP, не вимагає особливої підтримки коду;

– висока швидкість обчислень, адже використовуються прості арифметичні формули;

– відсутність навантаження на сервер, навіть при великій кількості одночасних користувачів, на відміну від нейронних мереж, інтеграція яких потребує значних ресурсів;

– ефективність на малих вибірках, що доречно в контексті прогнозування оцінки за конкретну тему, де зазвичай не буває більше 10 тестів;

– здатність побудувати лінію тренду навіть на невеликій кількості даних, що дає змогу прогнозувати успішність вже на основі трьох оцінок.

Недоліком використання такого методу при прогнозуванні майбутніх результатів навчання можна назвати лише неможливість врахувати складність теми або тесту при прогнозуванні оцінки.

У випадку розроблюваної системи, аби мінімізувати помилку краще використовувати метод найменших квадратів при обчисленні.

Формула для знаходження нахилу m [16]:

$$m = \frac{n(\sum xy) - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2}. \quad (4.2)$$

Формула (4.2) призначена для знаходження кута нахилу лінії тренду за допомогою методу найменших квадратів, це дозволяє визначити наскільки різко зростають або спадають оцінки.

Формула для знаходження перетину *b* [15]:

$$b = \frac{\sum y - m(\sum x)}{n}. \quad (4.3)$$

Формула (4.3) призначена для знаходження знаходження перетину *b* з віссю Y. У даній формулі n – це кількість пройдених тестів. В контексті розроблюваної системи це «базова» оцінка від якої відштовхується лінія тренду. Іншими словами це теоретична оцінка, яка припускається системою, щоб лінія тренду проходила максимально близько до усіх інших точок.

2.4 Висновки до розділу

У даному розділі було визначено основні принципи на яких має базуватись розроблювана система, поставлено визначено задачі, які вона має розв’язувати. Описано принципи надання доступу до функціоналу системи, визначено ролі користувачів та їх можливості, а також описано основні принципи функціонування системи, та обгрунтовано даний вибір.

Далі було описано та обгрунтовано вибір програмного пакету для запуску системи. Було визначено, що обраний набір програм MAMP чудово підходить для виконання поставлених задач реалізації, адже містить усі необхідні інструменти, зокрема мову для виконання скриптів PHP, а також БД MySQL та інструменти управління та адміністрування нею phpMyAdmin. Для створення файлів та написання програмного коду буде використовуватись середовище VS Code, його вибір обгрунтований підтримкою безлічі мов програмування, доступністю та зручністю.

Також, засобом для реалізації прогнозування майбутньої успішності було обрано алгоритм лінійної регресії. Було описано переваги, недоліки та особливості реалізації даного методу в контексті створюваної системи[15].

3 СТВОРЕННЯ БАЗИ ДАНИХ, ОПИС ЗВ’ЯЗКІВ ТА ТАБЛИЦЬ ДАНИХ

При створенні бази даних для розробленої системи було використано БД MySQL. Для проектування БД застосовано метод ER-моделювання, далі описано принципи цього методу моделювання баз даних, а також показано та описано ER-модель створеної для виконання поставлених задач БД, описано її таблиці та зв’язки між сутностями.

3.1 Створення ER-моделі БД

ER-модель (Entity-Relationship Model), або модель «сутність-зв’язок» називають схему, що показує структуру бази даних. Вона показує, таблиці бази даних та їх зв’язки між собою.

Основними елементами ER-моделі є сутності та атрибути (entities and attributes). Сутностями є таблиці розроблюваної бази даних (користувачі, предмети, тести, тощо), а атрибутами в свою чергу є колонки вищезгаданих таблиць (id, name, password, тощо). Зв’язки між сутностями відображаються лініями на схемі ER-моделей і показують як дані з однієї таблиці БД відносяться до даних з іншої таблиці, такі зв’язки в ER-моделюванні реалізуються за допомогою зовнішніх ключів (foreign keys) [17].

Зв’язки в ER-моделях бувають декількох типів:

– один до одного (1:1);

– один до багатьох (1:N);

– багато до багатьох (M:N).

Зв’язок один до одного означає, що лише один запис у одній таблиці пов'язаний лише з одним записом в іншій таблиці, або навпаки. Зв'язок один до багатьох є найпоширенішим видом зв'язку, коли запис в одній таблиці пов'язаний з багатьма записами в другій таблиці, але навпаки бути не може, тобто запис у другій таблиці пов'язаний лише з одним записом першої

таблиці. Тип зв'язку багато до багатьох означає, що запис однієї таблиці може бути пов'язаний з багатьма записами іншої таблиці, і так само може бути навпаки [17].

На рисунку 3.1 показно створену ER-модель БД розроблюваної системи.

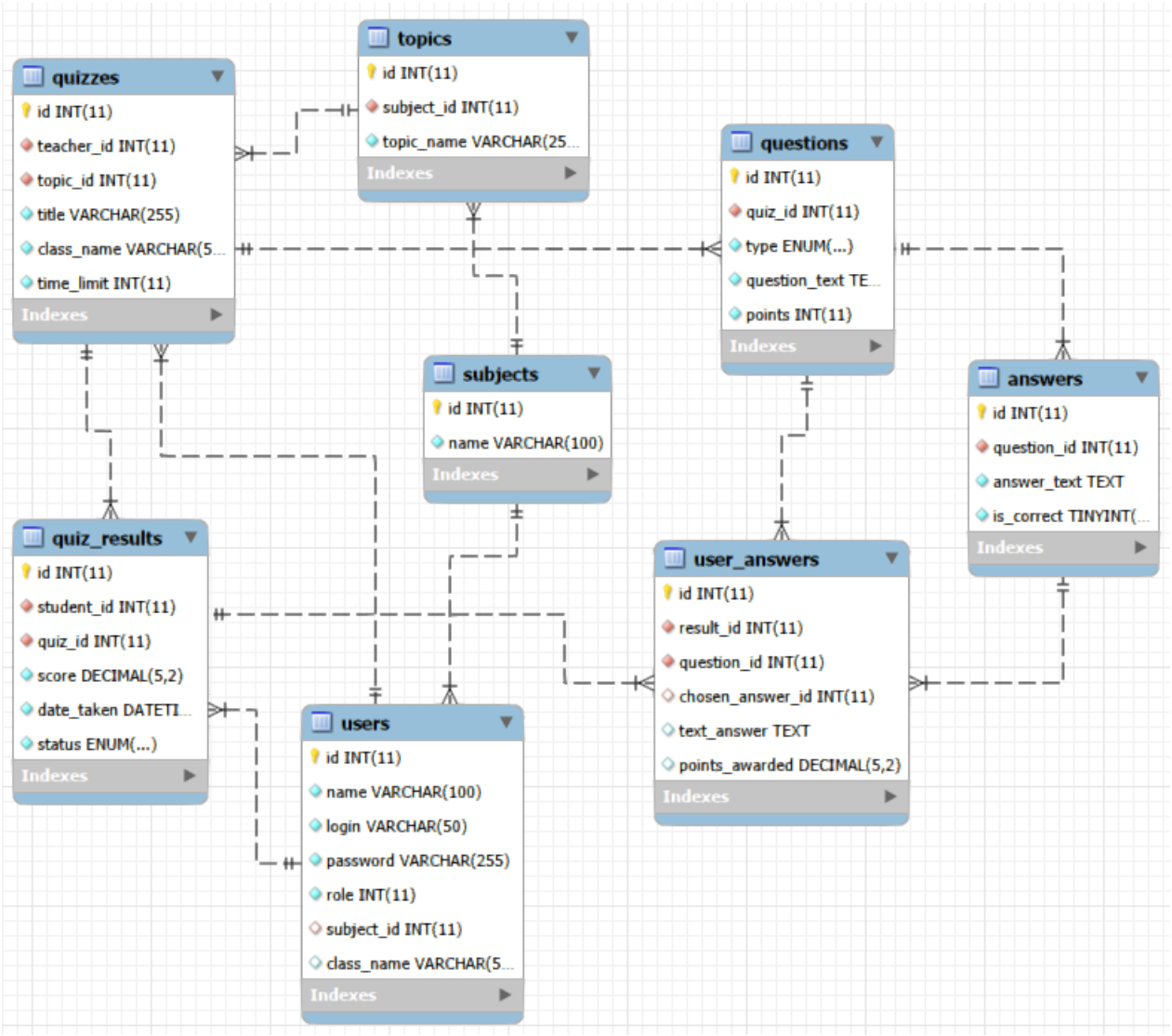

Рисунок 3.1 – ER-модель створеної бази даних

3.2 Опис таблиць та зв’язків БД

Для опису сутностей створеної бази даних та їх атрибутів було б зручно представити їх параметри у вигляді таблиці. Нижче представлені

таблиці з характеристиками та параметрами усіх таблиць створеної БД.

Таблиця «users» зберігає дані користувачів та є головною сутністю для авторизації в системі. Вона має зв’язок (N:1) з таблицею «subjects», що дає змогу закріпити за вчителем певний предмет, який він викладає. Таблиця «users» (табл. 3.1) також є батьківською таблицею для таблиць «quizzes» та «quiz_results», зв’язки (1:N).

