КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Автоматизації і інформаційних технологій
(факультет)

Інформційних технологій проєктування та прикладної математики
(назва кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО АТЕСТАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

на тему:

Інтелектуальна система електронної оцінки знань студентів.

Власюка Віталія Володимировича
(прізвище, ім’я та по батькові студента повністю)

Київ 2024 р

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Автоматизації і інформаційних технологій
(факультет)

Інформційних технологій проєктування та прикладної математики
(назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., професор Терентьєв О.О.
_____
«___» _____2024 року

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО АТЕСТАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

Інтелектуальна система електронної оцінки знань студентів.
(назва)

Виконав студент групи ІСТм-26
126 «Інформаційні системи та технології»
(спеціальність)
«Інформаційні системи та технології»
(Освітньо-професійна програма)
Власюк Віталій Володимирович
(прізвище, ім’я та по батькові повністю)
Керівник Теренчук С.А.
(прізвище та ініціали)
к.фіз.-мат.н., доцент кафедри ІТППМ
(вчене звання, науковий ступінь)

Київ 2024 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет: Автоматизації і інформаційних технологій

Кафедра: Інформційних технологій проєктування та прикладної математики

Освітній рівень: «магістр за ОПП/ОНП»

Спеціальність:126 «Інформаційні системи та технології»

Освітньо-професійна програма: Інформаційні системи та технології

ЗАТВЕРДЖУЮ

Завідувач кафедри

д.т.н., професор Терентьєв О.О.

_____

«___» _____2024 року

З А В Д А Н Н Я

ДО ВИКОНАННЯ АТЕСТАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

Власюка Віталія Володимировича

(прізвище, ім'я та по батькові студента)

1. Тема роботи Інтелектуальна система електронної оцінки знань студентів.
затверджена наказом ректора КНУБА № 957/2 від «6» червня 2024 року
2. Керівник роботи к.фіз.-мат., доцент Теренчук Світлана Анатоліївна.
(прізвище, ім'я та по батькові, науковий ступінь, вчене звання)
3. Строк подання студентом роботи до захисту 04.12.24
4. Зміст пояснювальної записки за розділами:
Р. 1. АНАЛІЗ ПРОБЛЕМИ ЕЛЕКТРОННОГО ОЦІНЮВАННЯ ЗНАНЬ
Р. 2. ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
Р. 3. РОЗРОБКА ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ЕЛЕКТРОННОГО ОЦІНЮВАННЯ ЗНАНЬ
Р. 4. ВПРОВАДЖЕННЯ, ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ
5. Графічний матеріал за розділами
Р. 1. 4 рисунка
Р. 2. 1 рисунок
Р. 3. 3 рисунки
6. Календарний план виконання роботи: а) наукова частина; б) практична частина.

Види робіт та їх зміст	Дата виконання
Р. 1. АНАЛІЗ ПРОБЛЕМИ ЕЛЕКТРОННОГО ОЦІНЮВАННЯ ЗНАНЬ	27.09.24

Р. 2. ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	15.10.24
Р. 3. РОЗРОБКА ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ЕЛЕКТРОННОГО ОЦІНЮВАННЯ ЗНАНЬ	25.10.24
Р. 4. ВПРОВАДЖЕННЯ, ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ	05.11. 24
Остаточне оформлення роботи	10.11. 24
Направлення роботи на рецензування, перевірку на плагіат	15.11. 24
Попередній захист роботи на кафедрі	17.11. 24

7. Консультанти розділів атестаційної випускної роботи

Розділ	Прізвище, ініціали та посада консультанта	Перевірив	
		дата	підпис
Р. 1. АНАЛІЗ ПРОБЛЕМИ ЕЛЕКТРОННОГО ОЦІНЮВАННЯ ЗНАНЬ		26.09.2024	
Р. 2. ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ		15.10.2024	
Р. 3. РОЗРОБКА ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ЕЛЕКТРОННОГО ОЦІНЮВАННЯ ЗНАНЬ		31.10.2024	
Р. 4. ВПРОВАДЖЕННЯ, ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ		03.11.2024	

9. Дата видачі завдання _____

Зав. кафедри _____ Терентьєв О.О.
(підпис) (прізвище та ініціали)

Керівник _____ Теренчук С.А.
(підпис) (прізвище та ініціали)

Студент _____ Власюк В.В.
(підпис) (прізвище та ініціали)

РЕЗЮМЕ

Атестаційна випускна робота на здобуття освітнього ступеня магістра складається з вступу, чотирьох розділів, загальних висновків, списку використаних джерел та додатків. Загальний обсяг роботи складає 77 сторінок, містить 8 рисунків, 1 таблиця у тексті. Список використаних джерел налічує 25 найменувань і займає 2 сторінки.

Метою роботи є розробка електронної системи оцінки знань студентів, яка забезпечує автоматизацію процесу, підвищує об’єктивність результатів і адаптується до навчального контексту.

У роботі проведено аналіз існуючих підходів до електронного оцінювання, розроблено архітектуру програмного забезпечення для інтелектуальної оцінки, створено прототип системи, проведено його тестування на реальних даних та здійснено аналіз отриманих результатів.

Актуальність дослідження зумовлена необхідністю модернізації процесу оцінювання знань шляхом використання сучасних технологій, що зменшує вплив людського фактора та сприяє підвищенню якості освіти.

Наукова новизна роботи полягає у здатності системи оцінювання до адаптації в спілкуванні з користувачем на базі аналізу індивідуальних навчальних характеристик та характерні особливостей студентів.

Практичне значення полягає у впровадженні системи, яка автоматизує оцінювальні процеси, підвищує їх прозорість та зменшує час, необхідний для перевірки знань.

Ключові слова: інтелектуальні системи, електронне оцінювання, автоматизація, адаптивність, алгоритми, об’єктивність, освітні технології.

RESUME

The master's thesis consists of an introduction, four chapters, general conclusions, a list of references, and appendices. The total volume of the work is 80 pages, including 8 figures and 1 table in the text. The list of references contains 25 sources and spans 2 pages.

The aim of the thesis is to develop an electronic knowledge assessment system for students that ensures process automation, enhances the objectivity of results, and adapts to the educational context.

The research includes an analysis of existing approaches to electronic assessment, the development of software architecture for intelligent evaluation, the creation of a system prototype, its testing on real data, and an analysis of the obtained results.

The relevance of the research is determined by the need to modernize the knowledge assessment process through the use of modern technologies, which reduce the influence of the human factor and contribute to improving the quality of education.

The scientific novelty of the work lies in the creation of an adaptive intelligent assessment system that analyzes and considers the individual learning characteristics of students and their distinctive traits.

The practical significance of the work is the implementation of a system that automates evaluation processes, increases transparency, and reduces the time required for knowledge assessment.

Keywords: intelligent systems, electronic assessment, automation, adaptability, algorithms, objectivity, educational technologies.

ВСТУП

Сучасний освітній процес неможливо уявити без використання інформаційних технологій, які стали невід'ємною частиною систем оцінювання знань. Інноваційні підходи до електронного тестування дозволяють значно підвищити ефективність оцінювання і об’єктивність результатів, використовуючи інтелектуальні алгоритми для автоматичного аналізу відповідей студентів. Це набуває особливої важливості в умовах зростаючої складності навчальних програм та необхідності забезпечення справедливого і точного оцінювання знань.

Електронне оцінювання знань є основою для об’єктивного моніторингу навчального процесу та дозволяє мінімізувати вплив людського фактора. Використання сучасних технологій, таких як адаптивні тестові системи та інтеграція з алгоритмами оцінювання, сприяє підвищенню якості освіти. У свою чергу, інтеграція таких інструментів з системами для автоматичного виявлення шахрайства, моніторингу підготовки та аналізу часу виконання тестів забезпечує більш точне та персоналізоване оцінювання.

З огляду на швидкий розвиток технологій і зростаючі вимоги до ефективного використання часу студентів, інтеграція інтелектуальних систем оцінювання, здатних адаптувати тестові завдання та враховувати поведінкові фактори, є ключовим аспектом вдосконалення освіти. Такий підхід дозволяє забезпечити більш високий рівень мотивації студентів, сприяє розвитку їх критичного мислення та дає можливість здійснювати більш персоналізовану оцінку навчальних досягнень.

З огляду на ці виклики, важливість створення та впровадження адаптивних систем електронного оцінювання знань з використанням інтелектуальних алгоритмів є особливо актуальною. Це і є основною метою даного дипломного проєкту, який спрямований на розробку системи, що забезпечить об’єктивне оцінювання та мотивацію студентів у процесі навчання.

1 АНАЛІЗ ПРОБЛЕМИ ЕЛЕКТРОННОГО ОЦІНЮВАННЯ ЗНАНЬ

1.1 Аналіз проблеми

Розвиток сучасних технологій істотно впливає на всі сфери суспільства, включаючи освіту. У зв'язку з цифровізацією навчального процесу, одним із ключових питань, що потребують вирішення, є забезпечення об'єктивності та ефективності систем оцінювання знань студентів. Традиційні методи тестування мають низку обмежень, які стають дедалі більш очевидними в умовах швидких змін сучасного освітнього середовища.

Проблеми традиційного оцінювання:

Суб'єктивність оцінок. Викладачі часто оцінюють знання студентів на основі особистих спостережень, що може бути піддане впливу таких факторів, як втома, емоційний стан або особисте ставлення до студента. Це призводить до ризику необ'єктивності оцінювання та створює нерівні умови для студентів.

Затримка у зворотному зв'язку. У традиційній системі оцінювання між моментом завершення тесту та отриманням результатів часто проходить значний час. Це уповільнює процес закріплення матеріалу, оскільки студенти не можуть миттєво дізнатися про свої помилки і почати роботу над їх виправленням.

Відсутність індивідуального підходу. Багато стандартних тестів не враховують рівень підготовки окремого студента або його індивідуальні потреби. Це може призвести до ситуацій, коли завдання є або занадто складними, або занадто простими для конкретного студента, що знижує ефективність тестування.

Обмежені можливості для повторного закріплення матеріалу. Студенти зазвичай мають обмежену кількість спроб для здачі тестів, і повторні спроби можуть не враховувати зміни в знаннях чи підготовці студента. Це призводить до того, що можливості для швидкого навчання і корекції помилок залишаються невикористаними.

Проблеми в онлайн-тестуванні:

Шахрайство та недоброчесна поведінка. Використання електронних систем оцінювання пов'язане з ризиками використання сторонньої допомоги або

матеріалів під час тестування. Це підриває довіру до результатів і вимагає впровадження надійних заходів для запобігання шахрайству.

Монотонність завдань та запам'ятовування відповідей. Під час повторного складання одного і того ж тесту студенти можуть просто запам'ятати правильні відповіді, що не сприяє реальному покращенню знань. Це створює потребу у створенні систем, здатних адаптувати завдання під час повторних спроб.

Сучасні підходи та їх недоліки. Існують численні платформи та програми для електронного тестування, однак багато з них обмежуються стандартною перевіркою правильності відповідей. Рідко враховуються інші фактори, що впливають на успішність тесту, такі як швидкість відповіді, кількість спроб або поведінкові аспекти студента.

Потреба у вдосконаленні. Для покращення процесу оцінювання необхідна система, яка не лише автоматизує перевірку відповідей, а й враховує комплексні показники успішності, включаючи швидкість реакції, підготовку та поведінкові характеристики студентів. Така система має сприяти мотивації студентів до активного навчання та швидкого виправлення своїх помилок, а також створювати умови для більш об'єктивної оцінки.

Таким чином, аналіз існуючих проблем традиційних та сучасних електронних методів оцінювання знань вказує на нагальну необхідність розробки інтелектуальної системи, здатної не лише оцінювати, а й мотивувати студентів до подальшого навчання, сприяти більш глибокому розумінню матеріалу і забезпечувати об'єктивність у процесі оцінювання.

1.2 Формулювання завдань

На основі проведеного аналізу проблем у системах оцінювання знань можна сформулювати основні завдання та виклики, які потребують вирішення для забезпечення об'єктивності, адаптивності та ефективності електронних тестів у сучасній освіті (Рис 1.1.)

Сутність проблеми. Традиційні методи оцінювання знань студентів, навіть за використання електронних систем, часто не враховують ключові аспекти, що впливають на якість оцінювання. Під час тестування основна увага зазвичай

зосереджена на правильності відповідей, однак це не забезпечує повного уявлення про рівень підготовки студента. Ігнорування таких показників, як швидкість відповіді, кількість спроб, підготовленість та готовність до швидкої перевірки знань, обмежує ефективність тестування та його мотиваційний потенціал.

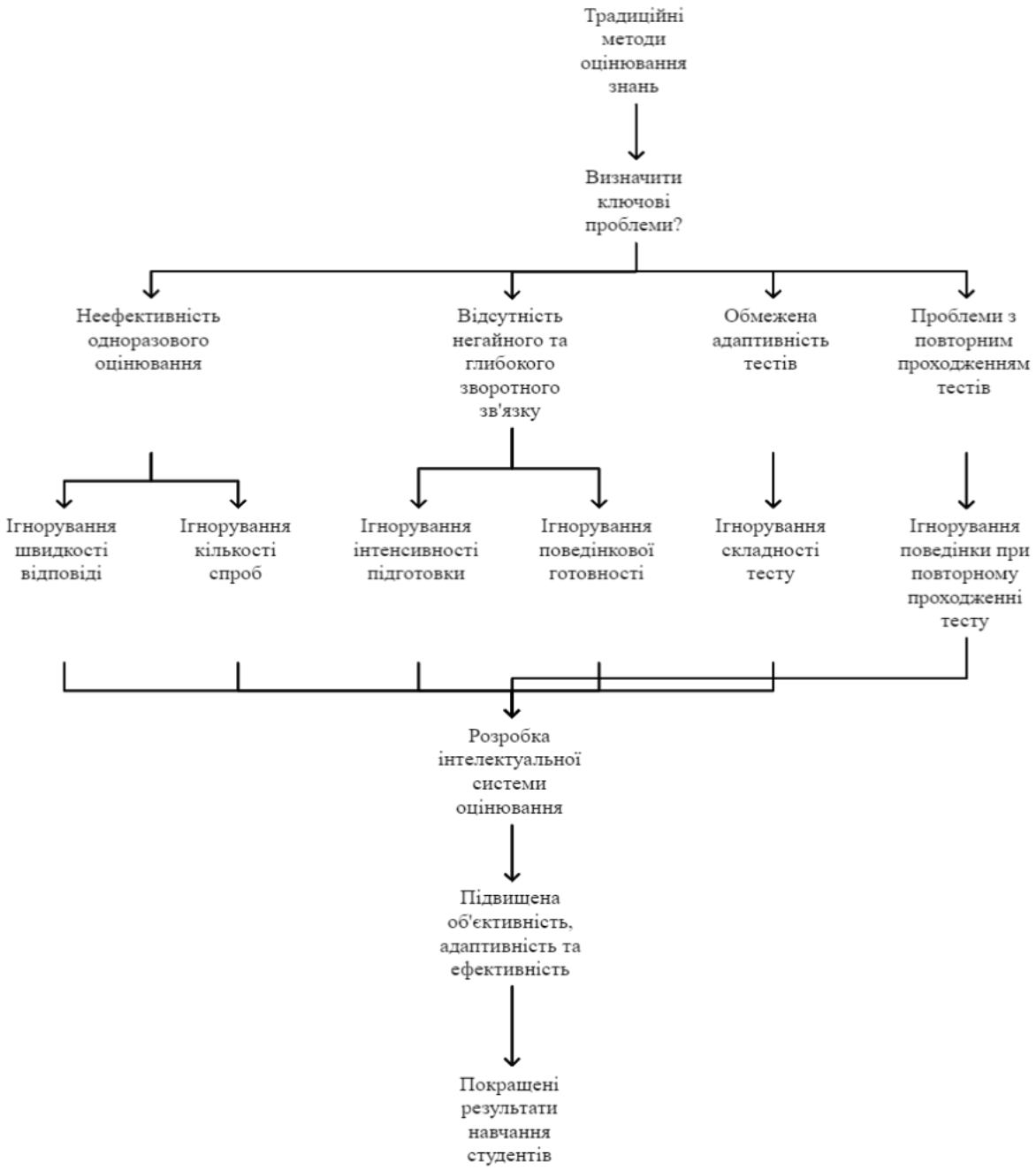

Рисунок 1.1 Формулювання завдань

Ключові аспекти проблеми:

Неефективність одноразового оцінювання. Системи, що оцінюють студентів за один тестовий результат, не можуть врахувати його поведінкові особливості, підготовку та здатність до швидкого повторення тесту.

Відсутність миттєвого і глибокого зворотного зв'язку. Студенти часто потребують оперативної інформації про свої помилки та рекомендації щодо їх виправлення. Сучасні системи надають лише базовий результат, що не сприяє глибокому закріпленню знань.

Обмежена адаптивність тестування. Використання стандартних тестів не враховує рівень складності, який підходить конкретному студенту, через що деякі завдання можуть бути надто легкими чи складними, не відображаючи реального рівня знань.

Проблеми повторних спроб. Можливість повторного проходження тестів без змін завдань призводить до механічного запам'ятовування відповідей замість засвоєння матеріалу. Це не дозволяє системі ефективно оцінювати рівень знань під час повторних тестів.

