

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТРАНСПОРТНИЙ УНІВЕРСИТЕТ
Навчально-науковий інститут управління, технологій та правових наук
Кафедра інформаційних технологій

МАГІСТЕРСЬКА РОБОТА

на тему:

ВЕБ-ОРІЄНТОВАНА СИСТЕМА ТЕСТУВАННЯ ЗНАНЬ СТУДЕНТІВ

Спеціальність: 121 «Інженерія програмного забезпечення»
ОПП «Інженерія програмного забезпечення»

Розробив: Анатолій ГОСТИЩЕВ

(ім'я, ПРІЗВИЩЕ)

_____ (підпис)

Керівник: к.ф.-м.н., доцент Олександр ТКАЧЕНКО

(науковий ступінь, вчене звання) (ім'я, ПРІЗВИЩЕ)

_____ (підпис)

Нормоконтроль: к.ф.-м.н., доцент Олександр ТКАЧЕНКО

(науковий ступінь, вчене звання) (ім'я, ПРІЗВИЩЕ)

_____ (підпис)

Допущений до захисту

« ____ » _____ 2025р.

В.о. завідувача кафедри: д.т.н., професор Олег НІКОНОВ

(науковий ступінь, вчене звання) (ім'я, ПРІЗВИЩЕ)

_____ (підпис)

Рецензент: к.т.н., доцент Ірина МАШКІНА

(науковий ступінь, вчене звання) (ім'я, ПРІЗВИЩЕ)

_____ (підпис)

Київ – 2025

Міністерство освіти і науки України
 Національний транспортний університет
 Навчально-науковий інститут управління, технологій та правових наук
 Кафедра інформаційних технологій

Рівень вищої освіти – другий (магістерський)
 Галузь знань 12 «Інформаційні технології»
 Спеціальність 121 «Інженерія програмного забезпечення»
 ОПП «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри
 інформаційних технологій
 д.т.н., проф. _____ Олег НІКОНОВ

« ____ » _____ 2025 р.

Завдання
на магістерську роботу здобувачу вищої освіти
Анатолію ГОСТИЩЕВУ

(ім'я, прізвище)

1. Тема роботи: Веб-орієнтована система тестування знань студентів

керівник роботи: к.ф.-м.н., доцент Олександр ТКАЧЕНКО

Затверджені наказом від «17» жовтня 2025 року № 1031

2. Строк подання здобувачем роботи 05 грудня 2025 року

3. Вихідні дані до роботи: веб-орієнтована система тестування знань студентів, база даних тестів веб-орієнтованої системи тестування знань студентів

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) огляд та аналіз області дослідження (аналіз існуючих підходів та методів контролю знань студентів, традиційне та комп'ютерне тестування, огляд та аналіз сучасних систем тестування), аналіз та шляхи розв'язку проблеми (використання нечіткої логіки для тестування знань студентів, онтологічне моделювання процесів тестування знань студентів), опис програмного продукту (Система тестування знань студентів: інструкція програміста, Інструкція студента, інструкція викладача

5. Перелік графічного матеріалу: презентація магістерської роботи, скріншоти, які демонструють функціонування системи тестування знань студентів

6. Консультанти роботи: к.ф.-м.н., доцент Олександр ТКАЧЕНКО

7. Календарний план виконання роботи

№ з/п	Найменування етапів магістерської роботи	Строк виконання етапів магістерської роботи	Примітки
1	Огляд та аналіз джерел тематичної області	17.10.25 – 22.10.25	
2	Огляд та аналіз шляхів розв'язання проблем роботи	23.10.25 – 28.10.25	
3	Опис власного підходу до вирішення проблеми	29.10.25 – 04.11.25	
4	Визначення вимог до програмного продукту та його структури	05.11.25 – 07.11.25	
5	Опис та реалізація основних функцій програмного продукту	08.11.25 – 28.11.25	
6	Оформлення пояснювальної записки	29.11.25 – 05.12.25	
7	Захист		

8. Дата видачі завдання «17» жовтня 2025 року

Здобувач вищої освіти

(підпис)

Гостіщев А.С.

(прізвище та ініціали)

Керівник роботи

(підпис)

Ткаченко О.А.

(прізвище та ініціали)

РЕФЕРАТ

Тема: «Веб-орієнтована система тестування знань студентів»

Магістерська робота: 96 с., 35 рис., 1 таблиця, 51 літературне джерело, 1 додаток.

Магістерська робота присвячена розробці веб-орієнтованої системи тестування знань студентів в умовах дистанційного навчання. Метою роботи є дослідження предметної області, моделювання процесів тестування, що дозволить оптимізувати процеси тестування й підвищити ефективність процесів навчання, та розробити систему тестування знань студентів.