Таблиця 3.1 – Опис таблиці «users»

Атрибут	Тип даних	Розмір даних	Атрибути	NULL	Додатково
id	int	11	UNSIGNED	Ні	AUTO_INCREMENT (Primary key)
name	varchar	100	–	Ні	ПІБ
login	varchar	50	–	Ні	Логін користувача
password	varchar	255	–	Ні	Пароль, або хеш
role	int	11	–	Ні	1 = викладач, 2 = студент, 3 = адміністратор
subject_id	int	11	–	Так	(Foreign key)
class_name	varchar	50	–	Так	Назва предмету (для викладача), назва групи (для студента)

Таблиця «subjects» (табл. 3.2) відповідає за навчальні дисципліни та зберігає дані про них. Має зв’язок (1:N) з таблицею «topics» (один предмет може мати багато тем), та зв’язок (1:N) з таблицею «users».

Таблиця 3.2 – Опис таблиці «subjects»

Атрибут	Тип даних	Розмір даних	Атрибути	NULL	Додатково
id	int	11	UNSIGNED	Hi	AUTO_INCREMENT (Primary key)
name	varchar	100	–	Hi	Назва предмету

Таблиця «topics» (табл. 3.3) необхідна для того щоб розділяти предмети на конкретні теми. Вона має зв’язок (1:N) з таблицею «subjects». Та виступає як окрема категорія для тестувань, тому має зв’язок (1:N) з таблицею «quizzes».

Таблиця 3.3 – Опис таблиці «topics»

Атрибут	Тип даних	Розмір даних	Атрибути	NULL	Додатково
id	int	11	UNSIGNED	Hi	AUTO_INCREMENT (Primary key)
subject_id	int	11	–	Hi	(Foreign key subjects_id)
topic_name	varchar	255	–	Hi	Назва теми

Таблиця «quizzes» (табл. 3.4) необхідна для збереження налаштувань тестів (час, назва, для якого класу призначений). Має зв’язок (N:1) з таблицею «users», зв’язок (N:1) з таблицею «topics» та зв’язок (1:N) з таблицею «quiz_results», а також зв’язок (1:N) з таблицею «questions».

Таблиця 3.4 – Опис таблиці «quizzes»

Атрибут	Тип даних	Розмір даних	Атрибути	NULL	Додатково
id	int	11	UNSIGNED	Hi	AUTO_INCREMENT (Primary key)
teacher_id	int	11	–	Hi	(Foreign key users_id)
topic_id	int	11	–	Hi	(Foreign key topics_id)
title	varchar	255	–	Hi	Назва тесту
class_name	varchar	50	–	Hi	Для якого класу
time_limit	int	11	–	Hi	Час тесту (у хв.)

Таблиця «questions» (табл. 3.5) необхідна для збереження тексту питань, кількості балів, та типу питання. Всього може бути 3 типи питань (single – питання з одним правильним варіантом відповіді, multiple – амтання з декількома правильними варіантами відповіді, та text – цк відкрите питання, відповідь на яке учень має ввести у відповідне текстове поле). Має зв’язок (1:N) з таблицею «answers», зв’язок (N:1) з таблицею «quizzes» та (1:N) з таблицею «user_answers».

Таблиця 3.5 – Опис таблиці «questions»

Атрибут	Тип даних	Розмір даних	Атрибути	NULL	Додатково
id	int	11	UNSIGNED	Hi	AUTO_INCREMENT (Primary key)
quiz_id	int	11	–	Hi	(Foreign key quizzes_id)
type	enum	–		Hi	Тип питання (single, multiple, або text)
question_text	text	–	–	Hi	Текст питання
point	int	11	–	Hi	Кількість балів

Таблиця «answers» (табл. 3.6) зберігає текстові відповіді на питання та маркери правильності чи неправильності відповідей. Має зв’язок (1:N) з таблицею «user_answers» та (N:1) з таблицею «questions».

Таблиця 3.6 – Опис таблиці «answers»

Атрибут	Тип даних	Розмір даних	Атрибути	NULL	Додатково
id	int	11	UNSIGNED	Ні	AUTO_INCREMENT (Primary key)
question_id	int	11	–	Ні	(Foreign key questions_id)
answer_text	text	–	–	Ні	Текст відповіді на питання
is_correct	tinyint	1	–	Ні	1 = вірно, 2 = невірно

Таблиця «quiz_results» (табл. 3.7) необхідна для запису проходження тесту студентом, збереження загальних балів та часу і дату проходження та статусу тесту чи питань. Має зв’язок (1:N) з таблицею «user_answers», зв’язок (N:1) з таблицею «quizzes» та (N:1) з таблицею «users».

Таблиця 3.7 – Опис таблиці «quiz_results»

Атрибут	Тип даних	Розмір даних	Атрибути	NULL	Додатково
id	int	11	UNSIGNED	Ні	AUTO_INCREMENT (Primary key)
student_id	int	11	–	Ні	(Foreign key users_id)
quiz_id	int	–	–	Ні	(Foreign key quizzes_id)
score	decimal	–	–	Ні	Загальний бал (%)

Продовження таблиці 3.7

Атрибут	Тип даних	Розмір даних	Атрибути	NULL	Додатково
date_taken	datetime	–	–	Ні	Дата проходження
status	enum	–	–	Ні	Завершено, або на перевірці

Таблиця «user_answers» (табл. 3.8) зберігає кожну окрему дію студента під час тестування. Має зв’язок (N:1) з таблицею «answers», (N:1) з таблицею «questions» та зв’язок (N:1) з таблицею «quiz_results».

Таблиця 3.8 – Опис таблиці «user_answers»

Атрибут	Тип даних	Розмір даних	Атрибути	NULL	Додатково
id	int	11	UNSIGNED	Ні	AUTO_INCREMENT (Primary key)
result_id	int	11	–	Ні	(Foreign key quiz_results_id)
question_id	int	11	–	Ні	(Foreign key questions_id)
chosen_answer_id	decimal	11	–	Так	(Foreign key answers_id)
text_answer	text	–	–	Так	Для текстових відповідей
points_awarded	decimal	–	–	Ні	Нараховані бали

3.3 Висновки до розділу

Для виконання поставлених задач при створенні системи було спроектовано та побудовано базу даних за допомогою середовища phpMyAdmin. При проектуванні було створено та описано ER-модель БД, на якій було визначено основні сутності та зв’язки між ними. Далі детально описано кожну таблицю сутностей та їх характеристики, а також зв’язки і залежності з іншими таблицями.

Створена база даних надає усі необхідні опції для виконання раніше поставлених задач. Сюди відноситься збереження даних користувачів, параметри створених тестувань, параметри створених питань, збережені відповіді, результати пройдених тестів, навчальні дисципліни, теми, тощо.

4 РЕАЛІЗАЦІЯ ПРОГРАМНОЇ ЛОГІКИ РОБОТИ СИСТЕМИ

4.1 Розробка блок-схеми програми

Для демонстрації логіки роботи розроблюваної системи, було створено блок-схему. Вона показує процеси, що відбуваються під час роботи програми та послідовність дій користувача системи при її використанні. На рисунку 4.1 показано блок-схему розроблюваної програми.