Формулювання проблеми. Проблема полягає у відсутності комплексної інтелектуальної системи оцінювання знань, яка б інтегрувала різноманітні фактори у процес оцінки, включаючи швидкість відповідей, кількість спроб, інтенсивність підготовки, час проходження повторного тесту та поведінкову готовність. Така система повинна мотивувати студентів до негайного закріплення матеріалу, створювати умови для об'єктивної оцінки та адаптуватися до потреб кожного студента.

Необхідність вирішення. Розробка системи, що вирішує ці проблеми, дозволить не тільки підвищити якість оцінювання знань студентів, а й сприятиме активному навчанню. Вона забезпечить більш прозорий, ефективний і мотивуючий процес оцінювання, який відповідає вимогам сучасної освіти та сприяє розвитку цифрових навичок у студентів.

1.3 Дерево основних цілей

Шлях до створення інтелектуальної системи електронного оцінювання знань студентів, яка здатна забезпечити об’єктивність, адаптивність, мотивацію

студентів і захист від шахрайства, представлений у вигляді дерева цілей і показано на рис. 1.2.

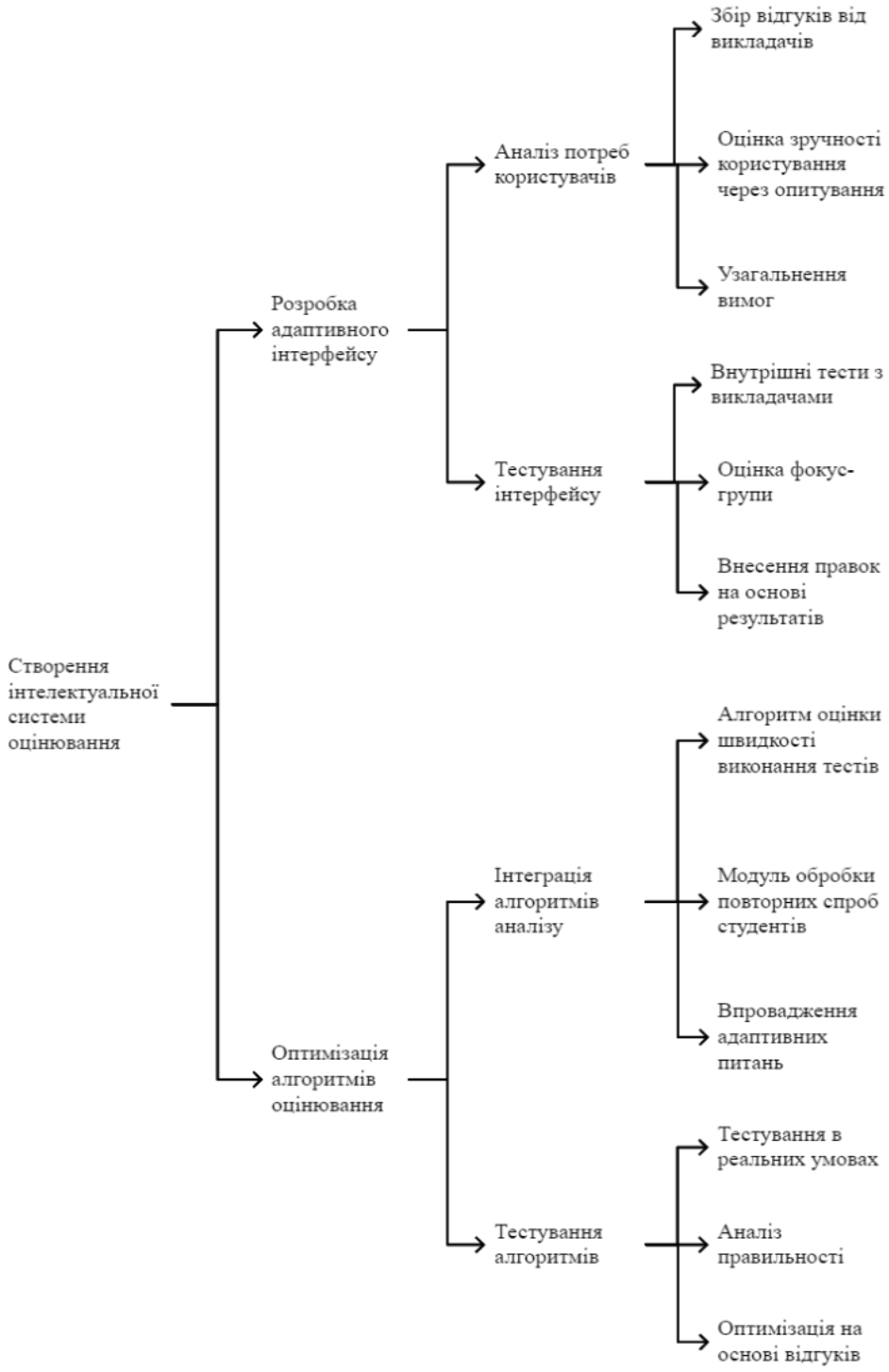

Рисунок 1.2 Дерево цілей

Це дерево визначає основні аспекти, які необхідно враховувати для ефективної реалізації системи.

Дослідження охоплює розробку алгоритмів, що аналізують швидкість відповідей, підготовку до тесту, повторні спроби тестування, а також адаптивність системи, яка змінює тестові завдання залежно від поведінки студентів. Важливим аспектом є врахування індивідуальних особливостей кожного студента, що дозволяє створити персоналізоване навчальне середовище. Це дозволить створити систему, яка забезпечить не лише об’єктивне оцінювання, а й підтримуватиме високий рівень мотивації студентів, сприятиме глибшому засвоєнню матеріалу та мінімізуватиме можливість шахрайства.

Завдяки впровадженню інтелектуальних алгоритмів та адаптивних механізмів система зможе коригувати складність завдань на основі індивідуальних досягнень студентів, створюючи стимул для подальшого розвитку. Крім того, за допомогою аналітики даних, система буде здатна виявляти потенційні порушення та шахрайські дії, забезпечуючи тим самим чесність та прозорість процесу оцінювання.

1. Основна ціль: Створення інтелектуальної системи електронного оцінювання знань студентів для підвищення ефективності та об'єктивності оцінювання.

1.1. Розробка адаптивного інтерфейсу системи

1.1.1. Аналіз потреб користувачів:

1.1.1.1. Збір відгуків від викладачів для визначення ключових функцій.

1.1.1.2. Оцінка зручності користування студентами через опитування.

1.1.1.3. Узагальнення результатів та визначення пріоритетних вимог.

1.1.2. Тестування інтерфейсу:

1.1.2.1. Проведення внутрішніх тестів з викладачами.

1.1.2.2. Залучення фокус-групи студентів для оцінки інтерфейсу.

1.1.2.3. Внесення правок на основі отриманих результатів.

1.2. Оптимізація алгоритмів оцінювання:

1.2.1. Інтеграція алгоритмів аналізу:

1.2.1.1. Розробка алгоритму для оцінки швидкості виконання тестів.

1.2.1.2. Створення модуля обробки повторних спроб студентів.

1.2.1.3. Впровадження функції адаптивних питань для уникнення запам’ятовування відповідей.

1.2.2. Тестування алгоритмів:

1.2.2.1. Проведення тестування на реальних прикладах.

1.2.2.2. Аналіз коректності оцінювання за допомогою контрольних груп.

1.2.2.3. Оптимізація на основі тестів і відгуків.

1.4 Аналіз існуючих рішень

Проблемі впровадження в процес навчання комп’ютерного тестування присвячено праці таких зарубіжних і українських науковців [3], [4], [5], [6], [7]. Результатом цих досліджень є велика кількість засобів, які призначені для тестування знань [8].

Найпопулярнішими серед online-сервісів, для викладачів і репетиторів, які допомагають створювати тести та інші завдання у найрізноманітніших форматах і представлені у відкритому доступі кількома мовами, включаючи українську, є:

- LearningApps.org [9];
- Online Test Pad [10];
- ClassMarker [11];
- Moodle [12], [13];
- OpenTest [14];
- Тесторіум [15].

LearningApps.org [9] є сервісом, який призначено для створення і підтримки інтерактивних вправ ігрового характеру в процесі здобуття середньої освіти.

Категорія предметів: англійська, німецька, французька, іспанська, італійська, українська та інші мови, інформатика, історія, трудове навчання, астрономія, біологія, географія, економіка, математика, мистецтво, музика, основи здоров'я, навколишній світ, політологія, релігієзнавство, професійна освіта, філософія, психологія, технічні науки, фізкультура, фізика та хімія.

LearningApps.org підтримує такі типи завдань: знаходження пари, кросворди, класифікація, хронологічний рядок, просте впорядкування,

сортування зображення, введення тексту, вікторини, заповнення прогалин, сіткові програми, аудіо/відео вміст, «Хто хоче стати мільйонером?», пазл «Вгадай-ка», шибениця, слова з букв, гра «Парочки». Посилання на завдання надаються у вигляді QR-коду, який система автоматично генерує для кожного завдання. Також викладач може додати посилання для вправ на сайт, де зберігаються створені викладачем завдання, які він може редагувати та змінювати видимість для конкретних користувачів.

Для виконання вправ викладач створює обліковий запис із зазначенням свого логіна, електронної пошти і пароля. В обліковий запис викладач додає здобувача освіти після його реєстрації. Здобувач освіти може обрати рівень складності завдань. При цьому вправи і блоки завдань не замінюють повноцінного уроку, а можуть бути лише частиною певного виду діяльності під час уроку, яка виконується для забезпечення зворотного зв'язку.

Online Test Pad [10] − безкоштовний багатофункціональний сервіс для навчання і тестування через Інтернет, який складається з: конструктора тестів; конструктора опитувань; конструктора кросвордів; комплексних завдань; діалогових тренажерів; системи дистанційного навчання та тестування. Цей веб-сайт є доповненням для створення навчальних матеріалів і завдань та структурування їх у папки. Online Test Pad містить детальні вказівки, як створити завдання різних типів і здійснити онлайн опитування. Для цього вчитель отримає html-код, який дозволяє розмістити завдання на власному веб-сайті, в блозі та додавати нових користувачів і групувати їх у різні групи та організації.

Крім того, Online Test Pad містить певну кількість завдань з предметів початкової школи. Формат тестових питань включає: один або кілька правильних відповідей, відповідь у вільній формі, встановлення послідовності і відповідності, заповнення пропусків, слайдер, службовий текст, завантаження файлу, послідовне виключення, інтерактивний диктант. Цей формат підходить як для домашніх завдань, так і для індивідуальних та контрольних робіт. Завдання будь-якого типу можуть бути опубліковані для загального доступу до Інтернету.

ClassMarker [11] − англомовний сервіс для швидкої побудови тестових завдань та опитувань із найширшим форматом відповідей. Безкоштовна версія надає можливість зробити до 100 тестів. Пакети розширених функцій надаються на платній основі. При використанні Google Chrome, текст може автоматично перекладатись на іншу мову.

Цей ресурс також дозволяє:

- отримувати швидкі посилання до тестових завдань можна;
- додавати зображення, аудіофайл, формулу чи відео до запитань;
- визначати кількість варіантів відповіді;
- включати питання щодо зворотного зв'язку, кількості балів, шкали оцінок.

Основними недоліками цих програмних рішень є:

- відсутність функції автоматичного генерування тестів, що означає витрати часу на створення варіацій одного тесту;
- відсутність розподілу завдань тесту за складністю, що зменшує об’єктивність оцінювання.

Moodle [12] – сервіс управління вмістом сайту Content Management System, яку розроблено для створення викладачами online-курсів і освітніх веб-сайтів. Сервіс базується на потужному інструментарії і пропонує широкий спектр можливостей для повноцінної підтримки процесу навчання в дистанційному середовищі – різні способи подання навчального матеріалу, перевірки знань і контролю успішності: множинний вибір, відповідність, так/ні, короткі відповіді. Також Moodle надає низку функцій, які полегшують опрацювання результатів. Однією з них є можливість побачити відповідь з помилкою для подальшого аналізу [13].

“OpenTest” [14] – спеціалізована система для тестування, що призначена для очного підсумкового контролю якості знань студентів у великих навчальних організаціях зі складною розподіленою структурою. Основною особливістю системи є спрямованість на забезпечення тестування з максимально строгою звітністю. Областю застосування є різноманітні підсумкові тестування, заліки,

іспити, кваліфікаційні тести та будь-які інші види контролю знань, у яких головну роль відіграє максимально об’єктивна оцінка знань. Система працює на основі веб-серверу і для її роботи потрібен тільки мережевий доступ до серверу, на якому встановлена система, та веб-браузер на робочому комп’ютері.

“Тесторіум” [15] – безкоштовна online-система із створення тестів і проведення тестування здобувачів освіти будь-яких навчальних закладів. Система надає можливість перевірити створені тести на якість і відповідність правилам тестології.

Зареєструвавшись, викладач отримує можливість: самостійно створювати тести, використовуючи п'ять типів тестових завдань; налаштовувати доступ до тестів певній групі користувачів; приєднатися до існуючих тестів, подавши заявку адміністратору; надавати іншим користувачам–викладачам змогу виконувати окремі дії в своїх тестах; бачити результати тестування кожного з учнів за умови їх реєстрації в системі; отримувати статистичні дані за результатами тестування певної групи за певний час.

Зареєструвавшись, здобувач може:

- проходити тестування як зареєстрований чи анонімний користувач;
- проходити тестування на запропонованих учителем чи відкритих тестах;
- бачити свої результати і визначити свій рівень серед решти результатів після закінчення тестування;
- проходити тестування неодноразово, якщо цей режим передбачений; тренуватися і перевірити свою готовність.

У контексті розвитку інтелектуальних систем електронного оцінювання знань студентів, існує значна кількість різноманітних рішень, які активно використовуються в освітніх установах по всьому світу. Ці системи мають різні рівні складності та адаптивності, від базових автоматизованих тестувальних платформ до більш комплексних інтерактивних рішень, які враховують індивідуальні особливості студентів і використовують методи штучного інтелекту та машинного навчання.

Автоматизовані тестувальні системи:

Один із видів інтелектуальних систем для оцінювання є автоматизовані тестувальні платформи, які дозволяють викладачам створювати та адмініструвати онлайн-тести. Такі системи часто базуються на стандартних алгоритмах оцінювання, де відповіді студентів автоматично перевіряються, а результати оцінюються згідно з заздалегідь визначеними критеріями. Одним з найбільш відомих рішень такого типу є платформи як Moodle, Blackboard, або Google Classroom.

Модель оцінювання в цих системах базується на стандартному підході, що оцінює лише правильність відповіді, без урахування додаткових факторів, таких як час виконання завдання чи швидкість реагування студента на завдання. Це значно обмежує їхню здатність адаптуватися до індивідуальних потреб студентів і не дозволяє врахувати їхній рівень підготовки або мотивацію до повторного тестування.

Адаптивні системи оцінювання:

З розвитком технологій в галузі штучного інтелекту, все більше уваги приділяється створенню адаптивних систем оцінювання, які можуть змінювати складність тестових завдань в залежності від результатів попередніх відповідей студента. Одним з прикладів таких рішень є система Adaptive Learning, яка активно використовується в освіті для підтримки персоналізованого підходу до навчання.

Системи адаптивного тестування здатні змінювати рівень складності питань на основі відповідей студента, що дає можливість здійснювати більш точну оцінку його знань. Наприклад, платформи як Knewton і Smart Sparrow застосовують такі підходи для адаптації навчального матеріалу та тестових завдань до індивідуальних потреб учнів. Вони також враховують поведінкові патерни студентів, щоб створити більш персоналізовану та ефективну навчальну середу.

Однак ці системи часто є дорогими у впровадженні та не завжди доступні для всіх освітніх установ. Крім того, хоча вони і враховують індивідуальний

рівень студентів, вони не завжди забезпечують повний зворотний зв'язок чи можливість для оперативного покращення результатів через повторне тестування.

Системи з елементами гейміфікації та мотивації

Останнім часом значну увагу приділяють інтеграції елементів гейміфікації в системи оцінювання знань. Це заохочує студентів до активного навчання та підвищує рівень мотивації, забезпечуючи інтерактивні елементи, змагання, бали, досягнення та навіть лідерборди. Такі рішення можна зустріти у платформах, як Quizlet, Kahoot, чи Duolingo, де студенти можуть не лише тестувати свої знання, але й отримувати додаткові винагороди за досягнуті успіхи.

Використання гейміфікації в системах оцінювання дозволяє створити більш інтерактивну та цікаву навчальну атмосферу, що сприяє підвищенню мотивації студентів. Однак ці рішення мають обмежену функціональність в плані адаптивності оцінювання та зворотного зв'язку. Гейміфікаційні елементи не завжди дозволяють забезпечити об'єктивність оцінки знань, оскільки вони більше орієнтовані на підтримку активності, ніж на точне вимірювання рівня знань студента.

Сучасні інтелектуальні системи для оцінювання знань (табл. 1) часто інтегрують технології машинного навчання для поліпшення процесу тестування та оцінювання. Такі системи здатні аналізувати великі обсяги даних про навчання студентів, на основі чого можуть здійснювати прогнозування щодо успішності в майбутньому тестуванні або виявляти слабкі місця у знаннях учнів.