Дослідження зосереджено на розробці веб-орієнтованої системи, що включає в себе функціонал для тестування знань студентів, з наданням різних рівнів складності тестових завдань. Застосовані сучасні веб-технології, такі як HTML, CSS, C# та бази даних для реалізації потрібного функціоналу системи.

У процесі розробки було проведено аналіз потреб щодо визначення рівня знань студентів та вимог до системи, що дозволило сформулювати ключові функціональні вимоги та визначити необхідність інтеграції з зовнішніми сервісами для отримання актуальних даних про рівні засвоєння студентами навчального контенту.

Результатом роботи є розроблена веб-орієнтована система тестування знань студентів, яка дозволяє студентам комфортно перевіряти свої знання з обраної теми, самостійно обираючи варіант відповіді на тестове питання відповідного рівня складності, а також забезпечує користувачам системи зручний інтерфейс для навігації системою.

Отримана система має потенціал для подальшого розширення та вдосконалення, включаючи можливості використання штучного інтелекту для надання студентам обґрунтованого аналізу помилок, що були виявлені під час тестування.

Ключові слова: веб-орієнтована система, навчальний контент, тестування, тестове питання, база даних тестів, рівень складності тесту.

ABSTRACT

Topic: "Web-based system for testing students' knowledge"

Master's thesis: 96 pages, 35 figures, 1 table, 51 literary sources, 1 appendix.

The master's thesis is devoted to the development of a web-oriented system for testing students' knowledge in distance learning. The aim of the work is to study the subject area, model testing processes, which will allow optimizing testing processes and increasing the efficiency of learning processes, and develop a system for testing students' knowledge.

The research focuses on the development of a web-oriented system that includes functionality for testing students' knowledge, providing different levels of complexity of test tasks. Modern web technologies, such as HTML, CSS, C# and databases, were used to implement the required functionality of the system.

During the development process, a needs analysis was conducted to determine the level of students' knowledge and requirements for the system, which allowed formulating key functional requirements and determining the need for integration with external services to obtain up-to-date data on the levels of students' assimilation of educational content.

The result of the work is a developed web-based system for testing students' knowledge, which allows students to comfortably test their knowledge on a selected topic, independently choosing an answer option for a test question of the appropriate level of complexity, and also provides system users with a convenient interface for navigating the system.

The resulting system has the potential for further expansion and improvement, including the possibility of using artificial intelligence to provide students with a substantiated analysis of errors that were detected during testing.

Keywords: web-based system, educational content, testing, test questions, test database, test difficulty level.

ЗМІСТ

ВСТУП	7
1 ОГЛЯД ТА АНАЛІЗ ОБЛАСТІ ДОСЛІДЖЕННЯ	10
1.1 Аналіз існуючих підходів та методів контролю знань студентів	10
1.1.1 Традиційне тестування	10
1.1.2 Комп'ютерне тестування	13
1.2 Огляд та аналіз сучасних систем тестування	16
1.2.1 Тестові технології навчання і контролю знань у системі Moodle	17
1.2.2 Google Forms	21
1.2.3 Kahoot!	23
1.2.4 Online-платформа ATutor	25
1.2.5 Інтерактивна платформа для створення тестів JustClass	26
1.2.6 Online-генератор тестів EasyTestMaker	28
2 АНАЛІЗ ТА ШЛЯХИ РОЗВ'ЯЗКУ ПРОБЛЕМИ	33
2.1 Використання нечіткої логіки для тестування знань студентів	33
2.2 Онтологічне моделювання процесів тестування знань студентів	36
2.2.1 Функції системи тестування, що використовує онтологічну модель	36
2.2.2 Онтологічна модель системи тестування знань студентів	37
3 ОПИС ПРОГРАМНОГО ПРОДУКТУ	43
3.1 Система тестування знань студентів: Інструкція програміста	43
3.2 Інструкція користувача системи тестування	56
3.2.1 Інструкція студента	56
3.2.2 Інструкція викладача	61
ВИСНОВКИ	70
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	72
ДОДАТКИ	76

ВСТУП

В сучасному освітньому середовищі важливо забезпечити ефективний контроль та оцінювання знань студентів.

Розвиток інформаційних навчальних систем (інтелектуальних навчальних систем, інформаційних навчальних систем з елементами інтелектуалізації), спрямованих на автоматизацію, об'єктивізацію та підвищення ефективності процесів у сфері освіти, а також на підвищення мотивації студентів до навчання, є актуальними проблемами.

Тому цілком обгрунтовано, що одним з основних компонентів таких систем є підсистема контролю знань студентів, яка дозволяє автоматизувати процеси тестування та оцінювання.

Розробка системи контролю знань (тестування) спрямована на автоматизацію процесу тестування, підвищення об'єктивності оцінювання та забезпечення зручного і доступного інструменту для викладачів і студентів.