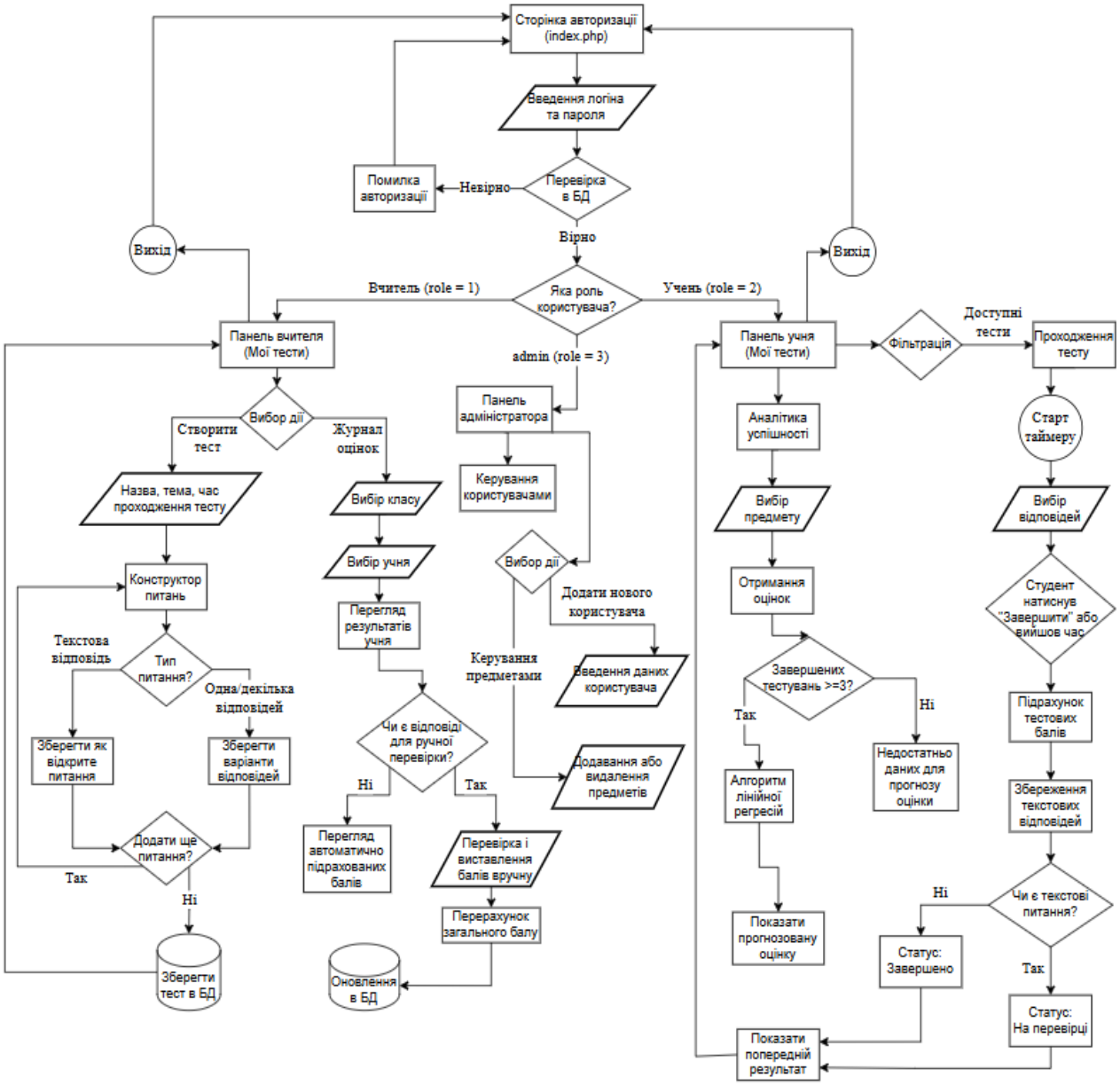

Рисунок 4.1 – Блок-схема розроблюваної програми [18]

Блок-схема (рис. 4.1) демонструє послідовність роботи системи, як можна побачити є два основних розподілення функціоналу, залежно від ролі у системі [18]. Цими ролями користувачів є «викладачі» та «студенти». «Адміністратор» є окремою ролею у системі. Він управляє іншими користувачами, може додавати, видаляти користувачів, а також редагувати їх дані (ПІБ, логін, пароль, класс, тощо), а також може додавати нові навчальні дисципліни чи видаляти старі.

4.2 Програмна реалізація компонентів системи

4.2.1 Реалізація під'єднання до БД та ініціалізація середовища

Даний етап є основою роботи системи. Це відбувається автоматично, коли завантажується будь-яка сторінка сайту (на початку кожного файлу вказано «include 'config.php';»). Код, що відповідає за процеси ініціалізації та під'єднання до бази даних знаходиться у файлі «config.php». Цей файл відповідає за початок сесії, аби система пам'ятала, хто є авторизованим у ній та за безпечне підключення до бази даних.

Код файлу «config.php»:

```
<?php
// Початок сесії на всіх сторінках
session_start();

// Налаштування Бази Даних
define('DB_HOST', 'localhost');
define('DB_USER', 'root'); // Логін
define('DB_PASS', 'root'); // Пароль
define('DB_NAME', 'student_success_db');

// Створення підключення
$db = new mysqli(DB_HOST, DB_USER, DB_PASS, DB_NAME);
```

```
// Перевірка підключення
if ($db->connect_error) {
    die("Помилка підключення до MySQL: " . $db->connect_error);
}

// Встановлення кодування
$db->set_charset("utf8mb4");

/**
 * Функція для перевірки авторизації та ролі
 * @param int $required_role 1=Вчитель, 2=Учень, 3=Адмін
 */
function check_auth($required_role) {
    // Якщо не авторизований або немає ролі в сесії
    if (!isset($_SESSION['user_id']) || !isset($_SESSION['role'])) {
        header('Location: index.php');
        exit;
    }

    // Якщо роль не співпадає з необхідною на цій сторінці
    if ($_SESSION['role'] != $required_role) {
        // Перенаправлення користувачів
        if ($_SESSION['role'] == 1) {
            header('Location: teacher_quizzes.php');
        } elseif ($_SESSION['role'] == 2) {
            header('Location: student_dashboard.php');
        } elseif ($_SESSION['role'] == 3) {
            header('Location: admin_dashboard.php');
        }
```

```
        exit;
    }
}
?>
```

Також варто зазначити, що використана функція «check_auth» запобігає тому, що користувач системи зайде на сторінку, до якої немає доступу. Наприклад, студент не зможе перейти до сторінку викладача, просто ввівши адресу відповідної сторінки в браузері [3].

4.2.2 Реалізація процесу авторизації в системі

При початку роботи з в системі користувач має пройти процес авторизації. Спочатку він потрапляє на сторінку авторизації, де має ввести свої авторизаційні дані (логін та пароль). Після перевірки авторизаційних даних користувача та хешу пароля, система розподіляє потоки користувачів залежно від їхньої ролі, для цього у БД системи у кожного користувача є параметр «role», який і відносить користувачів до конкретної ролі (role = 1 для вчителя, role = 2 для учня, role = 3 для адміністратора). Кожен користувач потрапляє на «стратову» сторінку залежно від своєї ролі в системі.

За вище перелічені процеси відповідає файл «login_handler.php»:

```
<?php
include 'config.php';

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $login = $_POST['login'];
    $password = $_POST['password'];

    // Отримуємо дані користувача
     $stmt = $db->prepare("SELECT id, name, password, role, subject_id FROM users WHERE login = ?");
```

```
$stmt->bind_param("s", $login);
$stmt->execute();
$result = $stmt->get_result();

if ($result->num_rows == 1) {
    $user = $result->fetch_assoc();

    // Перевірка хешу пароля
    if (password_verify($password, $user['password'])) {
        // Записуємо дані в сесію
        $_SESSION['user_id'] = $user['id'];
        $_SESSION['name'] = $user['name'];
        $_SESSION['role'] = $user['role'];
        $_SESSION['teacher_subject_id'] = $user['subject_id']; // Для вчителів

        // Перенаправлення користувачів на їх сторінки
        if ($user['role'] == 1) {
            header('Location: teacher_quizzes.php');
        } elseif ($user['role'] == 2) {
            header('Location: student_dashboard.php');
        } elseif ($user['role'] == 3) {
            // Панель адміна
            header('Location: admin_dashboard.php');
        }
        exit;
    }
}
// Якщо помилка
header('Location: index.php?error=1');
exit;
```

```
} else {
   header('Location: index.php');
   exit;
}
?>
```

Для виходу зі свого акаунту користувач натискає кнопку «Вийти» і знову перенаправляється на сторінку авторизації.

Це реалізовано у файлі «logout.php»:

```
<?php
include 'config.php';

// Закриваємо сесію
session_unset();
session_destroy();

// Перенаправлення на сторінку входу
header('Location: index.php');
exit;
?>
```

4.2.3 Реалізація панелі адміністратора

Для реалізації функціоналу адміністрування системою було реалізовано панель адміністратора. Адміністратор має контроль над іншими користувачами системи, він може видаляти користувачів, змінювати їх дані (ПІБ, логін, пароль, клас, дисципліна яку викладає), а також додавати нових користувачів системи. Для видалення даних користувачів біло реалізовано логіку каскадного видалення даних для збереження цілісності БД.

Керування користувачами реалізовано у головній сторінці адміністратора, файл «admin_dashboard.php»:

```
<?php
include ‘config.php’;
check_auth(3); // Доступ тільки для Адміністратора

// Логіка видалення
if (isset($_GET[‘delete_id’])) {
   $del_id = (int)$_GET[‘delete_id’];

   // не можна видалити самого себе
   if ($del_id == $_SESSION[‘user_id’]) {
            echo “<script>alert(‘Не можна видалити власний акаунт!’); location.href=’admin_dashboard.php’;</script>”;
      exit;
   }

   // Дізнаємось роль користувача перед видаленням
   $role_q = $db->query(“SELECT role FROM users WHERE id = $del_id”);

   if ($role_q->num_rows > 0) {
      $target_role = $role_q->fetch_assoc()[‘role’];

      // Очищення даних
      if ($target_role == 1) { // Якщо це Вчитель
         // Знаходимо всі його тести
         $quizzes = $db->query(“SELECT id FROM quizzes WHERE teacher_id = $del_id”);
         while ($q = $quizzes->fetch_assoc()) {
            $qid = $q[‘id’];

            // Видаляємо результати учнів за ці тести
```

```
            $db->query("DELETE FROM quiz_results WHERE quiz_id = $qid");

            $db->query("DELETE FROM questions WHERE quiz_id = $qid");

        }
        // Видалення тестів
        $db->query("DELETE FROM quizzes WHERE teacher_id = $del_id");

    } elseif ($target_role == 2) {
        // Для учня
        // Видаляємо загальні результати
            $db->query("DELETE FROM quiz_results WHERE student_id = $del_id");
        // Видаляємо детальні відповіді на питання
    }

    // Фінальне видалення користувача з системи
    if ($db->query("DELETE FROM users WHERE id = $del_id")) {
        header("Location: admin_dashboard.php");
        exit;
    } else {
        echo "Помилка видалення: " . $db->error;
    }
  }
}
```

Адміністратор також може керувати навчальними дисциплінами, що є в системі. Він може створювати нові предмети та видаляти вже існуючі дисципліни, якщо він видаляє існуючу дисципліну, то всі дані пов'язані з нею (тести, оцінки) видаляються.