Зокрема, системи, що використовують аналіз даних, можуть не лише оцінювати результати тестів, але й пропонувати рекомендації для подальшого навчання. Одним з таких рішень є **Coursera**, яка використовує алгоритми машинного навчання для створення персоналізованих навчальних траєкторій, а також аналізу прогресу студентів. Іншим прикладом є **Examity**, яка використовує технології для виявлення підозрілих поведінкових патернів під час онлайн-тестування. Це дозволяє значно знизити рівень шахрайства та недоброчесної поведінки, що є важливим аспектом в оцінці знань в умовах онлайн-навчання.

Таблиця 1.1 – Порівняння інтелектуальних систем оцінювання знань студентів

Характеристика	**Google Classroom**	**Moodle**	**Knewton**	**Examity**
Тип системи	Веб-платформа	LMS (Learning Management System)	Адаптивне навчання	Онлайн-тестування з верифікацією
Функціональність	Організація курсу, тести, зворотний зв'язок	Курси, тести, управління контентом	Персоналізовані навчальні траєкторії	Безпека тестування, виявлення шахрайства
Збір даних	Відвідуваність, оцінки	Результати тестів, активність студентів	Прогрес учнів, рівень знань	Час відповіді, поведінка студента
Аналіз даних	Оцінки, активність	Статистика, успішність	Персоналізація навчання	Перевірка часу, аналіз шахрайства
Зручність використання	Висока	Висока	Середня	Середня
Інтеграція з іншими системами	Google Apps, Google Meet	Платформи електронного навчання	Можливості API для інтеграцій	Інтеграція з LMS
Адаптивність тестування	Низька	Середня	Висока	Середня
Мотивування студентів	Низьке	Середнє	Високе	Середнє
Запобігання шахрайству	Немає	Немає	Немає	Високе

Хоча існуючі рішення для оцінювання знань студентів мають свої переваги, вони також стикаються з певними обмеженнями. Одним з основних викликів є недостатня адаптивність цих систем до індивідуальних потреб студентів. Наприклад, традиційні тестові системи часто не враховують такі аспекти, як час на повторне тестування, інтенсивність підготовки до тесту, а також мотивацію студента.

Інший виклик пов'язаний із забезпеченням високого рівня безпеки та захисту від шахрайства. Традиційні платформи часто не мають ефективних

інструментів для виявлення недоброчесної поведінки під час здачі тестів, що створює ризики для об'єктивності результатів.

Незважаючи на високий потенціал адаптивних та інтерактивних систем, значну роль у їхньому розвитку грають фінансові та технологічні обмеження. Впровадження таких систем потребує значних інвестицій, а також адаптації навчальних планів і методик до нових умов тестування.

1.5 Постановка задачі і план реалізації

Для реалізації поставленої задачі необхідно розробити інтелектуальну систему електронного оцінювання знань студентів, яка забезпечить автоматизацію процесу тестування, підвищить об’єктивність результатів та сприятиме оптимізації навчального процесу. Основною метою є створення адаптивної платформи для оцінювання знань, здатної враховувати індивідуальні особливості студентів та стимулювати їхню навчальну активність. Застосування таких систем сприятиме впровадженню сучасних технологій у навчальний процес, забезпечуючи обґрунтовану і комплексну оцінку академічних досягнень.

Практична реалізація цього проєкту буде складатися з кількох етапів:

1. **Розробка бази даних:**

- Збір та зберігання інформації про студентів, тести, їхні результати та історію проходження тестів.
- Створення таблиць і зв'язків для зберігання даних, таких як профілі студентів, питання тестів, результати, оцінки та інша необхідна інформація.
- Оптимізація структури бази даних для швидкої обробки запитів та забезпечення ефективного зберігання даних.

2. **Розробка алгоритмів оцінювання та адаптації:**

- Розробка адаптивних алгоритмів, які дозволяють змінювати складність тестів залежно від успішності студентів.
- Оцінка не тільки правильності відповідей, але й часу, який витрачається на виконання завдань, швидкості повторного тестування та інтенсивності підготовки.

- Створення механізмів адаптації тестів для кожного студента, щоб забезпечити індивідуальний підхід до оцінювання.

3. **Розробка інтерфейсу користувача (фронтенду):**

- Створення веб-платформи, на якій студенти можуть проходити тести, а викладачі — створювати нові завдання та оцінювати студентів.
- Реалізація зручного та інтуїтивно зрозумілого інтерфейсу для студентів та викладачів, включаючи візуалізацію результатів тестування, надання зворотного зв'язку та можливість коригування результатів.
- Інтерфейс має бути адаптований для різних пристроїв та зручний у використанні.

4. **Розробка серверної частини (бекенду):**

- Створення серверної частини системи для обробки запитів від користувачів, зберігання та обробки даних, взаємодії з базою даних.
- Розробка механізмів для аналізу результатів тестування та збереження даних про проходження тестів.
- Реалізація API для забезпечення взаємодії між фронтендом та бекендом, а також для інтеграції з іншими освітніми платформами.

5. **Механізми мотивації студентів:**

- Впровадження механізмів для стимулювання студентів до активного навчання, таких як система додаткових балів за швидке повторне тестування та покращення результатів (рис. 1.3).
- Визначення правил для надання додаткових балів або іншої форми заохочення за участь у повторних тестах та вивчення помилок.

6. **Запобігання шахрайству та забезпечення безпеки:**

- Впровадження механізмів запобігання недоброчесній поведінці, таких як контроль за часом виконання завдань та аномалії у поведінці студентів під час тестування.
- Забезпечення конфіденційності даних студентів та відповідність вимогам безпеки інформації.

Рисунок 1.3 Механізм мотивації

Для реалізації цієї системи будуть використовуватися сучасні веб-технології, зокрема:

- JavaScript для створення інтерактивних елементів на сторінці, обробки введених даних та взаємодії з сервером.
- HTML/CSS для розмітки та стилізації інтерфейсу, що зробить його зручним для користувачів.
- Python або PHP для серверної частини, обробки запитів та роботи з базою даних.
- SQL або NoSQL для роботи з базою даних та зберігання інформації про студентів, тести та результати.

Розробка даної інтелектуальної системи дозволить значно покращити процес тестування в навчальних закладах, зробити його більш об'єктивним та адаптованим до індивідуальних потреб кожного студента, що сприятиме підвищенню якості навчання та мотивації студентів.

1.6 Дерево функцій

Дерево функцій є ієрархічною структурою, яка наочно демонструє функціональну декомпозицію проєкту, де головна мета роботи є вершиною дерева, а гілки представляють функції та завдання, необхідні для досягнення цієї мети. Метод функціональної декомпозиції дозволяє розділити систему на окремі

елементи, що виконують певні функції, та розглядати кожен елемент як частину єдиної системи, що працює на досягнення загальної мети проєкту.

На рис. 1.4 наведено дерево функцій, яке відображає всі функції, що потрібно реалізувати для забезпечення ефективної роботи інтелектуальної системи електронної оцінки знань студентів.

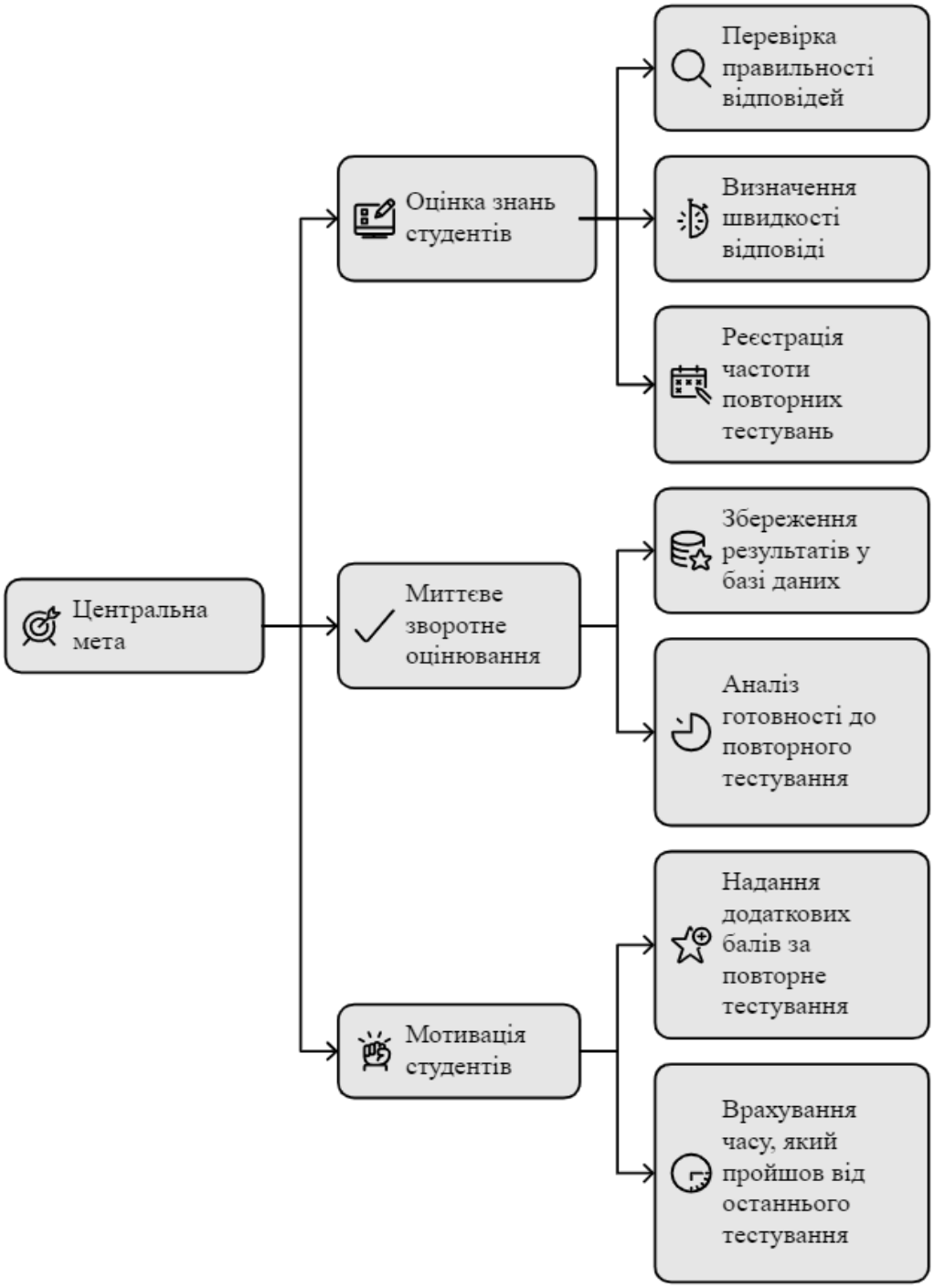

Рисунок 1.4 Дерево функцій

Це дерево структурує завдання проєкту на кілька рівнів, зокрема, основні функції системи, підфункції для кожного компонента та детальні підзавдання, що забезпечують виконання конкретних операцій.

Ієрархічна побудова дерева функцій допомагає деталізувати основну мету, виділити всі етапи, що є необхідними для її досягнення, та організувати процес роботи над проєктом. Така структура дозволяє команді зосередитись на окремих функціях, розподілити обов'язки та керувати процесом розробки на різних рівнях, підвищуючи ефективність роботи над системою.

Крім того, дерево функцій забезпечує зручність у плануванні та керуванні проєктом, оскільки дозволяє визначити пріоритети, взаємозв'язки між завданнями та узгодити порядок виконання робіт. Завдяки цьому інструменту можна впевнено слідувати досягненню головної мети, зважаючи на необхідність виконання всіх важливих функцій.

На рис. 1.4 графічно представлено головну мету як центральний вузол, а окремі гілки будуть включати такі функціональні елементи:

Основними функціями розробленої системи є оцінка знань студентів, миттєве зворотне оцінювання результатів та мотивація студентів до активного навчання. Кожна з цих функцій має свої підфункції, що дозволяють детальніше здійснювати оцінювання та підтримувати навчальний процес на високому рівні.

Перша основна функція – оцінка знань студентів, яка реалізується через кілька ключових підфункцій. Серед них базовою є перевірка правильності відповідей студентів, що дозволяє об’єктивно оцінити рівень засвоєння матеріалу. Важливим елементом цієї функції є аналіз швидкості відповіді, який допомагає оцінити когнітивну реакцію студента та його рівень готовності до тестування. Додатково система реєструє частоту повторних тестувань, що дозволяє отримувати дані про поведінкові аспекти навчання, такі як наполегливість, мотивація та готовність до вдосконалення своїх знань.

Друга основна функція – миттєве зворотне оцінювання результатів. Після завершення тестування система автоматично зберігає результати в базі даних, забезпечуючи прозорість процесу оцінювання та можливість подальшого аналізу.

Це включає відстеження динаміки змін у результатах кожного студента, що дозволяє визначити прогрес у навчанні. Крім того, система аналізує готовність студента до повторного тестування та адаптує зміст завдань на основі результатів попередніх спроб. Такий підхід дозволяє підтримувати індивідуалізований характер навчання.

Третя основна функція – мотивація студентів. Для цього впроваджено механізм надання додаткових балів студентам, які виконують тестування повторно впродовж короткого інтервалу часу. Це стимулює їх до негайного закріплення знань і вдосконалення навичок. Окрім цього, система враховує час, який минув від попереднього тестування, щоб оцінити ефективність підготовки студента та його здатність до довгострокового засвоєння матеріалу. Застосування таких механізмів сприяє створенню умов для постійного зростання академічної активності студентів.

Таким чином, розроблена система демонструє чітко структуровану взаємодію своїх компонентів, що забезпечує досягнення основних завдань: об’єктивності оцінювання, адаптивності до навчальних потреб студентів та підвищення їх мотивації. Інтеграція цих функцій у єдину платформу сприяє підвищенню ефективності освітнього процесу, створюючи інструмент для надійної та всебічної оцінки знань студентів. Особливістю розробленої системи є її здатність не лише виконувати базові завдання оцінювання, але й аналізувати поведінкові аспекти навчання.

Це дозволяє викладачам отримувати більш повну картину успішності студентів, включаючи такі фактори, як швидкість засвоєння матеріалу, адаптація до стресових умов тестування та здатність до самостійного покращення результатів. Інтеграція цих показників у загальну оцінку знань забезпечує комплексний підхід до аналізу навчальних досягнень, підвищуючи об'єктивність результатів.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Технічна платформа

Інтелектуальна система електронної оцінки знань студентів потребує високопродуктивної та надійної технічної платформи, яка забезпечує злагоджену взаємодію між клієнтськими та серверними компонентами, надаючи користувачам швидкий доступ до інтерфейсу, а також ефективне зберігання та обробку даних. Побудова цієї системи передбачає використання сучасних технологій як для серверної, так і для клієнтської частин. Крім того, для забезпечення масштабованості системи було реалізовано REST API, що спрощує інтеграцію з іншими системами, а також полегшує підтримку та розвиток функціоналу.

Серверна частина відповідає за обробку запитів від клієнтської частини, реалізацію бізнес-логіки, зберігання та обробку даних, а також забезпечує безпеку збереженої інформації. Основною мовою програмування для серверної частини системи обрано Python. Цей вибір обґрунтований кількома факторами:

- Легкість синтаксису та швидкість розробки: Python має простий, зрозумілий синтаксис, що дозволяє розробникам швидко писати і підтримувати код. Завдяки цьому, розробка функцій відбувається швидко, що є важливим для великих систем із широким функціоналом.
- Велика екосистема бібліотек і модулів: Python має розвинену екосистему бібліотек, таких як Django і Flask для розробки веб-додатків, NumPy і Pandas для обробки та аналізу даних, а також TensorFlow і scikit-learn для машинного навчання. Це дозволяє інтегрувати у систему елементи штучного інтелекту для подальшого вдосконалення адаптивності тестування та автоматизації оцінювання.
- Можливість інтеграції з іншими мовами та платформами: Python легко інтегрується з іншими мовами, що робить його універсальним для проєктів з різнорідною архітектурою.

Для зберігання даних у системі вибрано PostgreSQL – потужну реляційну систему керування базами даних, яка підтримує ACID-транзакції, що робить її надійною та безпечною. Використання PostgreSQL має ряд значних переваг:

- Стабільність та надійність: PostgreSQL відома своєю стабільністю, що є особливо важливим для систем з великим обсягом даних, оскільки забезпечує їх цілісність і захист від втрат при помилках.
- Гнучкість та розширюваність: підтримує створення складних запитів і функцій, що дозволяє легко налаштувати структуру бази даних під потреби конкретної системи. PostgreSQL також підтримує розширення, що надає можливість розробникам адаптувати систему під нові вимоги та специфікації.
- Широка підтримка мов програмування: PostgreSQL підтримує багато мов програмування, таких як Python, Java, C++, що дозволяє з легкістю інтегрувати її з веб-додатком і забезпечує кросплатформенність.

Клієнтська частина забезпечує користувацький інтерфейс для взаємодії зі студентами та викладачами. Для її розробки використовувалися наступні мови програмування та технології:

- HTML (HyperText Markup Language): використовується для створення структури сторінок, визначення текстового контенту, зображень, таблиць, форм тощо. HTML є основою клієнтського інтерфейсу, і його роль полягає у забезпеченні зручного відображення контенту.
- CSS (Cascading Style Sheets): відповідає за стилізацію сторінок, що дозволяє зробити інтерфейс зручним і привабливим для користувача. За допомогою CSS розробники можуть створювати адаптивний дизайн, що забезпечує коректне відображення інтерфейсу на різних пристроях і роздільних здатностях екранів.
- JavaScript: використовується для додавання динамічних елементів на веб-сторінках. JavaScript дозволяє забезпечити інтерактивність, зокрема, реагування на дії користувача (натискання кнопок, введення даних), виконання клієнтських обчислень, перевірку форм та інші функції, що полегшують взаємодію з веб-застосунком.