Перевагами, які надають системи тестування, зокрема є:

- підвищення рівня індивідуалізації, моніторингу та контролю знань студентів;
- виключення можливості використання сторонньої допомоги, використовуючи обмеження часу на виконання завдань тестів;
- автоматичне ведення звітності про результати тестування студентів з можливістю формування відповідних звітів;
- систему тестування знань студентів можна інтегрувати з наявними в закладі вищої освіти системами управління навчальним процесом та інформаційними навчальними системами.

Актуальність проблеми, що розглядається у даній магістерській роботі, полягає у тому, що велика кількість задач забезпечення тестування знань студентів, може бути вирішена за допомогою моделювання процесу тестування для різних навчальних дисциплін, різних видів тестів та

оптимізації процесу тестування для різних категорій користувачів (викладачів, студентів).

Науковою новизною роботи є використання онтологічного моделювання процесів тестування знань студентів при вирішенні проблем забезпечення оптимізації та індивідуалізації процесів навчання.

Таке моделювання процесів тестування знань студентів, зокрема:

- збільшує ефективність роботи всіх учасників навчального процесу (зокрема, процесу тестування знань студентів);
- зменшує час, необхідний викладачеві для перевірки виконання студентами тестів;
- підвищує рівень об'єктивізації оцінювання рівня знань студентів.

Метою роботи є дослідження предметної області (методів, видів та процесів тестування знань студентів), моделювання процесів тестування, що дозволить оптимізувати процеси тестування й підвищити ефективність процесів навчання, та розробити систему тестування знань студентів спеціальності 121 «Інженерія програмного забезпечення».

Для досягнення поставленої мети необхідно вирішити ряд завдань:

- зробити огляд та аналіз предметної області дослідження;
- визначити цілі, види, методи та підходи тестування знань студентів;
- проаналізувати існуючі системи і платформи тестування;
- обґрунтувати вибір онтологічної моделі для обраної предметної області;
- здійснити збір загальної інформації та експертних знань, необхідних для розробки відповідної моделі процесів тестування;
- розробити онтологічну модель процесів тестування знань студентів;
- визначити вимоги до системи тестування, її функції та основні компоненти;
- розробити систему тестування знань студентів спеціальності 121 «Інженерія програмного забезпечення».

Об'єктом дослідження в магістерській роботі є дані та знання, що

відповідають сфері тестування студентів.

Предметом дослідження в магістерській роботі є сукупність підходів до моделювання процесів тестування та розробки програмного продукту – системи тестування знань студентів

Практична цінність магістерської роботи полягає у впровадженні інформаційної системи тестування, побудованої на основі відповідної онтологічної моделі процесів тестування, що забезпечить обробку великих обсягів інформації, їх збереження та оперативну обробку.

Магістерська робота складається зі вступу, трьох основних розділів, висновків, списку використаних джерел та додатку, в яких розглядаються методи, підходи та принципи, що використовуються при тестуванні знань студентів та онтологічному моделюванні процесів тестування для оптимізації, контролю, моніторингу тестування знань студентів та візуалізації його результатів.

Особистий внесок магістра. Магістерська робота є самостійним завершеним дослідженням автора.

Публікації. дослідження були апробовані:

– Ткаченко К.О., Носов І.С., Гостіщев А.С. Розроблення інформаційних систем управління та організації навчального процесу: онтологічний підхід. *Актуальні питання розвитку галузей: збірник наукових праць з матеріалами VI Міжнародної наукової конференції* (м. Вінниця, 31 жовтня, 2025). Вінниця: ТОВ «УКРЛОГОС Груп, 2025. С.428 – 432. [50].

– результати дослідження опубліковані в науковій статті Ткаченко К.О., Ткаченко О.А., Носов І.С., Гостіщев А.С. Розробка інформаційних систем тестування знань студентів: онтологічний підхід. *Наука і техніка сьогодні*, 2025. № 10(51). С. 2039-2052. [https://doi.org/10.52058/2786-6025-2025-10\(51\)-2039-2052](https://doi.org/10.52058/2786-6025-2025-10(51)-2039-2052) [51].

Тези доповіді та стаття додаються до даної роботи.

1 ОГЛЯД ТА АНАЛІЗ ОБЛАСТІ ДОСЛІДЖЕННЯ

1.1 Аналіз існуючих підходів та методів контролю знань студентів

1.1.1 Традиційне тестування

Для ефективного контролю знань студентів існують різні підходи та методи, серед яких найбільш поширеними є традиційні тестування, письмові контрольні роботи, практичні завдання, усні опитування та інші. Однак, з розвитком інформаційних технологій з'явилися нові можливості для автоматизації цих процесів та покращення їх ефективності.