Даний функціонал реалізовано у файлі «admin_subjects.php»:

```
<?php
include 'config.php';
check_auth(3); // Тільки для адміну
// Додавання предмету
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['new_subject'])) {
    $name = trim($_POST['subject_name']);
    if (!empty($name)) {
        // Перевірка чи такий вже є
        $check = $db->query("SELECT id FROM subjects WHERE name = '$name'");
        if ($check->num_rows == 0) {
            $stmt = $db->prepare("INSERT INTO subjects (name) VALUES (?)");
            $stmt->bind_param("s", $name);
            $stmt->execute();
            $stmt->close();
        } else {
            $error = "Такий предмет вже існує!";
        }
    }
}
// Видалення предмету
if (isset($_GET['delete_id'])) {
    $del_id = (int)$_GET['delete_id'];
    // Спочатку перевіряємо, чи не прив'язаний предмет до вчителів або тестів (це важливо для цілісності)
    $db->query("DELETE FROM subjects WHERE id = $del_id");
    header("Location: admin_subjects.php");
```

```
    exit;
}
```

Окрім того адміністратор може редагувати дані користувачів за необхідності. Є можливість змінювати ПІБ, логін, пароль користувача, присвоювати навчальний клас (групу) «студенту», або присвоювати дисципліну викладання «вчителю».

Це реалізовано у файлі «admin_edit_user.php»:

```
<?php
include 'config.php';
check_auth(3); // Доступ тільки для адміна

$id = isset($_GET['id']) ? (int)$_GET['id'] : 0;
$user = ['name'=>'', 'login'=>'', 'role'=>2, 'class_name'=>'', 'subject_id'=>0];
$is_edit = false;

// дістаємо дані для редагування
if ($id) {
    $is_edit = true;
    $stmt = $db->prepare("SELECT * FROM users WHERE id = ?");
    $stmt->bind_param("i", $id);
    $stmt->execute();
    $user = $stmt->get_result()->fetch_assoc();
    if (!$user) die("Користувача не знайдено");
}

// Отримуємо список предметів для форми
$subjects = $db->query("SELECT * FROM subjects ORDER BY name");

// Обробка форми
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
```

```
    $name = $_POST['name'];
    $login = $_POST['login'];
    $role = (int)$_POST['role'];
      // Якщо роль не учень, клас очищаємо. Якщо не вчитель, предмет очищаємо.
    $class_name = ($role == 2) ? $_POST['class_name'] : NULL;
    $subject_id = ($role == 1) ? (int)$_POST['subject_id'] : NULL;
    $password = $_POST['password'];

    if ($is_edit) {
        // Оновлення
            $sql = "UPDATE users SET name=?, login=?, role=?, class_name=?, subject_id=? WHERE id=?";
        $stmt = $db->prepare($sql);
         $stmt->bind_param("ssisi", $name, $login, $role, $class_name, $subject_id, $id);
        $stmt->execute();

        // Оновлення пароля тільки якщо введено новий
        if (!empty($password)) {
            $hash = password_hash($password, PASSWORD_DEFAULT);
            $db->query("UPDATE users SET password='$hash' WHERE id=$id");
        }
    } else {
        // Створення
          if (empty($password)) die("Помилка: Пароль обов'язковий для нового користувача.");
        $hash = password_hash($password, PASSWORD_DEFAULT);

          $sql = "INSERT INTO users (name, login, password, role, class_name,
```

```
subject_id) VALUES (?, ?, ?, ?, ?, ?)";
      $stmt = $db->prepare($sql);
         $stmt->bind_param("sssisi", $name, $login, $hash, $role, $class_name, $subject_id);
      $stmt->execute();
   }

   // Повернення до списку
   header("Location: admin_dashboard.php");
   exit;
}
?>
```

4.2.4 Реалізація процесу створення та редагування тестувань

Процес створення та редагування тестувань вчителем реалізовано як багатоетапну процедуру. Вона поєднує класичну форму створення (PHP) та динамічний інтерфейс редагування (JavaScript + AJAX).

AJAX-запити (Asynchronous JavaScript and XML) дозволяють веб-сторінці обмінюватись даними з БД у фоновому режимі, оновлюючи лише потрібну її частину, а не перезавантажувати для цього усю сторінку [19].

Спочатку, при ініціалізації «викладачем» створення нового тесту, створюється контейнер для тесту. Тут вказуються назва тесту, тема, обирається клас для якого цей тест призначено та встановлюється ліміт часу на проходження тесту.

У файлі «create_quiz.php» реалізовано створення тестувань:

```
<?php
include 'config.php';
check_auth(1);
$teacher_id = $_SESSION['user_id'];
$subject_id = $_SESSION['teacher_subject_id'];
```

```
// Перевірка, чи призначено вчителю предмет
if (!$subject_id) {
    die("Помилка: Вашому акаунту не призначено предмет. Зверніться до адміністратора.");
}

// Отримуємо назву предмету для відображення на сторінці
$sub_query = $db->query("SELECT name FROM subjects WHERE id = $subject_id");
$subject_name = $sub_query->fetch_assoc()['name'];

// Обробка форми
if ($_SERVER["REQUEST_METHOD"] == "POST") {
   $title = trim($_POST['title']);
   $class_name = $_POST['class_name'];
   $topic_name = trim($_POST['topic_name']);
   $time_limit = (int)$_POST['time_limit'];

   if (empty($topic_name) || empty($title)) {
      die("Будь ласка, заповніть всі поля.");
   }

   // Логіка роботи з темою
    $check_topic = $db->prepare("SELECT id FROM topics WHERE topic_name = ? AND subject_id = ?");
   $check_topic->bind_param("si", $topic_name, $subject_id);
   $check_topic->execute();
   $topic_res = $check_topic->get_result();
```

```
    if ($topic_res->num_rows > 0) {
        $topic_row = $topic_res->fetch_assoc();
        $topic_id = $topic_row['id'];
    } else {
        $insert_topic = $db->prepare("INSERT INTO topics (subject_id, topic_name)
VALUES (?, ?)");
        $insert_topic->bind_param("is", $subject_id, $topic_name);
        $insert_topic->execute();
        $topic_id = $db->insert_id;
        $insert_topic->close();
    }
    $check_topic->close();

    // Створення тестів
      $stmt = $db->prepare("INSERT INTO quizzes (teacher_id, topic_id, title,
class_name, time_limit) VALUES (?, ?, ?, ?, ?)");
       $stmt->bind_param("iisss", $teacher_id, $topic_id, $title, $class_name,
$time_limit);

    if ($stmt->execute()) {
        $new_quiz_id = $db->insert_id;
        header('Location: edit_quiz.php?id=' . $new_quiz_id);
        exit;
    } else {
        echo "Помилка створення тесту: " . $db->error;
    }
    $stmt->close();
}
?>
```

Після налаштувань конструктора тесту, «викладач» переходить до створення питань та варіантів відповідей. Тут для уникнення перезавантажень сторінки після створеня кожного питання, використовується технологя AJAX-запитів [19].

При створенні нового питання, вказується текст питання, кількість балів та тип питання. Всього реалізовано три типи питання (з однією відповідю, з кількома відповідями, текстова відповідь).

Питання «з однією відповідю» означає, що для отримання вказаного балу за відповідь на питання «студент» має обрати лише одну правильну відповідь з декількох, інакше він отримує 0 балів за це питання.

Питання «з кількома правильними відповідями» означає, що для отримання максимального балу за відповідь на це питання «студент» має обрати кілька правильних відповідей (які задані «викладачем» при створенні тесту). Якщо «студент» обрав не всі правильні відповіді, то система автоматично підраховує бал за це питання таким чином, що «вартість» одного правильного вибору рівна максимальній кількості балів за це питання, поділеній на кількість правильних відповідей. За кожну не правильно обрану відповідь віднімається така ж «вартість» відповіді, тобто якщо за питання дається 2 бали і треба отбрати 2 правильні відповіді, а «студент» обрав 2 правильні та 1 неправильну відповідь, то він отримає 1 бал + 1 бал (за правильні відповіді) – 1 бал (за неправильну відповідь) = 1 бал (з 2 можливих).

Відкрите питання, або «питання з текстовою відповіддю» означає, що «студент» має ввести відповідь (числом або текстом) у відповідне текстове поле. Система автоматично нараховує 0 балів за відповідь на це питання, а вже потім вчитель у вікні перегляду результатів «в ручну» перегялдає відповідь на це питання та оцінює його.