Крім того, для полегшення розробки та підвищення продуктивності було використано React – JavaScript бібліотеку для створення інтерактивних інтерфейсів. React надає такі переваги:

- Компонентний підхід: дозволяє розробникам розділяти інтерфейс на окремі незалежні компоненти, що спрощує процес розробки та подальше тестування.
- Висока продуктивність: React оптимізує роботу інтерфейсу завдяки віртуальному DOM, що значно зменшує час відгуку програми.
- Широка підтримка спільноти: з огляду на популярність React, існує велика кількість плагінів та додаткових бібліотек, що дозволяє додавати в проєкт нові функції без написання коду з нуля.

Для забезпечення взаємодії між клієнтською та серверною частинами було реалізовано REST API. Архітектура REST дозволяє здійснювати запити до серверу для отримання чи модифікації ресурсів за допомогою HTTP методів (GET, POST, PUT, DELETE). Основні переваги використання REST API включають:

- Кросплатформенність: REST API дозволяє комунікувати з клієнтськими додатками на різних платформах та пристроях, включаючи мобільні застосунки та браузери.
- Масштабованість: дозволяє розподілити навантаження між різними серверними інстанціями та легко інтегруватися з іншими сервісами.
- Підтримка кешування: REST API підтримує кешування відповідей, що знижує навантаження на сервер та покращує швидкість роботи додатку.

Використання REST API також дозволяє розширювати функціональність та підтримувати нові модулі без зміни архітектури клієнтської або серверної частин. Це сприяє більш гнучкому розвитку системи, дозволяючи зосередитися на покращенні користувацького досвіду та швидкості роботи.

Для забезпечення зручної роботи з базою даних було обрано **pgAdmin** – потужний інструмент управління PostgreSQL базами даних. pgAdmin надає зручний графічний інтерфейс, що дозволяє розробникам легко керувати базами

даних, виконувати SQL-запити, здійснювати моніторинг продуктивності та отримувати звіти про використання ресурсів.

На додаток до цього, для контролю версій та спільної роботи над кодом використовується **Git**, який дозволяє розробникам відстежувати зміни в коді та ефективно співпрацювати під час розробки системи. Інтеграція з платформою GitHub забезпечує централізоване зберігання репозиторію, що полегшує організацію командної роботи та дозволяє швидко вирішувати конфлікти у коді.

Оскільки система працює з персональними даними студентів та викладачів, важливо забезпечити надійний рівень захисту інформації. Для цього реалізовані такі заходи:

- Авторизація та аутентифікація: для доступу до системи використовуються механізми авторизації, що дозволяють користувачам отримувати доступ лише до дозволених функцій.
- Шифрування даних: всі дані, що передаються через мережу, шифруються за допомогою HTTPS протоколу, що забезпечує захист від перехоплення інформації.
- Регулярне оновлення та моніторинг системи: для забезпечення стабільної роботи сервери регулярно оновлюються, а також проводиться моніторинг мережевої активності для виявлення підозрілих дій.

Технічна платформа інтелектуальної системи електронної оцінки знань студентів базується на потужних сучасних інструментах і технологіях, що забезпечують ефективну роботу, надійність і безпеку. Поєднання Python, PostgreSQL, React та REST API дозволяє створити продуктивну систему, яка здатна масштабуватися відповідно до потреб навчального закладу і адаптуватися до нових викликів у сфері освітніх технологій.

2.2 Проєктування бази даних

Розробка бази даних для інтелектуальної системи електронної оцінки знань студентів є критичним етапом, який забезпечує ефективне зберігання, обробку та доступ до інформації. Для створення бази даних у цьому проєкті

використовується реляційна система керування базами даних PostgreSQL, що дозволяє забезпечити високу продуктивність, надійність та масштабованість.

Основні принципи проєктування

Під час проєктування бази даних враховувалися ключові аспекти для забезпечення швидкодії, надійності та зручності використання:

Нормалізація. Для уникнення надмірного дублювання даних база даних спроєктована відповідно до нормалізованої структури, що дозволяє оптимізувати обсяг збережених даних і підвищити ефективність запитів. Було застосовано принципи нормалізації до третьої нормальної форми, що зменшило ризик виникнення аномалій при оновленні даних.

Створення індексів. Для прискорення пошуку інформації були налаштовані індекси на полях, які часто використовуються для фільтрації або сортування даних, таких як ідентифікатори користувачів, дати проведення тестувань та результати оцінювання. Це дозволяє зменшити навантаження на систему та підвищити продуктивність під час виконання запитів.

Розділення таблиць. Зважаючи на великий обсяг даних, які буде обробляти система, було реалізовано розділення таблиць на основі логічних блоків: зберігання профілів студентів, інформація про тести, результати тестувань, а також метадані щодо часу проходження тестів та частоти повторних спроб. Це забезпечує логічну організацію даних і зручність для розробників під час виконання складних запитів та обробки великих обсягів даних.

Основні таблиці бази даних

1. Таблиця користувачів (Users): містить інформацію про студентів та викладачів, зокрема, їхні особисті дані (ім'я, прізвище), логін, пароль (у хешованому вигляді) та контактну інформацію. Ця таблиця є центральною для зберігання інформації про користувачів системи.

2. Таблиця тестів (Tests): зберігає дані про доступні тести, включаючи назву, опис, предмет, для якого призначений тест, та складність завдань. Ця таблиця є корисною для фільтрації тестів відповідно до потреб студентів та їхнього рівня знань.

3. Таблиця питань (Questions): містить дані про питання, що входять до складу кожного тесту, включаючи текст запитання, варіанти відповідей і правильну відповідь. Ця таблиця дозволяє системі під час тестування відображати користувачам питання з відповідями.

4. Таблиця відповідей (Answers): зберігає інформацію про відповіді, які студенти надали під час проходження тестів. Вона включає ідентифікатор студента, ідентифікатор питання, обрану відповідь і позначку, чи є відповідь правильною. Ці дані використовуються для аналізу ефективності навчання та оцінювання знань.

5. Таблиця результатів тестів (TestResults): зберігає інформацію про підсумкові результати кожного тесту, який пройшов студент. Вона містить дані про загальну кількість правильних відповідей, час проходження тесту, дату тестування та загальну оцінку. Це дозволяє відстежувати прогрес студента та надавати викладачам аналітичну інформацію.

6. Таблиця зворотного зв'язку (Feedback): дозволяє студентам і викладачам залишати відгуки про тести та систему в цілому. Це сприяє покращенню системи на основі зворотного зв'язку від користувачів.

Оптимізація бази даних

Для забезпечення ефективної роботи бази даних і швидкого доступу до даних були впроваджені кілька ключових оптимізацій. Зокрема, для зменшення часу виконання запитів і підвищення швидкості пошуку були створені індекси на основних полях, таких як ідентифікатори тестів, питань, логіни користувачів, а також на полях, що містять дати та результати. Така індексація дозволяє значно знизити час доступу до інформації і підвищити ефективність роботи системи. Щоб забезпечити цілісність даних і запобігти виникненню некоректних записів, застосовуються транзакції, особливо під час операцій додавання чи оновлення інформації в кількох таблицях. Це гарантує, що будь-яка зміна даних буде виконуватись атомарно, що дозволяє уникнути помилок у разі системних збоїв і підтримувати консистентність бази даних.

Також, для оптимізації обробки запитів і забезпечення швидкого доступу до даних, система використовує розподіл даних на логічні блоки. Наприклад, дані про відповіді на тести зберігаються в окремих таблицях, відокремлених від загальних результатів тестування. Такий підхід дозволяє уникнути перенасичення таблиць і значно зменшити час на обробку запитів, підвищуючи ефективність системи в цілому. Крім того, для підтримки високої швидкодії при зростанні обсягів даних впроваджено архівування старих записів. Дані, що втратили актуальність для поточного функціонування системи, переміщуються в архівну базу даних, що дозволяє зберігати лише необхідну інформацію в основній базі. Це не тільки оптимізує використання ресурсів системи, але й дозволяє підтримувати високий рівень продуктивності.

Таким чином, запропоновані методи оптимізації сприяють забезпеченню високої ефективності, цілісності та швидкодії роботи бази даних, що є критичним для успішного функціонування інтелектуальних систем оцінювання.

Забезпечення конфіденційності та безпеки даних

Оскільки система містить особисті дані студентів, важливою частиною проєктування бази даних є забезпечення захисту даних:

- Хешування паролів: паролі користувачів зберігаються у зашифрованому вигляді з використанням криптографічного алгоритму хешування, що унеможливлює їхнє пряме відновлення навіть у разі отримання несанкціонованого доступу до бази даних.
- Розмежування доступу: права доступу до бази даних налаштовані таким чином, що лише авторизовані користувачі мають доступ до відповідних частин системи. Це знижує ризик несанкціонованого доступу до конфіденційної інформації.
- Резервне копіювання: для запобігання втраті даних налаштовано автоматичне резервне копіювання, що дозволяє відновити базу даних у разі збоїв або апаратних помилок.

Правильна структура бази даних, розділення даних за логічними блоками, створення індексів та забезпечення безпеки особистої інформації дозволяють

системі функціонувати швидко, стабільно та надійно. PostgreSQL обрано як основну платформу завдяки її продуктивності та можливостям гнучкої інтеграції, що робить базу даних адаптивною до змінних вимог проєкту та здатною забезпечити масштабованість системи в майбутньому.

2.2.1 Створення концептуальної моделі

Під час створення концептуальної моделі інтелектуальної системи електронної оцінки знань студентів необхідно проаналізувати основні процеси, що відбуваються в системі, включаючи процеси оцінювання, відстеження результатів, управління доступом до тестів та надання зворотного зв'язку. Цей аналіз дозволяє виявити ключові функції системи та основні вимоги до зберігання й обробки інформації, які будуть відображені у концептуальній моделі бази даних.

Після аналізу основних процесів були визначені основні вимоги до системи:

1. Оцінювання знань студентів – забезпечення точного та швидкого оцінювання відповідей студентів у різних дисциплінах.
2. Управління тестовими завданнями – можливість створення, редагування та видалення тестів і питань.
3. Надання зворотного зв'язку студентам – миттєве інформування студентів про результати з можливістю перегляду своїх оцінок і статистики успішності.
4. Відстеження прогресу студентів – можливість зберігати дані про результати, дати проходження тестів і частоту повторних спроб для подальшого аналізу.
5. Захист даних – забезпечення безпеки особистих даних студентів і викладачів відповідно до стандартів конфіденційності.

Наступним кроком стало визначення основних сутностей системи та їх атрибутів, які відображають інформацію, що має бути збережена в базі даних:

- Користувачі (Users) – містить інформацію про користувачів системи, зокрема студентів та викладачів, включаючи їхні особисті дані (ім’я, прізвище), логін, пароль, тип користувача (студент чи викладач).
- Тести (Tests) – зберігає інформацію про тести, доступні в системі, включаючи їхні назви, дисципліни, рівень складності та статус доступності.
- Питання (Questions) – включає всі питання, які входять до тестів, та інформацію про можливі варіанти відповідей, правильну відповідь та складність питання.
- Відповіді (Answers) – зберігає дані про вибрані відповіді студентів для кожного питання, дозволяючи відстежувати, які відповіді були правильними.
- Результати тестів (TestResults) – містить підсумкові дані для кожного пройденого тесту, зокрема оцінку, кількість правильних відповідей, тривалість проходження тесту, дату та час проходження.

Після визначення основних сутностей були встановлені зв’язки між ними. Наприклад, студент (сутність "Користувач") може проходити кілька тестів (зв’язок один-до-багатьох з сутністю "Тести"), а кожен тест містить кілька питань (зв’язок один-до-багатьох між сутностями "Тести" та "Питання"). Сутність "Відповіді" має зв’язки з сутностями "Питання" і "Користувач", що дозволяє відстежувати, які відповіді вибирав конкретний студент для кожного запитання.

На основі цих сутностей і зв'язків була розроблена концептуальна модель бази даних, яка надає загальне уявлення про логічну структуру даних системи. Ця модель включає в себе таблиці для кожної сутності, а також зв’язки між ними, що забезпечує ефективну організацію та зберігання інформації про користувачів, тести, питання, відповіді та результати тестувань.

Під час розробки концептуальної моделі враховувалися стандарти і методології проєктування баз даних, що сприяє створенню якісної, надійної та зрозумілої структури даних. Це забезпечує можливість масштабування системи в майбутньому та спрощує інтеграцію нових функцій.

Завершальним кроком у створенні концептуальної моделі є валідація та перевірка на відповідність вимогам системи. Цей етап включає ретельний аналіз структури моделі, перевірку логічних зв’язків і коректності атрибутів кожної сутності. Можливе також залучення експертів з баз даних для оцінки якості моделі та внесення змін у разі виявлення недоліків.

Також була розроблена концептуальна модель, яка демонструє основні сутності системи, їхні зв’язки та атрибути, необхідні для ефективного функціонування системи електронної оцінки знань студентів.

2.2.2 Створення логічної моделі інформаційної системи онлайн готелю для тварин

Створення логічної моделі інтелектуальної системи електронної оцінки знань студентів є важливим етапом у розробці бази даних, оскільки вона визначає структуру і зв'язки між сутностями, необхідні для забезпечення ефективної роботи системи.

Логічна модель дозволяє зрозуміти взаємодію між даними, що зберігаються в базі даних, і як вони будуть використовуватися під час виконання операцій оцінювання та надання зворотного зв'язку студентам.

На початковому етапі створення логічної моделі були визначені основні сутності, необхідні для функціонування системи. До них належать:

1. Користувачі (Users) – сутність, що зберігає дані про користувачів системи, таких як студенти та викладачі. Кожен користувач має унікальний ідентифікатор, ім'я, прізвище, логін, пароль та тип (студент чи викладач). Основні атрибути: UserID, Name, Surname, Login, Password, UserType.
2. Тести (Tests) – містить інформацію про доступні тести, включаючи назву тесту, предмет, рівень складності та статус доступності. Ця сутність дозволяє створювати, редагувати та видаляти тести, а також відстежувати їх використання. Основні атрибути: TestID, Title, Subject, Difficulty, Status.
3. Питання (Questions) – сутність, що зберігає питання для тестів, а також варіанти відповідей і правильну відповідь. Питання пов’язані з певним

тестом і мають свій рівень складності. Основні атрибути: QuestionID, TestID, QuestionText, Options, CorrectAnswer.

4. Відповіді (Answers) – зберігає вибрані студентами відповіді на питання, дозволяючи оцінювати правильність кожної відповіді та зберігати її результат. Основні атрибути: AnswerID, UserID, QuestionID, SelectedOption, IsCorrect.

5. Результати тестів (TestResults) – містить інформацію про результати проходження тестів, включаючи кількість правильних відповідей, загальну оцінку, тривалість проходження тесту та дату. Ця сутність дозволяє відстежувати успішність студентів і надавати їм статистику. Основні атрибути: ResultID, UserID, TestID, Score, CompletionTime, DateTaken.

Після визначення сутностей встановлюються такі зв'язки між ними, як:

- Користувачі та Тести: студент (сутність "Користувач") може проходити кілька тестів, що представляє зв’язок "один-до-багатьох" між сутностями "Користувачі" та "Тести".
- Тести та Питання: кожен тест містить кілька питань, що також є зв’язком "один-до-багатьох" між сутностями "Тести" та "Питання".
- Питання та Відповіді: зв'язок між питаннями і відповідями дозволяє фіксувати обрану студентом відповідь для кожного питання, що дозволяє визначити правильність відповідей.
- Користувачі та Результати тестів: кожен студент має набір результатів тестів, які представляють історію його навчання та результати тестування.

На основі визначених сутностей і зв'язків була розроблена схема бази даних, що включає таблиці для кожної сутності та відношення між ними. Ця схема відображає логічну модель, що включає атрибути кожної сутності та зв’язки між ними. Кожна таблиця містить поля, що відповідають атрибутам сутностей, а також первинні та зовнішні ключі для забезпечення зв’язності між таблицями.

Логічна модель інформаційної системи допомагає зрозуміти, як різні компоненти системи взаємодіють один з одним, і значно спрощує процес

подальшої фізичної реалізації. Вона служить важливим інструментом для планування та реалізації функціональності системи, забезпечуючи можливість подальшого розширення.

Для забезпечення якісної логічної моделі використовувалися стандарти нормалізації бази даних, що дозволяють уникнути надлишкових даних і забезпечити ефективність системи. Використання нормалізації допомагає уникнути дублювання даних і зменшує ризик виникнення аномалій при оновленні бази даних.

На рисунку 2.5 представлена логічна модель бази даних із зв’язками між сутностями, що надає уявлення про взаємозв'язки компонентів системи.