Традиційні методи контролю знань, такі як письмові тести, контрольні роботи та інші, використовуються в основному для перевірки теоретичних знань студентів.

Вони забезпечують досить об'єктивну оцінку, але можуть бути дуже довгими по часу та потребувати великих обсягів різних ресурсів кількості студентів, що проходять тестування, є дуже великою.

Одним з найбільш поширених традиційних методів контролю знань є письмове тестування. Цей метод дозволяє викладачам швидко оцінити рівень засвоєння матеріалу студентами за допомогою питань із закритими відповідями.

Письмові тести можуть бути використані для перевірки різних аспектів знань, від фактичної інформації до розвитку аналітичних та критичного мислення.

Ще одним традиційним методом є контрольні роботи. Ці завдання можуть бути написані або усно задані студентам для перевірки їх знань і розуміння певної теми або матеріалу. Контрольні роботи дозволяють викладачам оцінити не лише рівень знань, але й здатність студента застосовувати їх у практичних ситуаціях.

Однак, ці традиційні методи мають свої обмеження. Вони можуть бути обтяжливими для вчителя у великих групах студентів, а також не завжди забезпечують повну картину про рівень засвоєння матеріалу студентами.

Також, вони можуть бути менш ефективними в оцінці навичок, таких як комунікативність, креативність та інші.

Тест – це сукупність завдань, які треба виконати при проходженні тестування.

Тести, призначені для самоперевірки, містять незмінний набір завдань, що сформував викладач (автор, методист, тощо) при створенні навчального курсу (в тому числі й online-курсу) чи окремих його розділів (тем тощо).

Такі тести мають не тільки визначають рівень знань студентів, а й сприяють більш глибокому засвоєнню навчального матеріалу (навчального контенту) [1, 2].

Тестування може проводитися для різних етапів засвоєння матеріалу. Це може бути проміжний чи рубіжний (фінальний, кінцевий, семестровий) контроль успішності навчання. Це може бути також тестування для здійснення студентами самоперевірки рівня засвоєння ними того чи іншого навчального контенту розділу (теми, блоку тем, курсу чи якого окремого поняття).

За результатами тестувань студент отримує оцінку в балах, яка впливає на порядок подальшого вивчення навчального контенту курсу.

Студентові може надаватися більш складно чи навпаки більш детально (з великою кількістю прикладів тощо) викладений навчальний контент.

Результати тестування впливають на визначення індивідуальної успішності студента, тому при їх проведенні слід створювати умови для уникнення суб'єктивізму та неадекватності оцінки.

Тести мають бути придатними для визначення саме тих показників рівня засвоєння знань, які перевіряються тестуванням [3].

Результати тестів повинні об'єктивно та повно відображати рівень засвоєння студентами відповідного навчального матеріалу (контенту).

Для створення таких тестів їх автор повинен створити відповідну базу завдань, виділяючи:

- основні положення;
- поняття;
- твердження й дані, якими студент має володіти після вивчення курсу.

Тести повинні бути надійними, тобто відтворювати результати тестування при багаторазовому його виконанні одним і тим самим студентом (групою студентів).

Надійність тесту впливає на точність визначення оцінки та на довіру до неї. На надійність і точність тесту впливає кількість завдань, з яких він складається.

Збільшення розміру тесту сприяє підвищенню точності оцінки, бо розширює коло питань щодо навчального контенту курсу.

Тому автор (укладач) тестів прагне надати в тесті як можна більше завдань, але враховуючи при цьому обмеження, що накладаються на час тестування.

Оцінювання рівня знань за результатами тестування найчастіше полягає в простому способі нарахування балів, коли повністю правильна відповідь оцінюється в 1 бал, а всі інші – оцінюються нулем [4, 5].

Тобто лише за умови вибору студентом всіх правильних відповідей і відкидання всіх неправильних, він отримує 1 бал.

Якщо ж хоча б один з правильних варіантів не враховано або вибрано хоча б один неправильний, то така відповідь оцінюється в 0 балів.

Для більш точного і гнучкого оцінювання можна враховувати ще й частково правильні відповіді. Завдання складається з кількох тверджень, частина з яких неправильна.

Кількість правильних відповідей у завданні не вказується.

Таким чином, існує ймовірність обрати не всі правильні відповіді або разом з правильними вибрати й неправильні.

За кожну правильну відповідь 1 бал нараховується, а за кожну неправильну – віднімається.

Таким чином набрані в тесті бали є різницею між кількостями правильних і неправильних відповідей. Якщо ця різниця від'ємна, то все завдання оцінюється нулем балів. Максимальна оцінка за виконання завдання – це кількість правильних відповідей.

1.1.2 Комп'ютерне тестування

Основними завданнями в комп'ютерному тестуванні є створення (формування, складання) адекватних, непротивічних і надійних тестів та використання надійних способів обробки та інтерпретації результатів тестування.