У файлі «edit_quiz.php» реалізовано створення та редагування питань тестувань:

<script>

```
const QUIZ_ID = <?php echo $quiz_id; ?>;
// Універсальна функція для відправки AJAX запитів
async function sendAjax(data) {
    try {
        const response = await fetch('ajax_handler.php', {
            method: 'POST',
            headers: { 'Content-Type': 'application/x-www-form-urlencoded' },
            body: new URLSearchParams(data)
        });
        return await response.text();
    } catch (error) { console.error(error); }
}

// Додавання питання
async function addQuestion() {
    const textInput = document.getElementById('new-question-text');
    const pointsInput = document.getElementById('new-question-points');
    const typeInput = document.getElementById('new-question-type');

    if (textInput.value.trim() === "") return;

    // Відправка даних на сервер
    const responseHtml = await sendAjax({
        action: 'add_question',
        quiz_id: QUIZ_ID,
        question_text: textInput.value.trim(),
        points: pointsInput.value,
        type: typeInput.value
    });
```

```
            // Додавання отриманого HTML в список
            if (responseHtml) {
                        document.getElementById('quiz-questions').innerHTML += responseHtml;
                textInput.value = "";
                pointsInput.value = "1";
            }
        }

        // Додавання відповіді
        async function addAnswer(event, questionId) {
            event.preventDefault();
            const form = event.target;
                                                const answerText = form.querySelector('input[name="answer_text"]').value.trim();
            const isCorrect = form.querySelector('input[name="is_correct"]').checked ? 1 : 0;

            // Отримання типу питання
            const questionBlock = document.getElementById('question-' + questionId);
            const qType = questionBlock.getAttribute('data-type');

            if (answerText === "") return;

            const responseHtml = await sendAjax({
                action: 'add_answer',
                question_id: questionId,
                answer_text: answerText,
                is_correct: isCorrect,
                question_type: qType
```

```
        });
        if (responseHtml) {
            // Якщо це питання з однією правильною відповідю і ми додали правильну, то знімаємо галочку з усіх інших варіантів відповіді
            if (qType === 'single' && isCorrect) {
                const allAnswers = document.querySelectorAll(`#answers-for-${questionId} li`);
                allAnswers.forEach(li => li.classList.remove('correct'));
            }
            document.getElementById(`answers-for-${questionId}`).innerHTML += responseHtml;
            form.reset();
        }
    }

    // Видалення
    async function deleteItem(type, id) {
        if (confirm('Видалити?')) {
            const response = await sendAjax({ action: (type === 'question' ? 'delete_question' : 'delete_answer'), id: id });
            if (response === 'success') {
                document.getElementById(`${type}-${id}`).remove();
            } else {
                alert('Помилка видалення');
            }
        }
    }
  </script>
</body>
</html>
```

«Викладачі» можуть також редагувати тестування не лише при їх створенні, а й після того. Для цього на початковій сторінці вчителя є історія усіх створених ним тестувань, які можна фільтрувати за класом (групою) для якого він призначений. Кожен створений тест «викладач» може видалити, тоді всі результати учнів за цей тест видаляться, також можна редагувати вже створений тест, або переглянути оцінки та відповіді учнів за цей тест.

4.2.5 Реалізація процесів проходжень тестувань та оцінювання результатів

Коли «студент» авторизувався в системі, він потрапляє на початкову сторінку для своєї ролі, де показані доступні для проходження тестування та є можливість переглянути результати вже пройдених тестів. Доступ до тестувань «студентам» надається завдяки прив’язці до навчального класу (групи). Тобто кожен «студент» в системі прив’язаний до класу (групи) і з цього переліку класів (груп) викладачі обирають для кого буде призначено створений тест. Тоді система перевіряє до якої з груп належить студент і відображає доступні йому тестування і перевіряє чи кожне з тестувань пройденим, щоб унеможливити повторне проходження одного й того ж тесту одним студентом.

За процес проходження «студентом» тестування відповідає файл «take_test.php». Він перевіряє доступ студента до даного тесту, створює форму (інтерфейс) для проходження тестування, а також виконує перемішування порядку варіантів відповідей, аби запобігти повторенням (коли у різних «студентів» порядок варіантів відповідей в одному тексті однаковий). Також при початку тестування запускається «таймер» зі зворотнім відліком (час встановлений «викладачем» при створенні тетсування), після закінчення часу «таймеру» проходження тесту автоматично завершується.

Файл «take_test.php» відповідає за процес проходження тестувань:

```
<?php
```

```
include 'config.php';
check_auth(2); // Тільки учень

if (!isset($_GET['id'])) { header('Location: student_dashboard.php'); exit; }
$quiz_id = (int)$_GET['id'];
$student_id = $_SESSION['user_id'];

// Перевіряємо, чи тест вже не пройдено
$check_stmt = $db->prepare("SELECT id FROM quiz_results WHERE student_id
= ? AND quiz_id = ?");
$check_stmt->bind_param("ii", $student_id, $quiz_id);
$check_stmt->execute();
if ($check_stmt->get_result()->num_rows > 0) {
                die("Ви вже пройшли цей тест. <a
href='student_dashboard.php'>Повернутись</a>");
}
$check_stmt->close();

// Отримуємо інформацію про тест (назва, час)
$quiz_stmt = $db->prepare("SELECT title, time_limit FROM quizzes WHERE id
= ?");
$quiz_stmt->bind_param("i", $quiz_id);
$quiz_stmt->execute();
$quiz = $quiz_stmt->get_result()->fetch_assoc();
if (!$quiz) die("Тест не знайдено.");
?>

   <script>
      // Таймер
      let timeRemaining = <?php echo $quiz['time_limit'] * 60; ?>;
```

```
const display = document.getElementById('time-display');
const form = document.getElementById('testForm');
const modal = document.getElementById('resultModal');

function updateTimer() {
    let minutes = Math.floor(timeRemaining / 60);
    let seconds = timeRemaining % 60;
    display.textContent = `${minutes}:${seconds < 10 ? '0' : ''}${seconds}`;
    if (timeRemaining <= 0) { clearInterval(timerInterval); finishTest(); }
    timeRemaining--;
}
const timerInterval = setInterval(updateTimer, 1000);
updateTimer();

form.addEventListener('submit', function(e) {
    e.preventDefault();
    if(confirm('Завершити виконання тесту?')) finishTest();
});

async function finishTest() {
    clearInterval(timerInterval);
    const formData = new FormData(form);
    try {
        const response = await fetch('submit_test.php', { method: 'POST', body:
formData });
        const data = await response.json();
        if (data.success) {
                document.getElementById('modalPercent').textContent =
data.percentage + '%';
                document.getElementById('modalPoints').textContent =
```

```
data.earned_points + ' з ' + data.total_points;
                        document.getElementById('modalMessage').innerHTML =
data.message;
                modal.style.display = 'flex';
            } else { alert('Помилка: ' + data.message); }
        } catch (error) { console.error(error); alert('Помилка з\'єднання.'); }
    }
      window.onbeforeunload = function() { if (modal.style.display !== 'flex')
return "Тест не завершено!"; };
  </script>
</body>
</html>
```

За обробку результатів пройдених «студентами» тестувань відповідає файл програми «submit_test.php». Після перевірки правильності відповідей на кожне питань тесту, система підраховує та відображає результати проходження в балах та відсотках правильності відповідей. Відсоток успішності розраховується за формулою:

$$\%\text{усп.} = \frac{\sum \text{отримані бали}}{\sum \text{макс. можливі бали}} \cdot 100. \qquad (4.1)$$

За цією формулою також рахується загальний відсоток успішності учня за конкретною темою, враховуючи тестування лише за конкретною темою. Так само і рахується рахується загальний відсоток успішності учня за предмет (дисципліну) у цілому, враховуючи усі пройдені тестування з даного предмету.