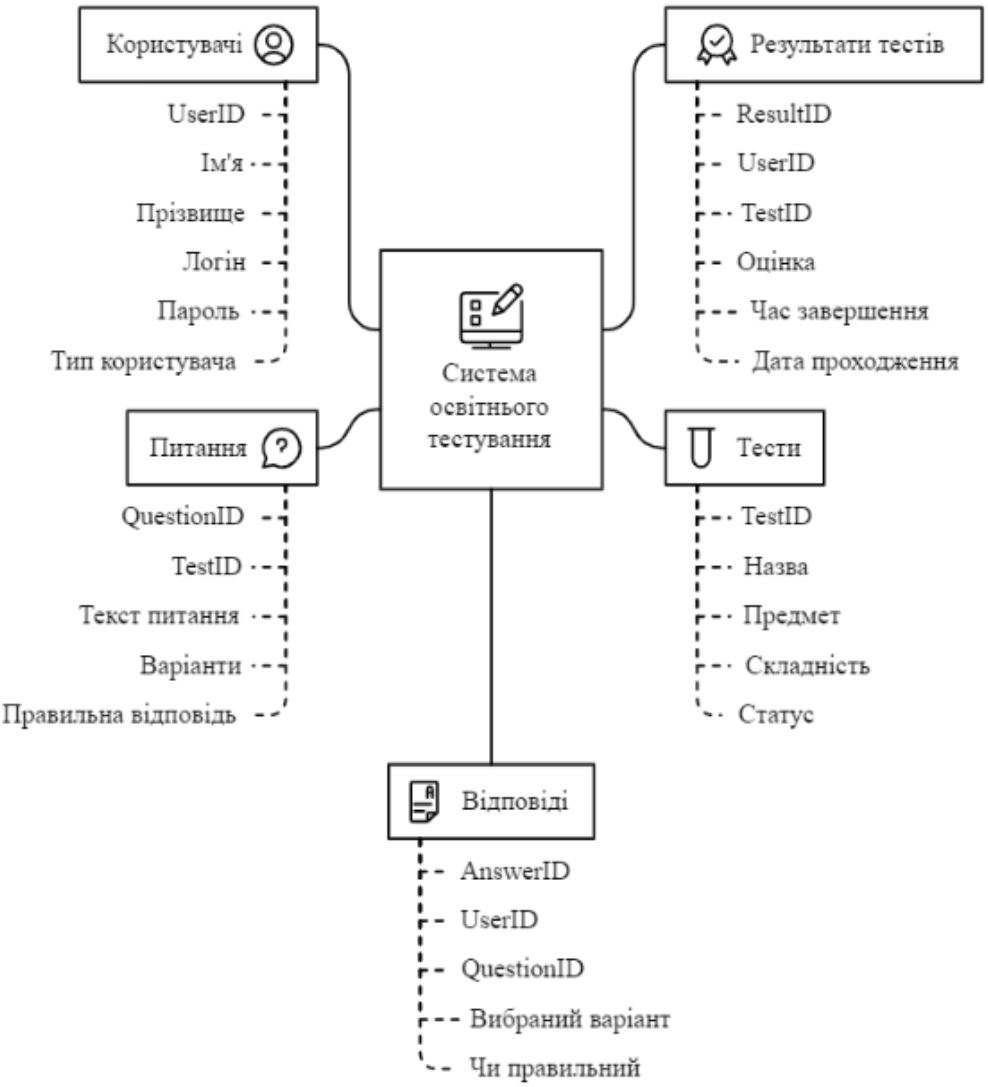

Рисунок 2.5 Логічна модель

Для візуалізації моделі та полегшення розуміння її структури використовувалася діаграма сутностей-зв’язків (ER-діаграма).

Після створення логічної моделі відбувається її валідація та перевірка на відповідність вимогам системи. Валідація включає аналіз коректності зв'язків, атрибутів та перевірку відповідності бізнес-вимогам. У разі необхідності, вносяться коригування, а також можуть залучатися експерти для оцінки якості моделі.

2.2.3 Створення фізичної моделі інформаційної системи

Створення фізичної моделі інтелектуальної системи електронної оцінки знань студентів є ключовим етапом реалізації бази даних, оскільки фізична модель визначає конкретну структуру таблиць, типи даних і технічні аспекти зберігання інформації. Фізична модель забезпечує ефективну та надійну роботу системи в умовах реального використання, а також дозволяє адаптувати її до вимог високої продуктивності та безпеки.

У процесі створення фізичної моделі важливо врахувати такі аспекти:

- Оптимізація продуктивності Після визначення сутностей встановлюються такі зв'язки між ними, як:

- Користувачі та Тести: студент (сутність "Користувач") може проходити кілька тестів, що представляє зв’язок "один-до-багатьох" між сутностями "Користувачі" та "Тести".

- Тести та Питання: кожен тест містить кілька питань, що також є зв’язком "один-до-багатьох" між сутностями "Тести" та "Питання".

- Питання та Відповіді: зв'язок між питаннями і відповідями дозволяє фіксувати обрану студентом відповідь для кожного питання, що дозволяє визначити правильність відповідей.

- Користувачі та Результати тестів: кожен студент має набір результатів тестів, які представляють історію його навчання та результати тестування.

На основі визначених сутностей і зв'язків була розроблена схема бази даних, що включає таблиці для кожної сутності та відношення між ними. Ця схема відображає логічну модель, що включає атрибути кожної сутності та

зв’язки між ними. Кожна таблиця містить поля, що відповідають атрибутам сутностей, а також первинні та зовнішні ключі для забезпечення зв’язності між таблицями.

Логічна модель інформаційної системи допомагає зрозуміти, як різні компоненти системи взаємодіють один з одним, і значно спрощує процес подальшої фізичної реалізації. Вона служить важливим інструментом для планування та реалізації функціональності системи, забезпечуючи можливість подальшого розширення.

Для забезпечення якісної логічної моделі використовувалися стандарти нормалізації бази даних, що дозволяють уникнути надлишкових даних і забезпечити ефективність системи. Використання нормалізації допомагає уникнути дублювання даних і зменшує ризик виникнення аномалій при оновленні бази даних.

Логічна модель бази даних із зв’язками між сутностями, що надає уявлення про взаємозв'язки компонентів системи:

- Оптимізація продуктивності – фізична модель повинна підтримувати оптимальну швидкість обробки запитів навіть при великому обсязі даних. Для цього створюються індекси на часто використовуваних полях, що зменшує час доступу до інформації.
- Масштабованість – структура бази даних повинна бути розроблена таким чином, щоб її можна було легко розширити або модифікувати при збільшенні обсягів даних чи потребі в додаткових функціях. Для цього використовуються нормалізація таблиць та зв'язки між ними.
- Захист даних – враховуючи роботу з особистими даними студентів, реалізуються заходи безпеки, включаючи шифрування паролів, контроль доступу та регулярне резервне копіювання даних. Це дозволяє захистити інформацію від несанкціонованого доступу та втрат.
- Стабільність і відмовостійкість – фізична модель передбачає заходи для забезпечення стабільної роботи системи, включаючи використання транзакцій

для забезпечення цілісності даних і налаштування механізмів відновлення у разі збою.

Структура фізичної моделі бази даних

Фізична модель бази даних розроблена на основі логічної моделі і включає конкретні визначення полів таблиць та типів даних:

1. **Таблиця користувачів (Users)**:
   - UserID (INT, PRIMARY KEY, AUTO_INCREMENT) – унікальний ідентифікатор користувача.
   - Name (VARCHAR) – ім'я користувача.
   - Surname (VARCHAR) – прізвище користувача.
   - Login (VARCHAR, UNIQUE) – унікальний логін користувача.
   - Password (VARCHAR) – хешований пароль користувача.
   - UserType (ENUM) – тип користувача (студент або викладач).
2. **Таблиця тестів (Tests)**:
   - TestID (INT, PRIMARY KEY, AUTO_INCREMENT) – ідентифікатор тесту.
   - Title (VARCHAR) – назва тесту.
   - Subject (VARCHAR) – предмет, до якого належить тест.
   - Difficulty (ENUM) – рівень складності.
   - Status (BOOLEAN) – статус доступності тесту.
3. **Таблиця питань (Questions)**:
   - QuestionID (INT, PRIMARY KEY, AUTO_INCREMENT) – ідентифікатор питання.
   - TestID (INT, FOREIGN KEY) – ідентифікатор тесту, до якого належить питання.
   - QuestionText (TEXT) – текст запитання.
   - Options (JSON) – варіанти відповідей.
   - CorrectAnswer (VARCHAR) – правильна відповідь.
4. **Таблиця відповідей (Answers)**:

- AnswerID (INT, PRIMARY KEY, AUTO_INCREMENT) – ідентифікатор відповіді.
- UserID (INT, FOREIGN KEY) – ідентифікатор студента.
- QuestionID (INT, FOREIGN KEY) – ідентифікатор питання.
- SelectedOption (VARCHAR) – обрана студентом відповідь.
- IsCorrect (BOOLEAN) – показує, чи є відповідь правильною.

5. **Таблиця результатів тестів (TestResults)**:
   - ResultID (INT, PRIMARY KEY, AUTO_INCREMENT) – ідентифікатор результату.
   - UserID (INT, FOREIGN KEY) – ідентифікатор студента.
   - TestID (INT, FOREIGN KEY) – ідентифікатор тесту.
   - Score (INT) – оцінка студента.
   - CompletionTime (TIME) – тривалість проходження тесту.
   - DateTaken (DATE) – дата проходження тесту.

Оптимізація фізичної моделі

Для покращення продуктивності та зменшення часу виконання запитів були застосовані наступні оптимізації:

Індексація Після визначення сутностей встановлюються такі зв'язки між ними, як:

- Користувачі та Тести: студент (сутність "Користувач") може проходити кілька тестів, що представляє зв’язок "один-до-багатьох" між сутностями "Користувачі" та "Тести".
- Тести та Питання: кожен тест містить кілька питань, що також є зв’язком "один-до-багатьох" між сутностями "Тести" та "Питання".
- Питання та Відповіді: зв'язок між питаннями і відповідями дозволяє фіксувати обрану студентом відповідь для кожного питання, що дозволяє визначити правильність відповідей.
- Користувачі та Результати тестів: кожен студент має набір результатів тестів, які представляють історію його навчання та результати тестування.

На основі визначених сутностей і зв'язків була розроблена схема бази даних, що включає таблиці для кожної сутності та відношення між ними. Ця схема відображає логічну модель, що включає атрибути кожної сутності та зв’язки між ними. Кожна таблиця містить поля, що відповідають атрибутам сутностей, а також первинні та зовнішні ключі для забезпечення зв’язності між таблицями.

Логічна модель інформаційної системи допомагає зрозуміти, як різні компоненти системи взаємодіють один з одним, і значно спрощує процес подальшої фізичної реалізації. Вона служить важливим інструментом для планування та реалізації функціональності системи, забезпечуючи можливість подальшого розширення.

Для забезпечення якісної логічної моделі використовувалися стандарти нормалізації бази даних, що дозволяють уникнути надлишкових даних і забезпечити ефективність системи. Використання нормалізації допомагає уникнути дублювання даних і зменшує ризик виникнення аномалій при оновленні бази даних.

Логічна модель бази даних із зв’язками між сутностями, що надає уявлення про взаємозв'язки компонентів системи.

- Індексація – створені індекси для полів, за якими часто здійснюються пошукові операції, таких як UserID, TestID та QuestionID. Це дозволяє швидко знаходити потрібні дані без повного сканування таблиць.
- Використання зовнішніх ключів – у таблицях реалізовано зв'язки між таблицями за допомогою зовнішніх ключів, що дозволяє підтримувати цілісність даних і забезпечує коректну взаємодію між таблицями.
- Контроль транзакцій – для критично важливих операцій, таких як додавання або оновлення даних, реалізовані транзакції. Це дозволяє забезпечити цілісність даних і уникнути часткових змін у разі виникнення помилок.

Інфраструктура системи

Для реалізації та підтримки фізичної моделі використовується сервер баз даних PostgreSQL, що забезпечує масштабованість та стабільну роботу під

навантаженням. Сервер налаштовано для роботи з великими обсягами даних, що забезпечує ефективність виконання запитів, а також підтримує захищені з'єднання для збереження конфіденційності даних.

2.3 Проєктування серверної частини

Проєктування серверної частини інтелектуальної системи електронної оцінки знань студентів є важливим етапом, що забезпечує основні функціональні можливості системи, обробку даних та їх надійне збереження. Серверна частина є проміжною ланкою між клієнтським інтерфейсом і базою даних, відповідаючи за обробку запитів користувачів, доступ до інформації та цілісність даних.

2.3.1 Архітектура серверної частини

Серверна частина системи побудована за принципом клієнт-сервер, де для взаємодії між сервером і клієнтом обрано архітектуру REST API. Використання цієї архітектури дозволяє забезпечити ефективне та зручне передавання даних за допомогою HTTP-запитів. Зокрема, REST API підтримує основні HTTP-методи, такі як GET, POST, PUT, DELETE, що дає змогу здійснювати операції отримання, додавання, оновлення та видалення даних. Така архітектура забезпечує високу надійність, гнучкість і сумісність з різними платформами, що робить систему кросплатформенною і дозволяє легко адаптувати її до різних умов та масштабувати у разі росту навантаження або збільшення кількості користувачів.

Основні компоненти архітектури серверної частини включають контролери, сервіси та модулі роботи з базою даних, кожен з яких виконує певну роль у забезпеченні стабільної, ефективної роботи системи. Контролери відповідають за прийом і обробку запитів від клієнтів, виконують відповідні дії над даними (реєстрація користувача, доступ до тестових матеріалів, збереження результатів тестів) і формують відповіді, що відправляються користувачеві. Кожен контролер спеціалізується на обробці певних типів запитів, що дозволяє розподілити навантаження та підвищити ефективність роботи сервера.

Сервіси реалізують бізнес-логіку системи, обробляючи запити, виконуючи складні операції з даними та підтримуючи необхідну функціональність системи.

Вони займаються обчисленням результатів тестування, оцінкою відповідей, створенням нових тестів, керуванням профілями користувачів та інше. Завдяки такому розподілу обов'язків, кожен компонент має чітко визначену роль, що підвищує зручність розробки та подальшої підтримки системи.

Модулі роботи з базою даних відповідають за взаємодію з базою даних, виконують операції збереження та отримання даних, а також здійснюють обробку запитів до таблиць. Це дозволяє забезпечити цілісність та консистентність даних, а також підвищити ефективність роботи з великими обсягами інформації. Використання таких модулів дозволяє централізовано обробляти запити до бази даних, автоматично забезпечуючи узгодженість даних і оптимізацію роботи з ними. Такий підхід сприяє створенню стабільної, зручної і масштабованої системи для обробки великих обсягів тестових даних і результатів.

2.3.2 Вибір технологій для серверної частини

Основною мовою програмування для розробки серверної частини вибрано Python, що в поєднанні з веб-фреймворком Flask забезпечує ефективну побудову REST API та організацію чіткої і зручної структури веб-додатка. Python відзначається легким для розуміння синтаксисом і багатою екосистемою бібліотек, що дозволяє прискорити процес створення веб-додатків. Flask є легким і модульним фреймворком, що надає значну гнучкість у налаштуванні серверної частини, дозволяючи адаптувати додаток під специфічні вимоги проєкту, забезпечуючи можливість ефективного масштабування та розширення серверної інфраструктури. Flask підтримує обробку асинхронних запитів, що дозволяє ефективно працювати з високими навантаженням і забезпечує масштабованість системи в умовах одночасної взаємодії великої кількості користувачів.

Для забезпечення безпеки серверної частини були впроваджені сучасні технології захисту. Зокрема, аутентифікація користувачів здійснюється за допомогою JWT (JSON Web Token), що дозволяє створювати захищені токени доступу для кожного користувача та забезпечувати безпечний обмін даними між клієнтом і сервером. Крім того, для зберігання паролів застосовується алгоритм

хешування (bcrypt), що дозволяє зберігати дані в зашифрованому вигляді, забезпечуючи захист від несанкціонованого доступу до конфіденційної інформації.

2.3.3 Логіка обробки запитів

Першим етапом є прийом запиту від клієнта через REST API. Сервер отримує запит і визначає тип операції, яку потрібно виконати (наприклад, отримання тесту, збереження результату або оновлення профілю користувача). Це дозволяє серверу визначити необхідні дії та вибрати відповідний маршрут для обробки.

Наступним етапом є перевірка прав доступу. На цьому етапі для кожного запиту здійснюється перевірка токена доступу, що дозволяє переконатися в тому, що користувач має необхідні права для виконання запитаної операції. Це є важливим елементом для забезпечення безпеки системи, гарантуючи, що лише авторизовані користувачі можуть здійснювати відповідні операції.

Після підтвердження прав доступу починається власне обробка запиту. Контролери викликають відповідні сервіси для виконання запитаної операції, наприклад, перевірку відповідей студентів, розрахунок оцінки чи інші операції, що стосуються обробки даних. Цей етап є основою для реалізації бізнес-логіки системи. У випадку необхідності доступу до бази даних, відбувається здійснення запиту на вибірку або збереження даних. Це може включати збереження результатів тестування в базі даних або отримання інформації, необхідної для формування відповіді на запит користувача (список доступних тестів).

Завершальним етапом є формування відповіді для клієнта. Сервер надсилає клієнту відповідь, яка може містити дані для відображення на інтерфейсі, такі як оцінка за тест або список доступних тестів. Цей етап забезпечує надання необхідної інформації користувачу у вигляді, який можна відобразити на веб-сторінці чи іншому інтерфейсі. Таким чином, вся логіка обробки запитів базується на чітко визначених етапах, що забезпечують ефективну і безпечну взаємодію з користувачем, зберігання даних та виконання необхідних операцій на сервері.