Для впровадження комп'ютерної перевірки знань слід мати відповідні системи тестування, які повинні мати програмне забезпечення для:

- створення тестових завдань;
- встановлення правильності виконання тестових завдань (і відповідно надання оцінки рівня знань студента).

Перевірка виконання завдань тесту відбувається так, що система порівнює відповіді студента з тими, що закладені в базу системи як правильні.

Система після порівняння відповідей зарахування правильну/неправильну відповідь залежно від того, було/не було зафіксовано співпадіння відповідей.

Системи тестування можуть бути окремим програмним продуктом чи програмним модулем, що входить до управління навчанням [6, 7].

Розвиток систем тестування здійснювався в напрямку об'єднання в одній системі:

- засобів створення завдань тестів,
- проведення тестування,

- обробки результатів тестування,
- захист та зберігання результатів тестування.

В наш час системи тестування значно ускладнились, а прості розробки перестали задовольняти сучасним вимогам.

В наш час вже існує багато достатньо досконалих безплатних систем тестування.

Текстові завдання можуть бути введені з клавіатури через інтерфейс системи тестування чи імпортовані до системи тестування у вигляді відповідним чином підготовлених файлів.

В сучасних системах тестування в тій чи іншій мірі передбачена можливість:

- використання шаблонів типових завдань, їх підготовка та редагування;
- імпорту та вставлення зображень, формул, схем, таблиць, HTML-коду, аудіо- та відеомультимедійних об'єктів (як для тестів, так і для відповідей).

Сучасні системи тестування направлені на те, щоб, зокрема:

- *Зменшити ймовірність фальшування при тестуванні.* Для цього тест має генеруватися індивідуально для кожного студента.

Найпростіше це зробити, змінюючи порядок надання завдань (питань) тесту з одного й того самого набору.

У розвинутих системах тестування використовується випадковий вибір потрібної кількості завдань (питань) з бази завдань щодо обраної предметної області тестування. Кількість завдань в тесті задається викладачем.

- *Проводити адаптивне тестування,* коли використовуються завдання, складність яких попередньо встановлена емпірично, а сам адаптивний тест формується безпосередньо під час його виконання, коли складність кожного наступного завдання залежить від результату виконання попереднього.

- *Скоротити час тестування,* що обумовлює, зокрема, наступне:

- час порогу втомлення не перевищується;
- тестування стає дешевшим, бо зменшуються витрати за використання апаратного забезпечення та спеціальних приміщень для тестування великих груп.

– *Надавати коментарі до наданих студентом відповідей.* Найчастіше студентові повідомляється, що відповідь зарахована як правильна/неправильна.

У більш розширених коментарях студентові надають правильну відповідь, а при неправильній відповіді система:

- надає пояснення щодо причини помилки;
- надає додатковий навчальний матеріал, щоб студент все ж таки засвоїв наданий йому навчальний контент.

При наданні коментарів можуть бути задіяні анімовані персонажі, які демонструють свою реакцію на відповідь, впливаючи на емоційну сферу студента.

– *Задавати умови тестування* (наприклад, обмеження на час всього тестування чи виконання окремого завдання (питання) тесту.

Програмне забезпечення сучасних систем тестування для забезпечення в повній мірі функціональних вимог повинно мати такі компоненти, як:

– *Внесення коефіцієнтів складності кожного завдання тесту.* Завдяки цьому обчислюються результати тестування та надається оцінка рівня знань студента після виконання тесту.

– *Створення звітів за результатами тестування студентів.* Форма звіту обирається викладачем з наявного в системі набору шаблонів.

Система тестування має забезпечувати зберігання та захист звітів від несанкціонованого доступу.

– *Підтримки локального та мережевого режимів тестування.* Програмне забезпечення системи тестування повинно бути стандартизованим і підтримувати широкий спектр процедур обробки даних, сумісних з наявними системами управління навчальним процесом.

– *Комфортного та зручного інтерфейсу.* Панель інструментів системи тестування повинна мати зручну та зрозумілу структуровану систему кнопок керування процесом тестування.

Інструментарій викладача для налаштування параметрів тестування та адміністрування вирізняється від кнопок, які використовуються студентами (наприклад, для входу до системи, надання відповідей, переходу до наступного тестового завдання).

1.2 Огляд та аналіз сучасних систем тестування

Для використання комп'ютерного тестування в університетах, які не використовують системи управління навчальним процесом ((наприклад, Moodle [6] та Ilias [7]), зазвичай застосовують спеціалізовані програмні продукти, наприклад такі, як:

- SunRav [8];
- TestOfficePro [8];
- WebQuiz XP [9];
- Course Lab [10];
- Adobe Captivate [11];
- My Test [12];
- AD Tester [13].