Файл «submit_test.php» відповідає за перевірку результатів та підрахунок балів:

```
// Отримуємо правильні відповіді для авто-перевірки
```

```
foreach ($questions_info as $qid => $info) {
    if ($info['type'] != 'text') {
        $a_stmt = $db->prepare("SELECT id FROM answers WHERE question_id =
? AND is_correct = 1");
        $a_stmt->bind_param("i", $qid);
        $a_stmt->execute();
        $a_res = $a_stmt->get_result();
        while ($a = $a_res->fetch_assoc()) {
            $questions_info[$qid]['correct_ids'][] = $a['id'];
        }
        $a_stmt->close();
    }
}

// Збереження відповідей та підрахунок
$earned_score = 0;
$db->begin_transaction();

try {
    $status = $has_text_questions ? 'pending' : 'completed';
    $result_stmt = $db->prepare("INSERT INTO quiz_results (student_id, quiz_id,
score, status, date_taken) VALUES (?, ?, 0, ?, NOW())");
    $result_stmt->bind_param("iis", $student_id, $quiz_id, $status);
    $result_stmt->execute();
    $result_id = $db->insert_id;
    $result_stmt->close();

     $save_answ_stmt = $db->prepare("INSERT INTO user_answers (result_id,
question_id, chosen_answer_id, text_answer, points_awarded) VALUES (?, ?, ?, ?,
?)");
```

```
    // А. Обробка тестових питань
    foreach ($student_answers_post as $q_id => $user_ans) {
        if (!isset($questions_info[$q_id])) continue;
        $q_data = $questions_info[$q_id];
        $selected_ids = is_array($user_ans) ? $user_ans : [$user_ans];
        $correct_ids = $q_data['correct_ids'];
        $points_for_q = $q_data['points'];

        $q_score = 0;
        if ($q_data['type'] == 'single') {
            if (count($selected_ids) == 1 && in_array($selected_ids[0], $correct_ids))
$q_score = $points_for_q;
        } elseif ($q_data['type'] == 'multiple') {
            $total_correct = count($correct_ids);
            if ($total_correct > 0) {
                $val = $points_for_q / $total_correct;
                foreach ($selected_ids as $sid) {
                    if (in_array($sid, $correct_ids)) $q_score += $val;
                    else $q_score -= $val; // Штраф за неправильну
                }
            }
        }
        if ($q_score < 0) $q_score = 0;
        $earned_score += $q_score;

        // Зберігаємо відповіді.
        $is_first = true;
        foreach ($selected_ids as $sid) {
            $txt = null;
```

```
            $pts = $is_first ? $q_score : 0;
            $save_answ_stmt->bind_param("iiisd", $result_id, $q_id, $sid, $txt, $pts);
            $save_answ_stmt->execute();
            $is_first = false;
        }
    }

    // Обробка текстових питань
    foreach ($student_text_answers as $q_id => $text) {
        $text = trim($text);
        $sid = null;
        $pts = 0; // 0 балів до перевірки вчителем
        $save_answ_stmt->bind_param("iiisd", $result_id, $q_id, $sid, $text, $pts);
        $save_answ_stmt->execute();
    }
    $save_answ_stmt->close();

    // Оновлюємо результат в БД
        $percentage_score = ($total_possible_score > 0) ? ($earned_score /
$total_possible_score) * 100 : 0;
    $percentage_score = round($percentage_score, 2);
    $earned_score = round($earned_score, 2);

    $upd = $db->prepare("UPDATE quiz_results SET score = ? WHERE id = ?");
    $upd->bind_param("di", $percentage_score, $result_id);
    $upd->execute();
    $upd->close();

    $db->commit();
```

```
    $msg = $has_text_questions ? "<span style='color:orange'>Відправлено на перевірку вчителю.</span>" : "Тест завершено!";

    echo json_encode([
        'success' => true,
        'earned_points' => $earned_score,
        'total_points' => $total_possible_score,
        'percentage' => $percentage_score,
        'message' => $msg
    ]);

} catch (Exception $e) {
    $db->rollback();
        echo json_encode(['success' => false, 'message' => 'Error: ' . $e->getMessage()]);
}
?>
```

4.2.6 Реалізація прогнозування успішності за допомогою алгоритму лінійної регресії

У системі було реалізовано можливість прогнозування успішності складання наступного тестування (у відсотках успішності) за конкретною темою з того чи іншого предмету, та за предметом загалом. Для цього застосовується метод лінійної регресії, а саме – метод найменших квадратів.

Основою алгоритму є рівняння прямої:

$$y = mx + b, \tag{4.2}$$

де y – це прогнозована успішність (залежна змінна);

x – це порядковий номер пройденого тесту;

m – це коефіцієнт нахилу (динаміка росту чи падіння успішності);

b – це вільний член (точка перетину з віссю Y).

Для знаходження коефіцієнту нахилу m використовується метод найменших квадратів:

$$m = \frac{n\sum(xy) - \sum x \sum y}{n\sum(x^2) - (\sum x)^2}. \quad (4.3)$$

Після того, як було знайдено коефіцієнт нахилу m, знаходиться точка перетину b за формулою зсуву:

$$b = \frac{\sum y - m\sum x}{n}. \quad (4.4)$$

Після того, як було знайдено коефіцієнт нахилу m та точка перетину b, то можна передбачити значення y для будь-якого майбутнього x. У даному випадку ми передбачаємо результат наступного тесту, тобто x = n + 1:

$$y(\text{наст.}) = m(n+1) + b. \quad (4.5)$$

Також важливо зазначити, що для прогнозування на наступний тест, має бути завершено хоча б три тестування з необхідної теми чи предмету, це зроблено для більшої точності прогнозування. Оскільки регресія може показати результат більше 100%, або менше 0%, якщо результати тестувань різко покращуються або різко погіршуються, то для нормалізації результату прогнозування в програмному коді встановлено обмеження, що прогнозована успішність наступного тестування може бути в межі від 0% до 100%.

У файлі «prediction.php» реалізовано прогнозування майбутніх результатів успішності:

<?php

```
function predictNextScore(array $scores) {
    $n = count($scores);

    // Мінімум 3 тесту для прогнозу
    if ($n < 3) {
        return null;
    }

    $sum_x = 0;
    $sum_y = 0;
    $sum_xy = 0;
    $sum_xx = 0;

    // x = порядковий номер, y = оцінка
    for ($i = 0; $i < $n; $i++) {
        $x = $i + 1;
        $y = (float)$scores[$i];

        $sum_x += $x;
        $sum_y += $y;
        $sum_xy += ($x * $y);
        $sum_xx += ($x * $x);
    }

    $denominator_m = ($n * $sum_xx) - ($sum_x * $sum_x);

    if ($denominator_m == 0) {
        // Якщо всі x однакові або помилка обчислень
        return round(array_sum($scores) / $n, 2); // Повертаємо середнє
    }
```

```
    $m = (($n * $sum_xy) - ($sum_x * $sum_y)) / $denominator_m;
    $b = ($sum_y - $m * $sum_x) / $n;

    // Прогнозуємо для наступного кроку (x = n + 1)
    $predicted_score = ($m * ($n + 1)) + $b;

    // Обмеження 0-100%
    if ($predicted_score > 100) $predicted_score = 100.0;
    if ($predicted_score < 0) $predicted_score = 0.0;

    return round($predicted_score, 2);
}
?>
```

4.3 Розрахунок стійкості системи

Стійкість розробленої системи обліку та аналізу успішності визначається її здатністю забезпечувати безперервний доступ користувачів («студентів» та «викладачів») до необхідних даних, зберігати цілісність БД (MySQL) та функціонувати при пікових навантаженнях, коли одночасно багато «студентів» проходять тестування [4].

Головними чинниками стійкості системи є:

– надійність апаратного забезпечення сервера;

– безперервність роботи при відключенні, або перебоях живлення;

– енергоефективність системи.

Надійність системи, у тому числі залежить від того, як працює кожний з компонентів серверу, на якому запушено веб-додаток системи, а саме від тривалості безвідмовної роботи цих компонентів.

Зробимо припущення, що для роботи сервера застосовуються такі

компоненти серверного класу з наступним середнім часом безвідмовної роботи (MTBF):

– 60 000 годин для серверного процесору (CPU);

– 50 000 годин для оперативної пам’ять з корекцією помилок (RAM);

– 40 000 годин для SSD-накопичувача, де зберігаються дані БД (SSD);

– 30 000 годин для блоку живлення серверу (PSU).

Надійність для кожного з елементів системи обчислюється за наступною формулою за час (t):

$$R_i = e^{-\lambda it}, \tag{4.6}$$

де $\lambda_i = \frac{1}{MTBF}$ є інтинсивністю відмов у роботі компонента системи.

Доцільним буде розрахувати надійність системи за період 1 року роботи, що складає 8760 годин.

Формула розрахунку надійності для кожного компонента:

$$R_i(t) = e^{-\lambda it}, \tag{4.7}$$

де $\lambda_i = \frac{1}{MTBF_i}$ є інтенсивністю відмов компоненту, а t є часом роботи системи у годинах, у даному випадку – 8760 годин.

Розрахунок надійності для серверного процесору (CPU):

$$\lambda_{cpu} = \frac{1}{60000},$$

$$R_{cpu}(8760) = e^{-\frac{8760}{60000}} = e^{-0,146} = 0,864.$$

Розрахунок надійності для оперативної пам’яті (RAM):

$$\lambda_{ram} = \frac{1}{50000},$$

$$R_{ram}(8760) = e^{-\frac{8760}{50000}} = e^{-0,1752} = 0,839.$$

Розрахунок надійності для SSD-накопичувача (БД):

$$\lambda_{ssd} = \frac{1}{40000},$$

$$R_{ssd}(8760) = e^{-\frac{8760}{40000}} = e^{-0,219} = 0,803.$$

Розрахунок надійності для блоку живлення (PSU):

$$\lambda_{psu} = \frac{1}{30000},$$

$$R_{psu}(8760) = e^{-\frac{8760}{30000}} = e^{-0,292} = 0,747.$$

Оскільки для роботи подібної системи необхідне одночасне функціонування усіх компонентів, то правильно буде розрахувати загальну надійність системи R_{sys} для паралельного підключення компонентів.