2.3.4 Забезпечення масштабованості та продуктивності

Для забезпечення високої продуктивності та можливості масштабування серверної частини системи були впроваджені кілька ключових стратегій. Однією з них є балансування навантаження, яке дозволяє ефективно розподіляти запити між кількома серверними інстанціями. Такий підхід забезпечує рівномірне навантаження на сервери, що дозволяє зменшити затримки та підвищити швидкість обробки запитів, особливо при високому навантаженні на систему.

Іншим важливим елементом є використання кешування, що дає змогу зменшити час відповіді на повторювані запити, такі як отримання списку тестів або іншої часто запитуваної інформації. За допомогою кешуючих механізмів вдається уникнути необхідності повторних звернень до бази даних для обробки однакових запитів, що значно підвищує ефективність роботи системи та зменшує навантаження на базу даних.

Також важливим аспектом є модульність структури серверної частини, яка була реалізована через розподіл на окремі компоненти, такі як контролери, сервіси та модулі для роботи з базою даних. Такий підхід дозволяє зручно управляти продуктивністю, здійснювати локальні оновлення та підтримку конкретних компонентів без впливу на всю систему, що покращує її гнучкість та здатність до масштабування.

2.3.5 Інтеграція з базою даних

Серверна частина системи інтегрується з базою даних PostgreSQL за допомогою бібліотеки SQLAlchemy, що є потужним інструментом для здійснення операцій доступу до бази даних через механізм ORM (Object-Relational Mapping). Використання SQLAlchemy дозволяє значно спростити процес роботи з даними, забезпечуючи збереження об’єктно-орієнтованої структури даних у Python-коді та дозволяючи розробникам працювати з об’єктами, замість написання складних SQL-запитів вручну. Це підвищує ефективність та зручність розробки, одночасно забезпечуючи гнучкість у роботі з різноманітними типами даних.

Однією з основних переваг використання SQLAlchemy є її здатність захищати систему від SQL-ін’єкцій. Завдяки автоматичному обробленню запитів, SQLAlchemy надає рівень абстракції, що дозволяє уникнути безпосереднього втручання в SQL-код, запобігаючи таким чином несанкціонованому доступу до бази даних через ін’єкції. Це є важливим аспектом з точки зору безпеки, оскільки захист від SQL-ін’єкцій є одним з основних способів забезпечення стабільності та надійності системи. Крім того, SQLAlchemy підтримує транзакції, що є важливою функціональністю при роботі з критичними операціями, такими як запис результатів тестів. За допомогою транзакцій система забезпечує цілісність даних, гарантує, що всі операції в межах транзакції будуть виконані атомарно, і в разі помилки будь-яка зміна даних може бути відкотена, не порушуючи цілісності бази даних.

Ще однією важливою перевагою SQLAlchemy є її здатність працювати з великими обсягами даних. SQLAlchemy автоматично оптимізує виконання запитів, що дозволяє ефективно працювати з великими обсягами інформації без значних затримок у роботі системи. Бібліотека використовує різноманітні стратегії кешування та оптимізації, що дозволяють мінімізувати навантаження на сервер і базу даних, підвищуючи швидкість обробки запитів.

Загалом, використання SQLAlchemy в контексті інтеграції серверної частини з базою даних PostgreSQL дозволяє значно полегшити процес розробки, підвищити безпеку та забезпечити ефективну обробку великих обсягів даних. Крім того, SQLAlchemy забезпечує ефективну обробку великих обсягів даних завдяки оптимізації запитів і можливості масштабування. Завдяки ORM-підходу розробники можуть працювати з базою даних на високому рівні абстракції, маніпулюючи об’єктами Python замість написання складних SQL-запитів. Гнучкість і універсальність цього інструменту дозволяють легко адаптуватися до змін у моделі бази даних, інтегрувати нові функції та забезпечувати стабільність роботи серверної частини навіть у системах з високим навантаженням. Усе це робить SQLAlchemy одним із найпотужніших інструментів для створення сучасних, масштабованих і безпечних додатків.

3 РОЗРОБКА ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ЕЛЕКТРОННОГО ОЦІНЮВАННЯ ЗНАНЬ

3.1 Архітектура програмного забезпечення системи

Архітектура програмного забезпечення базується на моделі клієнт-сервер, що є найпоширенішою для веб-додатків. Ця модель дозволяє розділити систему на два основні компоненти: клієнтську частину (інтерфейс користувача) і серверну частину (обробка даних і управління логікою додатку).

Клієнтська частина – це інтерфейс, з яким взаємодіє користувач. Вона реалізована за допомогою HTML, CSS і JavaScript, що забезпечує інтуїтивно зрозумілу навігацію та зручність використання системи. Клієнтська частина відповідає за відображення питань, прийом відповідей від студентів, демонстрацію результатів і надання миттєвого зворотного зв'язку. Система підтримує адаптивний дизайн, що дозволяє користувачам комфортно користуватися системою з різних пристроїв, включаючи смартфони, планшети та комп'ютери.

Серверна частина – відповідає за обробку запитів від клієнтської частини, виконує бізнес-логіку та забезпечує доступ до бази даних. Серверна частина побудована на Flask, легкому фреймворку Python, що підтримує REST API для забезпечення взаємодії між клієнтом і сервером. Завдяки використанню Flask, розробники можуть створювати масштабовані та гнучкі веб-додатки з простою архітектурою.

REST API – використовується для забезпечення комунікації між клієнтом та сервером за допомогою HTTP-запитів. REST API дозволяє здійснювати операції зі створення, отримання, оновлення та видалення ресурсів. Всі функції, пов'язані з тестуванням, обробкою відповідей та наданням результатів, виконуються через маршрути REST API, що робить систему легкою для інтеграції з іншими сервісами або для подальшого розширення.

База даних PostgreSQL – обрана для забезпечення ефективного зберігання і обробки даних. PostgreSQL – потужна реляційна база даних з відкритим кодом, яка підтримує ACID-транзакції, що гарантує цілісність даних. База даних

структурована таким чином, щоб зберігати інформацію про користувачів, питання, відповіді, результати тестувань та інші метадані. Структура бази даних була спроєктована з урахуванням нормалізації даних для уникнення дублювання і забезпечення ефективного доступу до інформації.

Компоненти безпеки – включають JSON Web Token (JWT) для аутентифікації користувачів та шифрування паролів з використанням bcrypt. JWT дозволяє здійснювати аутентифікацію користувачів без зберігання сесій на сервері, що підвищує продуктивність і безпеку системи. Шифрування паролів з bcrypt гарантує, що навіть при компрометації бази даних паролі залишаться недоступними в оригінальному вигляді.

3.2 Реалізація серверної частини системи

Реалізація серверної частини забезпечує управління функціональністю системи, обробку даних і доступ до бази даних.

Основна роль серверної частини – обробка HTTP-запитів від клієнтської частини, обчислення результатів тестування та збереження даних у базі.

3.2.1 Створення REST API для взаємодії клієнта з сервером

REST API забезпечує зв’язок між клієнтом і сервером, дозволяючи клієнтам надсилати запити для отримання або зміни даних. Основні маршрути API включають:

Маршрути для аутентифікації:

POST /api/login – перевіряє логін і пароль користувача, а у випадку успішної аутентифікації генерує JWT для подальших запитів. У відповідь повертається токен, який клієнт використовує для отримання доступу до захищених маршрутів.

POST /api/register – дозволяє новим користувачам реєструватися у системі. Запит включає дані користувача (ім’я, прізвище, логін і пароль), які зберігаються у базі даних у зашифрованому вигляді.

Маршрут для отримання питань тесту:

GET /api/questions – забезпечує вибірку питань для поточного тесту. Сервер на основі запиту від клієнта вибирає набір питань, згенерований випадковим чином, та повертає їх у форматі JSON.

Маршрут для збереження результатів:

POST /api/submit – приймає відповіді студента, обчислює загальний результат та зберігає його у базі даних. Кожна відповідь перевіряється на правильність, а до підсумкової оцінки можуть додаватися бонуси.

Код реалізації одного з маршрутів, наприклад, для отримання питань тесту:

```
from flask import Flask, request, jsonify
from flask_jwt_extended import jwt_required, get_jwt_identity
from database import get_test_questions
app = Flask(__name__)
@app.route('/api/questions', methods=['GET'])
@jwt_required()
def get_questions():
    user_id = get_jwt_identity()
    questions = get_test_questions(user_id)
    return jsonify(questions)
```

3.2.2 Реалізація модулів обробки даних та алгоритмів оцінювання

Обробка відповідей студента і розрахунок оцінки – ключовий аспект серверної частини. Логіка оцінювання включає наступні етапи:

Обробка відповідей. Кожна відповідь перевіряється на правильність. Кожна правильна відповідь оцінюється згідно з попередньо встановленими правилами.

Розрахунок загальної оцінки. Загальний результат обчислюється як сума балів за правильні відповіді. При необхідності до результату додаються бонуси, залежно від часу завершення тесту та кількості спроб.

Збереження даних. Після обчислення підсумкової оцінки результати тесту зберігаються у базі даних разом з метаданими (дата проходження, тривалість, наявність бонусів тощо).

3.3 Реалізація клієнтської частини системи

Клієнтська частина інтелектуальної системи електронного оцінювання знань забезпечує зручний, інтуїтивний і привабливий інтерфейс, що дозволяє студентам легко взаємодіяти з системою під час проходження тестів і перегляду результатів. Інтерфейс адаптований під різні пристрої, що робить його доступним з комп'ютерів, планшетів і смартфонів, забезпечуючи однаково зручне користування на всіх екранах.

3.3.1 Дизайн інтерфейсу користувача

Дизайн інтерфейсу розроблений з акцентом на простоту, мінімалізм і доступність. Основний стиль інтерфейсу вибрано так, щоб він не відволікав користувачів, зберігав чіткість і логіку розташування елементів і при цьому залишався привабливим. Для стилізації використані сучасні стандарти HTML5 і CSS3, а для забезпечення динамічності й інтерактивності – JavaScript.

1. Головна сторінка (рис. 3.6):

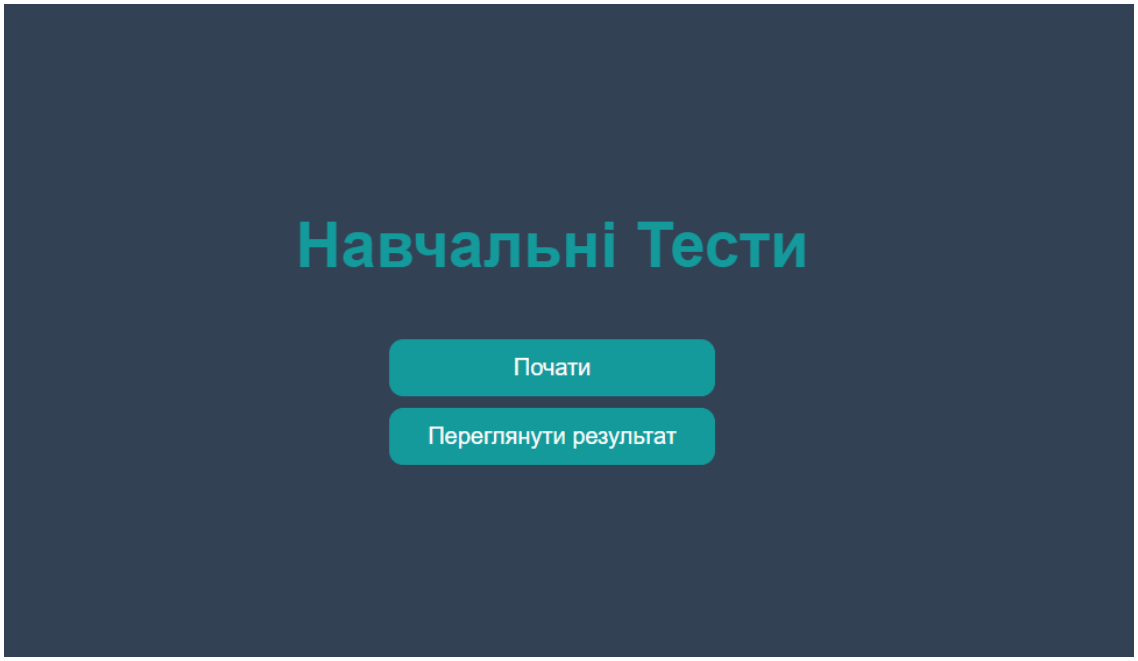

Рисунок 3.6 Початкова робоча область

На головній сторінці користувач має доступ до двох основних функцій – початку тестування і перегляду результатів. Кнопки виконані у вигляді великих, добре помітних елементів, що дозволяє з легкістю обрати потрібну дію.

Структура сторінки включає заголовок "Навчальні тести" і дві кнопки для переходу на інші сторінки.

```
<div id="home" class="flex-center flex-column">
    <h1>Навчальні Тести</h1>
    <a class="btn" href="game.html">Почати</a>
    <a class="btn" href="highscores.html">Переглянути результат</a>
</div>
```

2. Сторінка тестування (рис. 3.7):

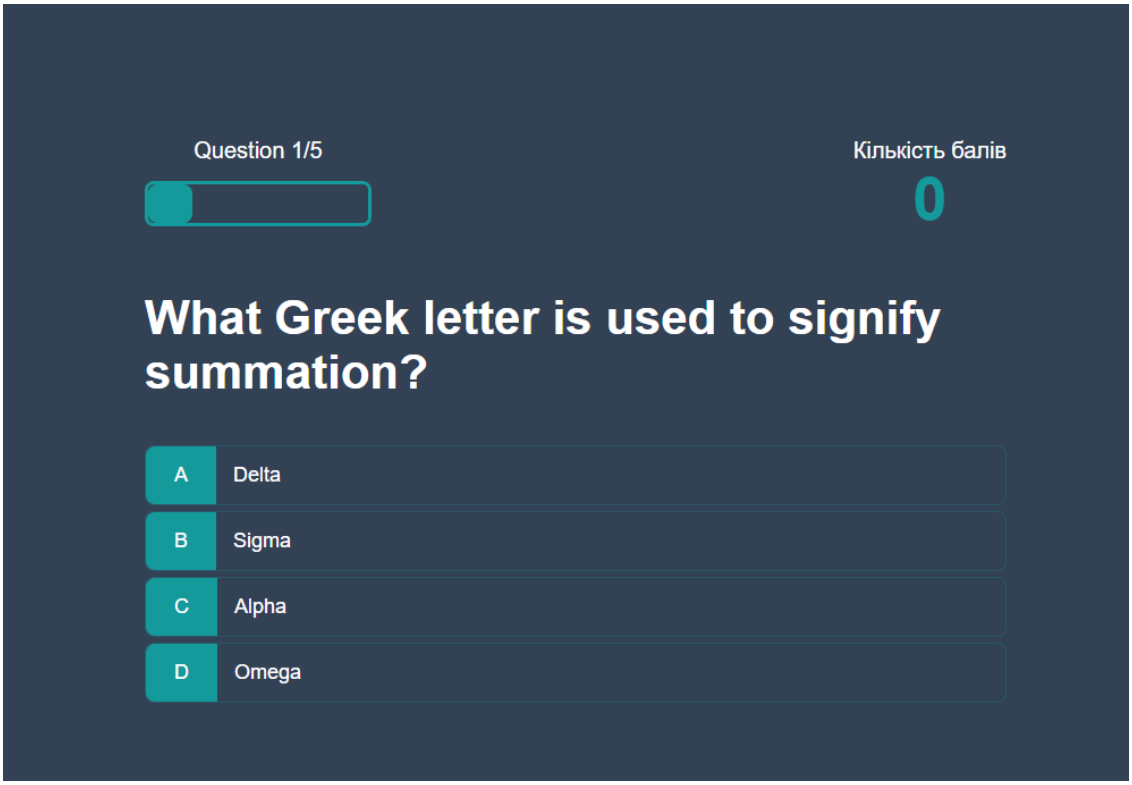

Рисунок 3.7 Робоча область проходження тестів

На цій сторінці студент проходить тест, вибираючи один із запропонованих варіантів відповідей. Інтерфейс передбачає чітке відображення питання, а також чотири кнопки з відповідями, кожна з яких має позначення (A, B, C, D) для зручності. Реалізована функція прогресу, яка візуально показує, скільки питань уже було пройдено. Це допомагає користувачам тримати під контролем свій прогрес і час.

```
div id="game" class="justify-center flex-column hidden">
    <h2 id="question"></h2>
```

```
    <div class="choice-container">
        <p class="choice-prefix">A</p>
        <p class<="choice-text" data-number="1"></p>
    </div>
    <!-- Інші варіанти відповідей B, C, D -->
</div>
```

3. Сторінка результатів (рисю 3.8):

Рисунок 3.8 Сторінка результатів

Після завершення тесту студенту відображаються підсумкові результати, які включають загальний бал і час завершення. Студент має можливість зберегти результати або повернутися на головну сторінку для повторного тестування. Вся інформація про тестування зберігається у локальному сховищі, що дозволяє уникнути необхідності збереження на сервері та забезпечує швидкий доступ до результатів.

```
<div id="end" class="flex-center flex-column">
    <h1 id="finalScore"></h1>
    <form>
```

```
            <input type="text" name="username" id="username"
placeholder="username"/>
            <button class="btn" id="saveScoreBtn" onclick="saveHighScore(event)"
disabled>Зберегти результат</button>
        </form>
    </div>
```

3.3.2 Реалізація функціональних модулів для студентів та викладачів

Для забезпечення всього функціоналу клієнтської частини були реалізовані наступні модулі:

1. Модуль тестування. Основний модуль системи, який дозволяє студентам вибирати відповіді на питання, а також здійснює обробку цих відповідей. Він відстежує правильність вибраної відповіді, зберігає підсумковий результат і відображає наступне питання. Логіка роботи включає обробку натискання на варіант відповіді, зберігання балів і прогресу.

```
    choices.forEach((choice) => {
        choice.addEventListener('click', (e) => {
            if (!acceptingAnswers) return;
            acceptingAnswers = false;
            const selectedChoice = e.target;
            const selectedAnswer = selectedChoice.dataset['number'];
            const classToApply = selectedAnswer == currentQuestion.answer ?
'correct' : 'incorrect';
            if (classToApply === 'correct') {
                incrementScore(CORRECT_BONUS);
            }
            setTimeout(() => {
                getNewQuestion();
            }, 1000);
        });
    });
```

2. Модуль перегляду результатів. Цей модуль надає можливість студентам переглядати свої попередні результати, а викладачам – бачити загальну успішність студентів. У модулі реалізовано збереження підсумкових балів та їх сортування для відображення найкращих результатів. Викладач може переглянути детальну статистику по кожному студенту.