Їх структура зазвичай однотипна. Тестові програми мають такі програмні блоки:

- створення тестів (конструктор тестів) – виконується викладачами з використанням відповідного програмного забезпечення;
- представлення завдань для виконання їх студентами;
- обробки результатів тестування студентів.

Системи тестування студентів використовуються для створення тестів, проведення безпосередньо тестування, автоматизації процесу перевірки виконаних завдань тесту, підтримки гнучкого навчання.

Популярними системами є Google Forms [14, 15], Kahoot! [16 – 18], Quizizz [19], Learning Apps [20].

1.2.1 Тестові технології навчання і контролю знань у системі Moodle

Одним з найважливіших модулів системи управління процесами навчання Moodle (завантаження системи показано на рис. 1.1) [6, 21 – 24] є тест, проведення якого мінімізує витрати часу викладача на перевірку знань студентів і дозволяє обробити та зберегти отримані результати.

Використання технологій тестування сприяє підвищенню якості та об'єктивності оцінювання знань студентів, знижуючи суб'єктивність оцінки, що надається студенту.

При тестуванні у Moodle студенти можуть обирати навчальний режим тестування або підсумковий контроль знань.

Навчальний режим тестування передбачає визначення рівня засвоєння окремих тем, розділів (тобто невеликих елементів навчального контенту).

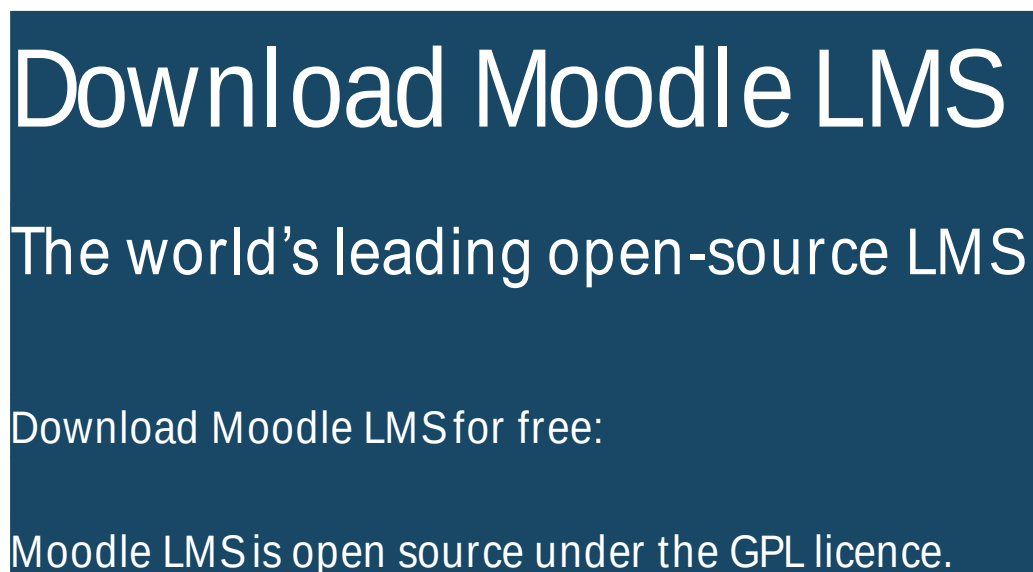


Рисунок 1.1 – Короткий опис системи Moodle при її завантаженні

Такий режим надає студенту можливість аналізу результатів навчання та виправлення помилок, зроблених під час тестування.

Після опанування студентом навчального контенту та визначення рівня отриманих знань система Moodle надає студенту результат його роботи, який стає стимулом для подальшої навчання (в тому числі й з системою Moodle).

Moodle надає багато можливостей для створення тестів різних типів та рівнів [6, 21] з:

- визначеною кількістю спроб проходження тесту;
- встановленням часових затримок між спробами;
- вибором методу оцінювання при кількох спробах (вища/середня оцінка, перша/остання спроба);
- перемішуванням завдань (питань) тесту та варіантів відповідей;
- побудовою тесту на основі випадкового вибору завдань (питань);
- навчальним режимом (можливістю багаторазової відповіді на питання у межах однієї спроби, штрафними балами за кожну неправильну відповідь);
- налаштуванням режиму перегляду результатів;
- наданням коментарів (пояснень) до кожного питання, варіанту відповіді чи всього тесту.

Перед проведенням тестування треба визначити параметри відповідного тесту (табл. 1.1) [6].