Тоді формула для розрахунку загальної надійності системи виглядатиме наступним чином:

$$Rsys = Rcpu \cdot Rram \cdot Rssd \cdot Rpsu = 0,864 \cdot 0,839 \cdot 0,803 \cdot 0,747 = 0,435.$$

Значення, яке було отримано (0,435 або 43,5%) є ймовірністю безвідмовної роботи одного серверу на протязі року (8760 годин) без жодного обслуговування. Для того, аби підвищити рівень стійкості системи, у реальних умовах можна застосовувати дублювання серверу (резервування). Для цього застосовується паралельне підключення, тоді формула для розрахунку загальної надійності системи виглядатиме наступним чином:

$$R = 1 - (1 - Rsys)^2 = 1 - (1 - 0{,}435)^2 = 1 - 0{,}319 = 0{,}68.$$

Таким чином надійність системи за один рік роботи (8760 годин) має показник 0,68 або 68%. Це є хорошим показником стійкості і ймовірність безвідмовної роботи протягом року є достатньою.

Для забезпечення безперервності з'єднання та можливості відновлення даних у системі застосовано налаштування «Master-Slave» реплікації у MySQL для збереження оцінок студентів. А також використовується механізм транзакцій у PHP-скриптах ($db->begin_transaction()) для того, щоб гарантувати цілісність даних при раптовій втраті чи розриві з'єднання [4].

Для того, аби оцінити енергоефективність системи, треба розрахувати загальне енергоспоживання серверу, що необхідний для функціонування системи (MySQL, Apache, PHP).

Вихідними даними про енергоспоживання системи є:

– серверний процесор під навантаженням, близько 95 Вт;

– оперативна пам'ять, близько 10 Вт;

– SSD-накопичувач, близько 5 Вт;

– система охолодження та материнська плата, приблизно 40 Вт.

Тоді можна розрахувати сумарну потужність *P*:

$$P = 95\ \text{Вт} + 10\ \text{Вт} + 5\ \text{Вт} + 40\ \text{Вт} = 150\ \text{Вт}.$$

Сумарну потужність енергоспоживання за 1 рік роботи, або 8760 годин можна розрахувати наступним чином:

$$E = 150 \cdot 8760 = 1314000 \frac{\text{Вт}}{\text{год}} = 1314 \frac{\text{кВт}}{\text{год}}.$$

Необхідно забезпечити джерело безперебійного живлення (ДБЖ) у разі відключення електроенергії для автономної роботи системи протягом 20

хвилин для коректного завершення транзакцій у БД та безпечного завершення роботи сервера.

Дані для розрахунку ДБЖ:

P – потужність системи = 150 Вт;

t_a – час автономної роботи 20 хв = 0,33 год;

n – середній коєфіцієнт ефективності ДБЖ = 0,9;

V – напруга батареї = 12 В.

Тоді можна розрахувати ємність батареї *C* для ДБЖ:

$$C = \frac{P \cdot t_a}{n \cdot V} = \frac{150 \cdot 0{,}33}{0{,}9 \cdot 12} = \frac{49{,}5}{10{,}8} = 4{,}58 \frac{\text{А}}{\text{год}}.$$

Отже, для забезпечення 20 хвилин безперебійного живлення системи при екстреному відключенні електроенергії для завершення транзакцій у БД та безпечного завершення роботи сервера достатньо джерела безперебійного живлення з батареєю, ємність якої складає щонайменше 4,58 А/год.

4.4 Забезпечення безпечних умов праці при розробці автоматизованої системи обліку та аналізу поточної успішності студентів

Під час розробки та при впровадженні компонентів автоматизованої системи обліку та аналізу поточної успішності студентів, розробник дуже тривалий час проводить за комп’ютером. Аби знизити ризики для здоров’я, необхідно дотримуватись санітарних норм та вимог охорони праці.

Необхідний комплекс заходів, необхідних для забезпечення вимог охорони праці та санітарних норм повинен включати організацію ергономіки робочого місця розробника, забезпечення правильного освітлення у робочій кімнаті, забезпечення правильного мікроклімату, а також організацію захисту від різного шуму та електромагнітних випромінювань. Окрім того варто враховувати електробезпеку та пожежну безпеку [21].

Для організації безпечного робочого середовища розробника необхідно

забезпечити наступне:

Для забезпечення ергономіки робочого місця розробника, висота столу за яким він працює має бути від 72 до 75 см. Висота стільця повинна мати можливість регулювання в межах від 40 до 50 см, для того щоб забезпечити прямий кут у ліктьовому та колінному суглобах. Монітор має бути розташовано на відстані від 50 до 70 см від очей, верхній край монітору має бути на лінії погляду, задля збереження зору та зниження навантаження на очі [21].

При організації освітлення на робочому місці для роботи з комп’ютером, варто забезпечити рівень штучного освітлення в межах від 300 до 500 люкс. Для приміщення, лабораторії чи офісу площею 20 м² необхідний світловий потік становить приблизно від 6000 до 8000 люмен. Даний показник можна забезпечити, використовуючи 8 світлодіодних LED-ламп, потужність яких складає 10 – 12 Вт на кожну. Сумарна потужність освітлення має становити близько 80 – 100 Вт, щоб уникнути мерехтіння та забезпечити рівномірне падіння світлових променів.

Щоб підтримувати належний мікроклімат, слід врахувати об’єм приміщення. Якщо площі приміщення складає 20 м², а висота стелі дорівнює 2,7 м, то об’єм повітря в приміщенні становить 54 м³. Згідно з нормами, для для нормальної розумової роботи однієї особи, необхідно забезпечити циркуляцію повітря приблизно 30 м³/год на одну особу. Якщо команда розробників складається з двох осіб, то кімнати площею 20 м² достатньо для забезпечення комфортного мікроклімату, якщо систему вентиляції налаштовують на забезпечення обміну 60 м³/год циркуляції повітря [21].

Допустимий рівень електромагнітного випромінювання для персоналу, що працює з комп'ютерною технікою:

– напруга електричного поля (5 Гц – 2 кГц) не має бути більше 25 В/м;

– щільність магнітного потоку не має становити більше 250 нТл;

Окрім того, варто забезпечити правильне заземлення для системних блоків ПК та зробити екранування кабелів для мінімізації впливу

електромагнітного випромінювання.

Робоче приміщення необхідно забезпечити засобами гасіння пожежі. Так як у приміщенні імовірно знаходяться сервери та ПК, то правильним буде забезпечити наявність вуглекислотного вогнегаснику, який на відміну від порошкового не пошкоджує електроніку при гасінні. Також важливим чинником є наявність димових сповіщувачів, що підключені до пожежної сигналізації [21].

Варто розрахувати навантаження на електричну мережу. Для розробки та тестування системи використовується 2 робочі ПК розробників, в середньому по 400 Вт енергоспоживання (разом 800 Вт). Також наявний локальний сервер для хостингу БД та веб-сервера (300 Вт споживання), а також різне периферійне обладнання (монітори, мережевий комутатор, тощо) у сумі приблизно 200 Вт.

Отже, загальне споживання становить 1300 Вт. Тоді, можна розрахувати струм при стандартній напрузі мережі 220 В:

$$I = \frac{P}{U} = \frac{1300}{220} = 5{,}9\ A.$$

Стандартної електропроводки (мідний кабель перерізом 1,5 мм²) та розетки, що розраховані на струм 16 А буде достатньо для забезпечення безпечних умов експлуатації.

Для правильного забезпечення охорони праці, необхідно враховувати усі вище згадані параметри. Тому за результатами розрахунків, робоче місце з вказаними параметрами відповідає заданим нормам площі приміщення, необхідно освітлення, мікроклімату та електробезпеки, а також дозволяє мінімізувати шкідливі фактори при роботі над програмним продуктом і забезпечити правила пожежної безпеки.

4.5 Висновки до розділу

У даному розділі було створено та описано блок-схему розроблюваної системи, що дає змогу побачити основні процеси та функції.

Далі було детально описано процес програмної реалізації системної логіки та її компонентів. На даному етапі було описано реалізацію процесів підключення до БД та ініціалізації системи, реалізацію процесів авторизації та перенаправлення користувачів, було описано створення інструментів адміністратора системи, інструменти створення та редагування тестувань у ролі «викладача», процеси проходження тестувань «студентами», розглянуто реалізацію розрахунку оцінок. Було розглянуто процес програмної реалізації прогнозування успішності за допомогою лінійної регресії.

Були проведені розрахунки показників стійкості розробленої системи та визначено параметри її надійності.

Після чого було розглянуто забезпечення безпечних умов праці та норм пожежної безпеки.

Загалом, створена система готова для використання, або подальшої модифікації чи покращення.

ВИСНОВКИ

Розробка автоматизованої системи обліку та аналізу поточної успішності студентів є актуальним питанням сучасності. В контексті різних факторів все частіше відбувається перехід до навчання з використанням дистанційних технологій. Подібна система в майбутньому буде покращувати якість навчання, робити його більш доступним а також допоможе оптимізувати та автоматизувати основні процеси моніторингу та перевірки рівня знань.