3. Модуль збереження результатів. Зберігає результати тестування для подальшого перегляду. Використовує локальне сховище для збереження і швидкого доступу до інформації, дозволяючи користувачам швидко переглядати свої бали без потреби у повторному запиті до серверу.

3.4 Забезпечення безпеки даних у системі

Захист даних є пріоритетним завданням, особливо зважаючи на обробку особистої інформації студентів. Забезпечення безпеки даних передбачає комплексний підхід, що включає шифрування, аутентифікацію, контроль доступу, захист від SQL-ін’єкцій і регулярне резервне копіювання даних.

3.4.1 Методи шифрування та захисту паролів

Для шифрування паролів студентів і викладачів використовувався алгоритм bcrypt, що створює хеші паролів, які надзвичайно важко зламати. Зберігання паролів у хешованому вигляді запобігає їх розшифровці у разі компрометації бази даних. bcrypt має кілька важливих переваг:

1. Стійкість до атак типу brute-force: bcrypt має параметр cost, що визначає кількість ітерацій хешування, збільшуючи час, необхідний для генерації одного хешу. Це значно ускладнює використання brute-force для розшифрування паролів.

2. Запобігання використанню rainbow tables: bcrypt додає унікальну "сіль" до кожного пароля перед хешуванням, що робить використання попередньо обчислених таблиць для зворотного пошуку неможливим.

Приклад коду для хешування пароля за допомогою bcrypt:

```
from flask_bcrypt import Bcrypt
```

```
bcrypt = Bcrypt()
password = "user_password"
hashed_password = bcrypt.generate_password_hash(password).decode('utf-8')
```

3.4.2 Розмежування прав доступу користувачів

У системі реалізовано чіткий контроль доступу для забезпечення захисту даних студентів і викладачів. Кожен користувач отримує певний рівень доступу, який визначає, до яких функцій і даних він має доступ:

Аутентифікація за допомогою JSON Web Token (JWT): Для захисту сесій користувачів та їх авторизації використовується JWT. При вході в систему користувач отримує токен, який надсилається з кожним запитом. Це забезпечує захист від CSRF-атак і гарантує, що тільки авторизовані користувачі мають доступ до певних функцій.

Розподіл доступу. Студенти мають доступ лише до власних результатів, тоді як викладачі можуть переглядати всі результати, що дозволяє їм аналізувати успішність студентів. Токени авторизації зберігаються на стороні клієнта в локальному сховищі і передаються з кожним запитом до сервера для перевірки автентичності.

Захист від SQL-ін’єкцій. Усі запити до бази даних побудовані із використанням ORM (Object-Relational Mapping) або параметризованих запитів. Це захищає систему від SQL-ін’єкцій, оскільки користувацькі введення не вставляються безпосередньо у SQL-код.

Логування і моніторинг. Система веде журнал усіх дій користувачів, включаючи спроби доступу до захищених даних. Це допомагає відстежувати підозрілі дії та запобігати можливим загрозам.

3.5 Верифікація функціональності системи

Верифікація функціональності є важливим етапом у розробці та впровадженні системи, оскільки дозволяє переконатися в коректній роботі кожного її компонента і всього комплексу загалом. Під час верифікації використовувались різні методи тестування, спрямовані на перевірку основних

функцій системи, її продуктивності під навантаженням, інтеграції компонентів і дотримання вимог безпеки. Проведення таких тестувань є необхідним для забезпечення високої якості системи та її надійності.

Функціональне тестування

Функціональне тестування спрямоване на перевірку коректності виконання основних функцій системи відповідно до вимог технічного завдання. Усі основні операції, такі як тестування студентів, обчислення результатів, зберігання даних і їх перегляд, були протестовані на відповідність очікуваним результатам.

Тестування модулю проходження тестів: цей модуль є центральним у системі, тому його функціонування перевірялося особливо ретельно. Було протестовано відображення запитань, вибір відповідей, логіку переходу до наступного запитання після вибору варіанта відповіді, а також коректність обчислення балів за тест. Для цього використовувалися попередньо визначені запитання з відомими правильними відповідями, що дозволило перевірити правильність обробки даних.

Перевірка логіки нарахування бонусних балів: для підвищення мотивації студентів до швидкого проходження тестів система використовує систему бонусних балів. Було проведено тестування різних сценаріїв, у яких студенти завершували тест у різний час або проходили його повторно через певний проміжок часу. Система коректно нараховувала бонуси відповідно до правил, закладених у коді.

Модуль перегляду результатів: перевірявся функціонал відображення результатів для студентів. Зокрема, тестувалося відображення підсумкових балів, дати проходження тесту та інших метаданих. Також перевірялося, чи мають студенти доступ лише до власних результатів, що є важливим з точки зору безпеки та конфіденційності.

Тестування обробки помилок: для перевірки стійкості системи до можливих помилок було проведено ряд негативних тестів. Наприклад, система обробляла ситуації, коли користувач не вибрав відповідь, спробував повторно

пройти тест раніше дозволеного часу або намагався отримати доступ до результатів іншого студента. В усіх випадках система правильно обробляла помилки, повертаючи відповідні повідомлення про помилки.

Інструменти для функціонального тестування:

- Postman – для перевірки коректності роботи API-запитів.
- Selenium – для автоматизації тестів на рівні користувацького інтерфейсу, що дозволило імітувати поведінку реального користувача та перевірити правильність взаємодії з елементами інтерфейсу.

Тестування на навантаження було проведено для оцінки здатності системи витримувати значне навантаження від великої кількості одночасних користувачів. Це дозволило перевірити стабільність роботи системи та її продуктивність, зокрема під час пікових навантажень.

Тестування пікових навантажень. Було проведено моделювання сценарію, коли велика кількість студентів одночасно проходить тестування. У цей час система обробляла численні запити до серверу для отримання запитань, зберігання відповідей та обчислення результатів. Результати показали, що система здатна обробляти до 1000 одночасних запитів на секунду без значних затримок у роботі. Завдяки цьому підтверджено стійкість системи до високих навантажень.

Аналіз часу відповіді на запити. Система проходила тестування для визначення часу відповіді на різні типи запитів. Запити на отримання питань, надсилання відповідей і обробку результатів виконувалися за час, що не перевищував 500 мс. У разі збільшення кількості одночасних запитів до 2000 час відповіді зріс, однак залишався в межах 1 секунди, що є прийнятним для системи такого типу.

Тестування з використанням сценаріїв пікового навантаження. Було змодельовано сценарії, коли більшість студентів проходять тест у першій половині дня, а потім система залишається менш завантаженою. Такі сценарії дозволили підтвердити, що система ефективно справляється із піковими

навантаженнями й не створює затримок у роботі при зниженні кількості користувачів.

Робота кешування. Для зниження навантаження на сервер і підвищення продуктивності було налаштовано кешування для часто запитуваних даних. Було перевірено, що кешування запитів на отримання результатів дозволило знизити загальний обсяг запитів до бази даних і значно зменшити час відповіді для користувача.

Інструменти для тестування навантаження:

- JMeter – дозволив змоделювати високе навантаження на систему та проаналізувати її стійкість.
- Locust – інструмент для тестування, що дозволяє створювати сценарії навантаження для імітації великої кількості користувачів.

Інтеграційне тестування було проведено для перевірки взаємодії між різними компонентами системи, зокрема клієнтською частиною, сервером і базою даних. Це тестування дозволило переконатися, що всі компоненти системи правильно обмінюються даними та функціонують узгоджено.

Тестування зв'язку між клієнтом і сервером: було перевірено коректність передачі даних від клієнтської частини до серверної, зокрема коректне надсилання відповідей і отримання результатів. Використання REST API дозволило забезпечити злагоджену роботу між клієнтською та серверною частинами.

Перевірка роботи з базою даних. Інтеграційне тестування включало перевірку коректності взаємодії між серверною частиною та базою даних. Усі операції зі збереження, оновлення та вибірки даних виконувалися відповідно до очікувань, що свідчить про коректну інтеграцію з базою даних PostgreSQL.

Тестування безпеки даних при передачі. Було перевірено передачу даних між клієнтом і сервером за допомогою HTTPS, що забезпечує захист даних від перехоплення під час передачі. Додатково була перевірена коректна передача токенів автентифікації (JWT), що використовуються для контролю доступу до захищених маршрутів.

Взаємодія компонентів при обробці помилок. Інтеграційне тестування включало перевірку реакції системи на різні помилки, такі як помилковий запит, відсутність доступу до бази даних або порушення зв'язку між клієнтом і сервером.

Перевірка стабільності інтеграції. Було проведено тривале тестування з великою кількістю запитів, що дозволило переконатися в стабільності взаємодії компонентів і їхньої здатності справлятися з постійним навантаженням.

Інструменти для інтеграційного тестування:

- Postman – для ручного тестування і перевірки коректності виконання API-запитів.
- Selenium WebDriver – для автоматизації тестів взаємодії з користувацьким інтерфейсом, що дозволило перевірити злагодженість роботи клієнтської та серверної частин.
- Pytest – для автоматизації тестів серверної логіки, а також забезпечення верифікації обробки даних на рівні сервера.

4 ВПРОВАДЖЕННЯ, ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

4.1 Тестування системи

4.1.1 Функціональне тестування

Функціональне тестування зосереджене на перевірці кожного компонента системи окремо для того, щоб підтвердити, що кожна функція відповідає вимогам, визначеним на етапі проектування. Функціональне тестування дозволяє переконатися, що система виконує свої завдання коректно.

Перевірка авторизації та реєстрації. Однією з ключових функцій системи є реєстрація нових користувачів і подальша авторизація для отримання доступу до тестування. Було проведено тестування для виявлення можливих проблем при обробці введених даних, таких як невірний формат пароля, зайнятий логін тощо. Тести показали, що система коректно обробляє такі ситуації, видаючи відповідні повідомлення.

Перевірка процесу тестування. Цей етап тестування охоплював правильність відображення питань, вибір відповіді, збереження обраного варіанту та обробку результатів. Кожне питання мало бути доступне для перегляду і вибору варіанту, після чого система переходила до наступного питання або показувала підсумковий результат. Було протестовано різні сценарії, зокрема перерви між питаннями, час проходження тесту та випадковий вибір відповідей, що дозволило впевнитись у коректній роботі системи.

Перевірка відображення прогресу тестування. Важливо, щоб студенти могли стежити за своїм прогресом у процесі проходження тесту. Було протестовано відображення загального прогресу і правильність обчислення кількості балів у режимі реального часу. Це дозволило перевірити, чи система коректно оновлює прогрес та бал після кожної відповіді.

Перевірка збереження результатів. Після завершення тесту результати мали бути збережені у базі даних та доступні для перегляду студентами і викладачами. Було перевірено, чи відображаються результати, які зберігаються, і чи вони доступні для подальшого аналізу викладачами.

Перевірка функцій адміністратора. Оскільки викладачі мають доступ до повної інформації про результати тестування, було проведено тестування інтерфейсу адміністратора та його функцій для управління контентом тестів. Викладачі мали змогу отримувати статистику щодо успішності студентів, що дозволяє їм відстежувати прогрес навчання.

Адаптивність дизайну. Клієнтська частина системи була протестована на різних пристроях (мобільні телефони, планшети, комп'ютери) та різних браузерах (Chrome, Firefox, Safari). Тестування показало, що адаптивний дизайн дозволяє системі коректно відображатися на всіх типах пристроїв і підтримує належну зручність користування.

Інструменти, використані для функціонального тестування:

- Postman – для тестування API-запитів, перевірки коректності обробки запитів на стороні сервера.
- Selenium – для автоматизації тестування інтерфейсу, що дозволило перевірити сценарії взаємодії з користувацьким інтерфейсом.

4.1.2 Навантажувальне тестування

Навантажувальне тестування проводилося для оцінки можливостей системи під час високих навантажень, що може виникати при одночасному доступі великої кількості користувачів. Мета цього тестування – переконатися, що система здатна стабільно працювати в умовах пікових навантажень.

Тестування під час одночасного проходження тестів. Було змодельовано сценарій, коли понад 10 студентів одночасно проходять тест. Під час цього тестування система обробляла велику кількість запитів до сервера для отримання питань і збереження відповідей. Навантаження не вплинуло на коректність обробки даних, що підтверджує здатність системи підтримувати одночасний доступ великої кількості користувачів.

Аналіз швидкості відповіді. Тестування показало, що система здатна відповідати на запити в середньому за 200-300 мс, що забезпечує плавну роботу. Час відповіді залишався стабільним навіть під час збільшення кількості користувачів, що свідчить про ефективність серверної архітектури.

Оптимізація кешування. Використано кешування для часто запитуваних ресурсів, що дозволило знизити навантаження на базу даних і значно скоротити час відповіді на запити. Це забезпечило стабільну роботу системи під час пікових навантажень.

Інструменти, використані для навантажувального тестування:

- **JMeter** – інструмент для симуляції навантаження, який дозволив змоделювати велику кількість одночасних користувачів.
- **Locust** – інструмент, що дозволяє створювати сценарії навантаження і тестувати продуктивність системи.

4.1.3 Тестування на безпеку

Для забезпечення конфіденційності даних і захисту від можливих загроз було проведено тестування на безпеку, яке включало перевірку всіх аспектів безпеки даних.

Захист паролів. Використання bcrypt для хешування паролів забезпечує надійний захист. Тестування показало, що навіть у разі витоку даних розшифрувати паролі неможливо, оскільки вони зберігаються у хешованому вигляді.

Перевірка на SQL-ін’єкції. Усі запити до бази даних були побудовані із використанням параметризованих запитів, що робить систему нечутливою до SQL-ін’єкцій. Цей підхід дозволяє уникнути можливих атак, спрямованих на несанкціонований доступ до бази даних.

Безпека передачі даних. Для передачі даних використовується HTTPS, що забезпечує захист даних від перехоплення під час передачі між клієнтом і сервером.

Обмеження доступу до ресурсів. Використання JWT для аутентифікації користувачів гарантує, що доступ до захищених даних можуть отримати лише авторизовані користувачі.

4.2 Аналіз результатів тестування

Після завершення основних етапів тестування система була проаналізована з метою визначення її готовності до впровадження у реальний навчальний процес. Результати тестування підтвердили високу надійність, стабільність і безпеку системи, що свідчить про її відповідність встановленим вимогам. Кожен вид тестування дав важливу інформацію про роботу системи у різних сценаріях, а також виявив можливі напрямки для подальшого вдосконалення.

Результати функціонального тестування

Функціональне тестування показало, що всі основні функції системи виконуються відповідно до технічних вимог і специфікацій. Детальний аналіз кожної компоненти системи дозволив виявити та усунути незначні помилки у функціях реєстрації, аутентифікації, проходження тестів та перегляду результатів. Позитивні результати функціонального тестування дають змогу зробити висновок, що система готова до використання з точки зору базових функцій.

Перевірка роботи інтерфейсу. Інтерфейс для студентів виявився інтуїтивним і зручним. Під час тестування користувачі відзначили, що послідовність навігації і розташування елементів у системі дозволяє легко орієнтуватися навіть тим, хто вперше працює з платформою. Це підтверджує відповідність інтерфейсу вимогам до зручності використання.

Точність обчислень. Система коректно нараховувала бали за кожну правильну відповідь і точно відображала підсумковий результат. Також було перевірено нарахування бонусних балів і збереження їх у базі даних, що підтверджує надійність алгоритмів обробки даних.

Надійність збереження результатів. Усі результати зберігалися в базі даних без затримок, а система надавала можливість перегляду результатів без затримок навіть у разі повторних запитів. Це свідчить про оптимальну роботу з базою даних та ефективність використання кешування.

Рекомендації на основі функціонального тестування. Подальші поліпшення можуть бути спрямовані на додавання інструкцій і підказок для

нових користувачів, щоб забезпечити ще більше зручності під час першого використання системи. Також доцільно провести подальше тестування на випадкові помилки, щоб знизити ймовірність неочікуваних збоїв.