Таблиця 1.1 – Параметри тестів при різних режимах тестування

Параметри тестового завдання	Рекомендації щодо	
	навчального режиму	підсумкового контролю
1	2	3
Початок тестування	Проводиться протягом 2–3 занять	Проводиться протягом одного заняття
Закінчення тестування		

Продовження таблиці 1.1

1	2	3
Обмеження часу (хвилин)	Не обмежено	30 хв на 30 питань
Кількість спроб	Не обмежено	1
Метод оцінювання	Остання спроба	Перша спроба
Випадковий порядок питань	Ні	Так
Всі питання на одній сторінці	Так	Кожне питання з нової сторінки
Метод навігації по питанням	Вільний	Послідовний
Необхідний пароль	—	За рішенням викладача
Випадковий порядок відповідей	Ні	Так
Налаштування режиму перегляду	За рішенням викладача (можна активізувати всі опції)	Відображати кількість балів
Необхідна мережна адреса	—	За рішенням викладача

При розробці навчального online-курсу його доцільно наповнити тестами (наприклад, такими як тест самоконтролю, тренінг) [22 – 24].

Основною характеристикою таких тестів має бути можливість аналізу і/або виправлення своїх помилок студентом.

Для цього студентк потрібно надати можливість:

- кілька разів проходити тест;
- перегляду результатів тестування;
- отримання пояснення/додавання коментаря.

Тест самоконтролю призначений для самостійної перевірки студентом рівня засвоєння відповідного навчального контенту.

Для проходження тесту студенту надається декілька спроб. На рис. 1.2 показано результат самоконтролю, проведеного в Moodle [21].

При налаштуванні тесту самоконтролю слід вказати:

- час, що відводиться на тестування;
- режим перемішування (питань та відповідей);
- кількість спроб, що надається;
- середню оцінку.

Дозволених спроб: 3

Обмеження за часом: 10 хв

Метод оцінювання: Середня оцінка

Результати ваших попередніх спроб

Спроба	Стан	Оцінка / 10,00	Перегляд
1	Завершено Надіслано неділя 26 квітень 2015 11:20	8,00	Перегляд

Середня оцінка: 8,00 / 10,00.

[Пройти тест заново](#)

Рисунок 1.2 – Приклад результату самоконтролю вивченого навчального контенту за окремою темою

На рис. 1.3 продемонстровано створення тестів для перевірки знань в online-курсі за темою «Операційні системи» [21].

Показано, з яких ресурсів можуть бути вибрані питання (завдання) тестів.

Тренінг призначений для концентрації уваги студента на найважливіших моментах теми.

Він виконується в адаптивному режимі, коли після відповіді на питання, студенту, якщо він відповів неправильно, надаються ще спроби, а за неправильні відповіді вираховуються штрафні бали.

Використання цих технологій тестування здійснює оптимізацію роботи викладача та студентів.