При написанні кваліфікаційної роботи було розглянуто та виконано наступні етапи:

– розглянуто інтеграцію автоматизованих технологій у навчальні процеси;

– розглянуто сучасні методи контролю та оцінки успішності;

– проаналізовано приклади сучасних систем управління навчанням (LMS), виділено основні переваги та недоліки;

– виконано постановку задач, які має виконувати розроблювана система;

– зроблено вибір та обгрунтування програмних засобів для реалізації поставлених задач;

– розглянуто та описано прогнозування успішності за допомогою лінійної регресії;

– розроблено базу даних системи та її ER-модель;

– описано таблиці БД, їх зв’язки та параметри;

– розроблено блок-схему програми;

– описано програмну реалізацію компонентів системи;

– проведено розрахунок параметрів стійкості системи;

– описано забезпечення безпечних умов праці та пожежної безпеки.

Створена система контролю та аналізу успішності має зручний і

інтуїтивно зрозумілий при використанні інтерфейс. Для користування системою, користувач повинен мати заздалегідь створений акаунт, або його має створити адміністратор (для авторизації використовується логін та пароль). Система передбачає дві ролі користувачів «викладач» та «студент», кожна з яких має свої властивості та можливості. Окремо існує роль адміністратора системи, який керує іншими користувачами. Роль «викладача» в системі – створювати, редагувати тестування за своєю навчальною дисципліною і якщо це потрібно, перевіряти відповіді студентів та оцінювати їх. Також «викладач» має доступ до історії проходження учнями тестувань з їх предмету, а також має повну навчальну статистику усіх свої учнів, як за предметом у цілому, так і за конкретною темою. «Студенти» в свою чергу можуть проходити доступні їм тестування, отримувати оцінки, бачити результати (правильні та неправильні відповіді), а також, подібно до «викладачів» повну навчальну статистику, але «студенти» бачать лише свої оцінки за обраним предметом чи темою.

Також варто зазначити, що розроблена система окрім простих елементів сортування та статистики навчальних результатів, також має елементи прогнозування майбутньої успішності. Це реалізовано завдяки математичному алгоритму лінійної регресії, реалізованому у програмному коді. Дана функція дозволяє прогнозувати успішність студента (у відсотках) на наступний тест за конкретною темою, або успішність за конкретним предметом загалом. Для отримання прогнозу успішності (у відсотках) студент має пройти хоча б три тестування за темою чи предметом.

Поставлені на кваліфікаційну роботу задачі було виконано у повному обсязі. Розроблену систему було реалізовано таким чином, що вона може застосовуватись для перевірки та моніторингу чи прогнозування успішності учнів у закладах середньої та вищої освіти. Також є можливості для модифікації чи подальшого вдосконалення системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Методичні вказівки з підготовки та захисту кваліфікаційної роботи здобувачами другого (магістерського) рівня вищої освіти спеціальності 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка, освітньо-професійних програм: «Комп'ютерно-інтегровані технологічні процеси і виробництва», «Комп'ютеризовані та робототехнічні системи» / Упоряд. І. Ш. Невлюдов, Р. В. Артюх, В. В. Безкоровайний, Н. П. Демська, В. В. Євсєєв, О. І. Филипенко, О. М. Цимбал. Харків: ХНУРЕ, 2024. 57 с.

2. ДСТУ 3008: 2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. К.: ДП «УкрНДНЦ». 2016. 30 с.

3. Обривко Є.В. Аналіз методів і функцій захисту даних для ресурсів дистанційного навчання / Автоматизація та Приладобудування («Automation and Development of Electronic Devices» ADED-2024) [Електронний ресурс] : збірник студентських наукових статей / Харківський національний університет радіоелектроніки ; [редкол.: І.Ш. Невлюдов та ін.]. – Харків : ХНУРЕ, 2024. – Вип. 2. – с. :107 – 112.

4. Обривко Є.В. Аналіз методів роботи системи дистанційного навчання при навантаженні/ Автоматизація та Приладобудування («Automation and Development of Electronic Devices» ADED-2025) [Електронний ресурс] : збірник студентських наукових статей / Харківський національний університет радіоелектроніки ; [редкол.: І.Ш. Невлюдов та ін.]. – Харків : ХНУРЕ, 2024. – Вип. 1. – с. :17 – 22.

5. Гуменний О. Д. Інтеграція цифрових технологій в освітній процес: Smart EcoSystem – унікальна цифрова екосистема для персоналізованого фахового навчання [Електронний ресурс] / О. Д. Гуменний, О. І. Федоренко // Педагогічна академія: наукові записки. – 2025. – Режим доступу: https://doi.org/10.5281/zenodo.15067293.

6. Основні форми контролю знань студентів. Реферат: osvita.ua. [Електронний ресурс].- URL: https://osvita.ua/vnz/reports/pedagog/14679/ (дата звернення: 23.09.2025).

7. Мазурок Т. Л. Системи управління навчанням: навч. посібник для здобувачів другого (магістерського) рівня вищої освіти ОПП «Середня освіта (Інформатика Мова та література (англійська))», ОПП «Середня освіта (Інформатика)» спеціальності 014 «Середня освіта (Інформатика)» / Т. Л. Мазурок. - Одеса: ПНПУ імені К.Д. Ушинського, 2021. - 201 с.

8. Технології електронного навчання: наукове електронне видання. № 6 / Ред. рада: В.Є. Величко, О.Г. Федоренко, Н.В. Кайдан, А.В. Стьопкін, Я.В. Топольник. Слов'янськ: ДВНЗ «ДДПУ». 2022. 86 с.

9. Постова М. В. Використання системи електронного навчання MOODLE для контролю і оцінювання навчальної діяльності студентів ВНЗ: Методичний посібник / М. В. Постова, В. В. Фомін, В. В. Шаров. Черкаси: ЧДТУ. 2013. 48 с.

10. Google Forms. Онлайн-редактор форм: workspace.google.com. [Електронний ресурс].- URL: https://workspace.google.com/intl/uk/products/forms/ (дата звернення: 28.09.2025).

11. Невлюдов І.Ш. Техніко-економічне обґрунтування інженерних рішень: Підручник / І.Ш. Невлюдов. Кривий Ріг : Криворізький коледж НАУ. 2019. 448 с.

12. Встановлення та налаштування локального веб-сервера MAMP на Windows [Електронний ресурс] // HostPro. – Електрон. текстові дані. – Режим доступу: https://hostpro.ua/wiki/ua/instructions/installing-and-configuring-the-local-mamp-web-server-on-windows/ (дата звернення: 03.10.2025).

13. Булатецька Л. В. Мова запитів SQL : текст лекцій нормативної навчальної дисципліни «Бази даних та розподілені інформаційно-аналітичні системи» [Електронний ресурс] / Л. В. Булатецька, В. В. Булатецький. – Луцьк : СНУ імені Лесі Українки, 2018. – 92 с. – Режим доступу:

http://evnuir.vnu.edu.ua/handle/123456789/17722.

14. Лінійна регресія [Електронний ресурс] // Вікіпедія : вільна енциклопедія. – Електрон. текстові дані. – Режим доступу: https://uk.wikipedia.org/wiki/Лінійна_регресія (дата звернення: 10.10.2025).

15. Барковський В. В. Теорія ймовірностей та математична статистика : навч. посіб. / В. В. Барковський, Н. В. Барковська, О. К. Лопатін. – 5-те вид. – Київ : Центр учбової літератури, 2010. – 424 с. – ISBN 978-966-364-992-4.

16. Метод найменших квадратів [Електронний ресурс] // Вікіпедія : вільна енциклопедія. – Електрон. дані. – Режим доступу: https://uk.wikipedia.org/wiki/Метод_найменших_квадратів (дата звернення: 12.10.2025).

17. Пасічник В. В. Організація баз даних та знань : підручник / В. В. Пасічник, В. А. Резніченко. – Київ : Видавнича група BHV, 2006. – 384 с. – ISBN 966-552-156-X.

18. Блок-схема [Електронний ресурс] // Вікіпедія : вільна енциклопедія. – Електрон. текстові дані. – Режим доступу: https://uk.wikipedia.org/wiki/Блок-схема (дата звернення: 15.10.2025).

19. Мережеві запити [Електронний ресурс] // Сучасний підручник з JavaScript. – Електрон. текстові дані. – Режим доступу: https://uk.javascript.info/network (дата звернення: 21.10.2025).

20. Тиш Є. В., Литвиненко Я.В. Надійність, контроль, діагностика та експлуатація ЕОМ : конспект лекцій [Електронний ресурс] / Тиш Є. В., Литвиненко Я.В. – Тернопіль : НТУ імені І. Пуюля, 2020. – 107 с.

21. Методичні вказівки до виконання розділу "Охорона праці" у випускних роботах ОКР "бакалавр" усіх форм навчання / упоряд.: В. А. Айвазов. Т. Є. Стиценко., Н. Л. Березуцька ; М-во освіти і науки України, ХНУРЕ. – Харків : ХНУРЕ, 2018. – 28 с. – 1,81.