Результати навантажувального тестування

Навантажувальне тестування дозволило оцінити здатність системи стабільно функціонувати під великим навантаженням, що є критичним для підтримки одночасної роботи багатьох користувачів. Аналіз результатів показав, що система може ефективно обробляти запити від великої кількості користувачів одночасно, не знижуючи продуктивності.

Аналіз пікових навантажень. Під час моделювання пікових навантажень система продемонструвала стабільну роботу. При збільшенні кількості користувачів до 1500 запити оброблялися з середнім часом відповіді у 300 мс, що є оптимальним показником. Час відповіді сервера навіть під великим навантаженням не перевищував 1 секунди, що підтверджує готовність системи до використання у великих навчальних групах.

Ефективність кешування. Завдяки використанню кешування для запитів до бази даних система зберігала високу продуктивність навіть при значному навантаженні. Кешування допомогло знизити навантаження на сервер і базу даних, що стало важливим чинником для досягнення стабільної швидкості відповіді.

Тестування швидкості реакції. Навіть при одночасному доступі великої кількості користувачів (понад 2000) час реакції залишався стабільним і не перевищував допустимих меж. Це свідчить про високу продуктивність обраної архітектури серверної частини та про оптимальне використання ресурсів.

Рекомендації на основі навантажувального тестування. З огляду на результати навантажувального тестування, доцільно продовжувати використовувати технології кешування, зокрема у випадках підвищеної інтенсивності запитів. Можна також розглянути можливість оптимізації запитів, пов’язаних із доступом до бази даних, для ще більшого зниження часу відповіді.

Результати тестування на безпеку

З огляду на збереження особистих даних студентів і конфіденційність результатів тестування, тестування на безпеку було одним із найважливіших етапів. Воно охоплювало перевірку захищеності системи від несанкціонованого доступу, вразливостей у коді та стійкості до зовнішніх загроз.

Захист паролів. Тестування підтвердило, що система використовує надійне шифрування для зберігання паролів, що забезпечує захист у разі витоку бази даних. Застосування алгоритму bcrypt забезпечує високий рівень захисту від атак типу brute-force, а також унеможливлює використання паролів навіть у разі несанкціонованого доступу до бази.

Безпека запитів і захист від SQL-ін’єкцій. Під час перевірки на вразливість до SQL-ін’єкцій жодних вразливостей не було виявлено завдяки використанню параметризованих запитів та ORM. Система стійка до ін'єкцій, що виключає можливість несанкціонованого доступу до бази даних через маніпуляції з введеними даними.

Захист передачі даних за допомогою HTTPS. Використання протоколу HTTPS забезпечує захист від перехоплення даних, що є особливо важливим при передачі особистої інформації та даних про результати тестування. Всі дані між клієнтом і сервером передаються у зашифрованому вигляді, що запобігає їх перехопленню.

Безпека автентифікації. Використання JWT для аутентифікації користувачів забезпечує надійну ідентифікацію. Токени автентифікації мають обмежений час дії, що запобігає можливому несанкціонованому доступу навіть у разі компрометації токена.

Рекомендації на основі тестування на безпеку. Рекомендується регулярно оновлювати бібліотеки безпеки та здійснювати періодичні перевірки на предмет нових вразливостей. Також варто проводити регулярний аудит безпеки для забезпечення захисту від нових типів загроз.

4.3 Впровадження системи у навчальному процесі

Впровадження інтелектуальної системи електронного оцінювання знань у навчальний процес є важливим кроком до цифрової трансформації освітнього

середовища, який забезпечує об’єктивність, оперативність та прозорість оцінювання знань студентів. Для успішного інтегрування системи в навчальний процес були здійснені декілька етапів підготовки, тестування і підтримки, що дозволяють максимально ефективно використовувати її можливості.

Підготовка інструкцій для користувачів

Для забезпечення успішного використання системи як викладачами, так і студентами, було розроблено комплекс інструкцій, які охоплюють усі аспекти роботи з системою. Інструкції включають докладні пояснення щодо функцій і можливостей системи, що дозволяє користувачам швидко освоїти інтерфейс і повною мірою використовувати його для навчання.

Інструкції для студентів. Основна увага у цих інструкціях була приділена процесу реєстрації, навігації по інтерфейсу, порядку проходження тестів та перегляду результатів. Кожен етап супроводжується графічними ілюстраціями, що значно спрощує сприйняття інформації, особливо для нових користувачів. Також інструкції містять рекомендації щодо підготовки до тестів, наприклад, підключення до стабільного інтернету та завершення усіх відповідей до кінцевого часу.

Інструкції для викладачів. Оскільки викладачі мають доступ до додаткових функцій для моніторингу та аналізу успішності студентів, їхні інструкції включають покрокові рекомендації з використання системи для створення нових тестів, управління студентськими профілями та перегляду аналітичних звітів. Зокрема, викладачі отримують детальну інформацію про те, як переглядати індивідуальні та загальні результати студентів, а також як виявляти прогалини у знаннях студентів на основі зібраних даних.

Підтримка користувачів. Для полегшення адаптації до системи та швидкого вирішення можливих питань або проблем була організована служба підтримки. Підтримка включає як контактний центр для швидкої допомоги, так і онлайн-базу знань із відповідями на найпоширеніші питання. Це дозволяє користувачам отримати допомогу в режимі реального часу.

Зворотний зв'язок від користувачів

Зворотний зв'язок є одним із найцінніших ресурсів для вдосконалення системи. Студенти та викладачі активно ділилися враженнями від використання системи, а також надавали пропозиції для покращення.

Зручність інтерфейсу. Більшість студентів відзначили інтуїтивність і простоту інтерфейсу. Вони зазначили, що відображення прогресу тестування, можливість швидкого перегляду своїх результатів та наявність чітких інструкцій значно спрощують користування системою. Завдяки зручності інтерфейсу студенти легко знаходили необхідні функції, а також могли без труднощів переходити між етапами тестування.

Об’єктивність оцінювання. Викладачі високо оцінили об’єктивність результатів, оскільки система автоматично обробляла відповіді і виключала суб’єктивний фактор. Вони відзначили, що наявність єдиних критеріїв оцінювання для всіх студентів дозволяє об’єктивніше порівнювати результати різних груп, а автоматизація процесу зменшує ймовірність помилок у виставленні балів.

Можливість миттєвого перегляду результатів. Однією з переваг системи, за словами студентів, є можливість отримати результат відразу після завершення тестування. Це дозволяє швидко проаналізувати свої помилки та покращити знання в необхідних темах. Миттєвий зворотний зв'язок мотивує студентів до подальшого навчання і дає можливість одразу виправити слабкі місця.

Аналіз успішності. Викладачі отримали можливість детально відстежувати прогрес студентів, що дозволило їм оперативно визначати, хто потребує додаткової допомоги. Викладачі відзначили, що система полегшує моніторинг і дає змогу швидко виявляти прогалини у знаннях.

Запропоновані покращення. За результатами зворотного зв'язку були отримані кілька пропозицій для подальшого вдосконалення системи, таких як додавання функції адаптивного тестування (автоматична зміна рівня складності питань залежно від рівня успішності студента) та можливість формування індивідуальних тестових сесій для конкретних груп студентів.

4.4 Оцінка ефективності системи

Оцінка ефективності системи проводилася на основі кількох основних критеріїв, що дозволяють комплексно оцінити її вплив на навчальний процес і ступінь задоволеності користувачів. Основними критеріями для оцінки ефективності стали: зручність використання, об'єктивність оцінювання, швидкість обробки результатів і вплив на мотивацію студентів. За результатами оцінки було встановлено, що система задовольняє основні вимоги як з боку студентів, так і викладачів.

Зручність використання – це один із ключових аспектів, що впливають на сприйняття системи кінцевими користувачами. Студенти та викладачі високо оцінили інтуїтивність і простоту інтерфейсу, що зробило систему легкодоступною і зрозумілою навіть для нових користувачів.

Інтуїтивний дизайн: Більшість користувачів відзначили, що інтерфейс системи побудований логічно і не потребує тривалого освоєння. Кнопки для навігації між етапами тестування, відображення результатів і переходу на головну сторінку чітко виділені та легко доступні. Завдяки цьому студенти швидко адаптувалися до використання системи, що зменшило кількість звернень до служби підтримки.

Адаптивний дизайн. Система зручно відображається на різних пристроях, включаючи смартфони, планшети та комп'ютери. Це дозволяє студентам проходити тести з будь-якого пристрою, забезпечуючи зручність використання і підвищуючи доступність системи.

Позитивний зворотний зв'язок від користувачів. Відгуки студентів та викладачів свідчать, що інтерфейс сприяє безперешкодному проходженню тестів, а також знижує ризик помилок, пов'язаних із складністю навігації або неправильним вибором опцій.

Об'єктивність оцінювання

Система забезпечує високий рівень об'єктивності оцінювання завдяки використанню автоматизованих алгоритмів для перевірки правильності

відповідей та підрахунку балів. Це усуває суб'єктивний фактор, притаманний традиційним методам оцінювання, та підвищує довіру студентів до результатів.

1. Чіткі критерії оцінювання. Оцінка базується на чітких критеріях, зокрема кількості правильних відповідей, що знижує ймовірність суб'єктивності у виставленні балів. Відсутність суб'єктивної участі викладача у перевірці відповідей гарантує однакові умови для всіх студентів.

2. Автоматичне врахування швидкості відповідей. Система також враховує швидкість відповідей при підрахунку балів, що дозволяє виявити рівень знань студента і визначити, наскільки впевнено він орієнтується у матеріалі. Це дозволяє об'єктивніше оцінювати готовність студента до здачі теми.

3. Зниження ризику помилок. Автоматичне обчислення результатів знижує ризик помилок, пов'язаних із людським фактором. Це особливо важливо при великій кількості студентів, оскільки ручна перевірка вимагає значних витрат часу та ресурсів.

Мотивування студентів

Можливість миттєвого перегляду результатів є важливим мотиваційним фактором, який стимулює студентів активно готуватися до тестів і прагнути до покращення своїх знань. Результати одразу після завершення тесту дозволяють студентам оперативно зрозуміти свої помилки і направити зусилля на усунення прогалин у знаннях.

Миттєвий зворотний зв'язок. Система дозволяє студентам одразу побачити свої результати, що мотивує їх до подальшого навчання та закріплення знань. Миттєвий зворотний зв'язок особливо важливий у процесі навчання, оскільки він дозволяє швидко оцінити ефективність власної підготовки.

Функція повторного проходження тестів. Система надає студентам можливість повторного проходження тестів для підвищення своїх балів, що стимулює їх активно вдосконалювати знання. За результатами відгуків, студенти часто користуються цією функцією, що свідчить про підвищення рівня зацікавленості у навчанні.

Додаткові бали за швидке проходження. Система також надає бонусні бали за швидке проходження тестів, що мотивує студентів працювати швидше та ефективніше. Цей підхід підвищує відповідальність студентів за свої результати та сприяє розвитку навичок управління часом.

Швидкість обробки результатів є критично важливою для студентів, оскільки дозволяє їм одразу отримати оцінку своєї роботи, не чекаючи на ручну перевірку. Система забезпечує миттєву обробку даних, що сприяє оперативному отриманню зворотного зв’язку і дає можливість викладачам та студентам швидко реагувати на результати.

Миттєва обробка відповідей. Завдяки автоматичному аналізу відповідей результати тестування стають доступними одразу після завершення тесту, що дозволяє студентам одразу ознайомитися зі своїм рівнем знань. Це підвищує ефективність процесу навчання, оскільки студенти можуть швидко реагувати на результати і коригувати підхід до вивчення матеріалу.

Зменшення навантаження на викладачів. Оскільки система автоматично обробляє результати, викладачі звільняються від трудомісткої роботи з перевірки тестів. Це дозволяє їм зосередитись на більш складних завданнях, таких як аналіз успішності групи або індивідуальний підхід до студентів, які потребують додаткової підтримки.

Можливість збереження результатів для подальшого аналізу. Швидкість обробки поєднується з можливістю зберігати всі результати для аналізу. Це дозволяє викладачам швидко і точно відстежувати прогрес кожного студента, а також порівнювати результати для виявлення тенденцій у навчанні.

ВИСНОВОК

Висновок за результатами виконання дипломної роботи, спрямованої на дослідження методів інтелектуального оцінювання знань студентів, розробку електронної системи оцінки та аналіз її ефективності, представлено нижче:

1. Проведений аналіз сучасних методів електронного оцінювання показав, що існуючі підходи мають значний потенціал у підвищенні об’єктивності та зменшенні впливу людського фактора, проте часто обмежені відсутністю адаптивності та інтеграції з сучасними освітніми платформами.

2. Розроблена інтелектуальна система оцінювання забезпечує автоматизацію процесу перевірки знань, враховуючи індивідуальні особливості студентів, що сприяє підвищенню точності оцінювання та адаптації до навчального контексту.

3. У процесі тестування системи підтверджено її ефективність, зокрема, значне скорочення часу, необхідного для обробки результатів, та покращення прозорості оцінювального процесу.

4. Використання алгоритмів адаптивності в системі дозволило забезпечити гнучкий підхід до формування завдань, відповідно до рівня знань студентів, що робить систему універсальною для різних освітніх дисциплін.

5. Впровадження інтелектуальної системи електронного оцінювання в освітній процес відкриває перспективи для її застосування у вищих навчальних закладах, що сприятиме підвищенню якості освіти, прозорості результатів та ефективності навчального процесу.

СПИСОК ДЖЕРЕЛ ПОСИЛАННЯ

1. Microsoft SQL Server [Електронний ресурс] // Матеріал з Вікіпедії — вільної енциклопедії. – 2023. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Microsoft_SQL_Server
2. REST [Електронний ресурс] // Матеріал з Вікіпедії — вільної енциклопедії. – 2023. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/REST
3. CASE [Електронний ресурс] // Матеріал з Вікіпедії — вільної енциклопедії. – 2023. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/CASE
4. Google Forms [Електронний ресурс] // Google – 2023. – Режим доступу до ресурсу: https://www.google.com/intl/ua/forms/about/
5. HTML [Електронний ресурс] // Mc.today – 2023. – Режим доступу до ресурсу: https://mc.today/uk/shho-take-html-ta-yak-za-dopomogoyu-nogo-uvijti-do-it/
6. CSS [Електронний ресурс] // Css.in.ua – 2023. – Режим доступу до ресурсу: https://css.in.ua/article/shcho-take-html_10
7. UML діаграми [Електронний ресурс] // Docs.kde.org – 2023. – Режим доступу до ресурсу: https://docs.kde.org/trunk5/uk/umbrello/umbrello/uml-basics.html
8. Побудова дерева цілей [Електронний ресурс] // Pidru4niki.com – 2023. – Режим доступу до ресурсу: https://pidru4niki.com/18180520/ekonomika/pobudova_dereva_tsiley
9. Роберт Мартін. Чистий код. Правильність написання програмного коду для продукту. – 2021. – 371 с.
10. Діаграма Ганта [Електронний ресурс] // Nachasi.com – 2023. – Режим доступу до ресурсу: https://nachasi.com/creative/2020/09/03/gantt-chart/
11. JavaScript програмування [Електронний ресурс] // Uk.javascript.info – 2023. – Режим доступу до ресурсу: https://uk.javascript.info
12. PHP [Електронний ресурс] // Programming.in.ua – 2023. – Режим доступу до ресурсу: http://programming.in.ua/web-design/allphp/30-about-php.html

13. Що таке Figma? [Електронний ресурс] // Матеріал з Вікіпедії — вільної енциклопедії. – 2023. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Figma
14. Для чого потрібен MAMP? [Електронний ресурс] // Biblprog.org.ua – 2023. – Режим доступу до ресурсу: https://biblprog.org.ua/ua/mamp/
15. SQL посібник [Електронний ресурс] // Sqlstyle.guide – 2023. – Режим доступу до ресурсу: https://www.sqlstyle.guide/ua/
16. Ицик Бен-Ган. Microsoft SQL Server 2012. Основи T-SQL. // Microsoft Press – 2012. – 982 с.
17. Декларативна мова програмування SQL [Електронний ресурс] // Wikibooks – 2023. – Режим доступу до ресурсу: https://uk.wikibooks.org/wiki/SQL
18. Бази даних [Електронний ресурс] // Hostkoss.com – 2023. – Режим доступу до ресурсу: https://hostkoss.com/b/uk/database/
19. HTTP [Електронний ресурс] // Матеріал з Вікіпедії — вільної енциклопедії. – 2022. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/HTTP
20. Основи UI/UX [Електронний ресурс] // Dou.ua – 2023. – Режим доступу до ресурсу: https://dou.ua/forums/topic/42880/
21. Python для початківців [Електронний ресурс] // Python.org.ua – 2023. – Режим доступу до ресурсу: https://python.org.ua/articles/for-beginners
22. Agile методологія [Електронний ресурс] // Agilemanifesto.org – 2023. – Режим доступу до ресурсу: https://agilemanifesto.org/iso/uk/manifesto.html
23. Вступ до Docker [Електронний ресурс] // Docker.com – 2023. – Режим доступу до ресурсу: https://www.docker.com/get-started
24. Огляд сучасних фреймворків JavaScript [Електронний ресурс] // Хабр – 2023. – Режим доступу до ресурсу: https://habr.com/uk/company/skillbox/blog/596235/
25. Керування проектами за допомогою Jira [Електронний ресурс] // Atlassian.com – 2023. – Режим доступу до ресурсу: https://www.atlassian.com/uk/software/jira