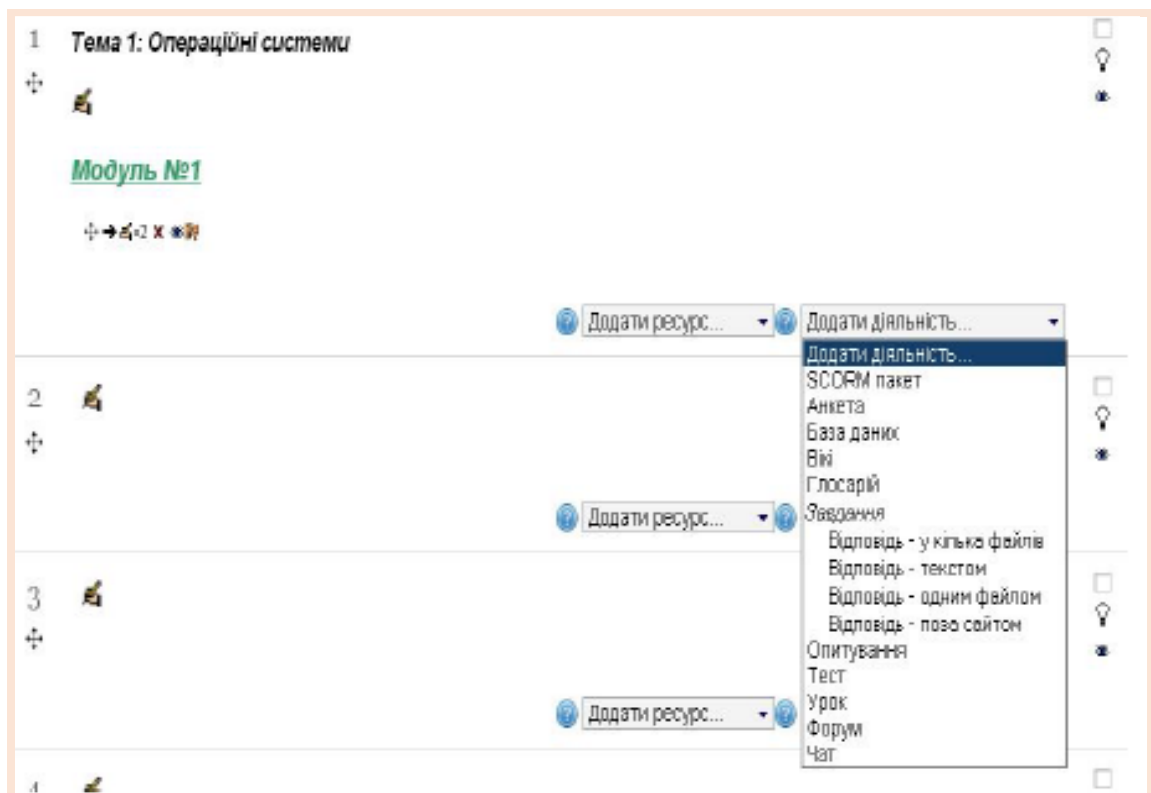


Рисунок 1.3 – Приклад результату самоконтролю вивченого навчального контенту за окремою темою

1.2.2 Google Forms

Google Forms – це інструмент, який надає безкоштовну можливість створення опитувань та тестів в Інтернеті (рис. 1.4).

Він є частиною сервісу Google Документи і має простий та зрозумілий інтерфейс, що дозволяє швидко створювати анкети (перелік витань) навіть

людям без спеціальних навичок у роботі з персональними комп'ютерами та іншими мобільними гаджетами [14, 15].

Однією з ключових переваг Google Forms є можливість створювати різні типи питань, такі як одноразовий вибір, множинний вибір, текстові відповіді, шкали оцінювання тощо.

Це дозволяє створювати різноманітні тести та анкети для збору різноманітної інформації.

Також надається можливість автоматичного аналізу відповідей.

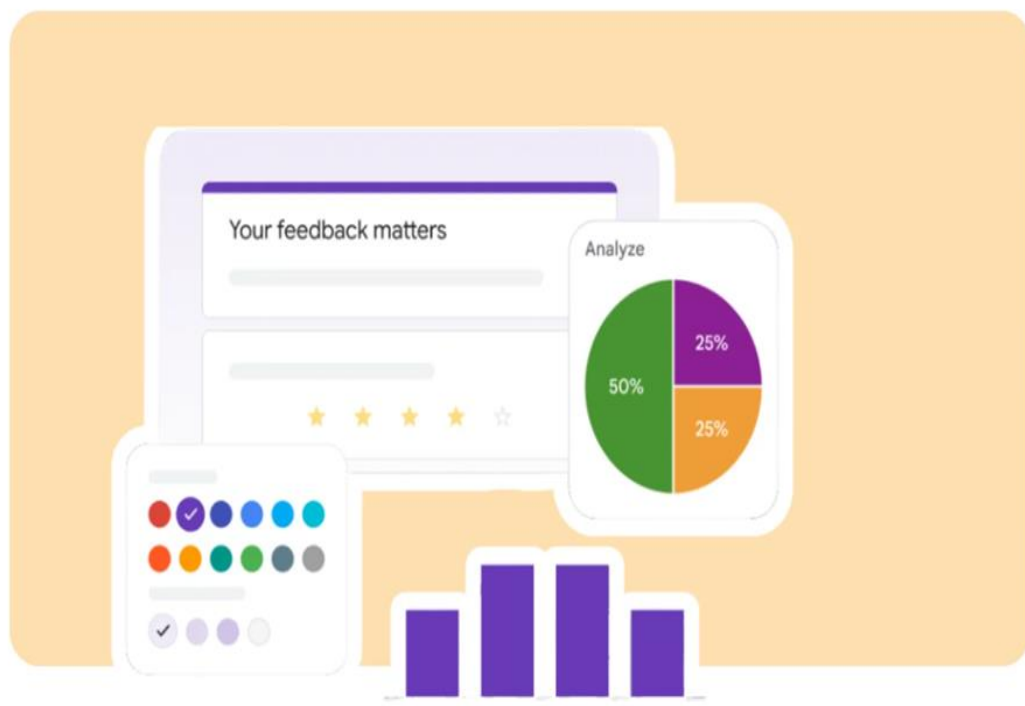


Рисунок 1.4 – Емблема Google Forms

Після того, як користувачі заповняють анкету або тест, система автоматично аналізує їхні відповіді і генерує звіти.

Це дозволяє швидко і ефективно оцінити результати тестування або зібрану інформацію в анкеті.

Крім того, Google Forms має можливість інтеграції з іншими сервісами Google, такими як Google Sheets і Google Drive [14, 15]. Це дозволяє автоматично зберігати дані з анкет або тестів у вказаний файл таблиці або зберігати їх у хмарному сховищі Google Drive.

Загалом, Google Forms – це потужний інструмент для створення опитувань та тестів, який є гарним рішенням для:

- викладачів, студентів,
- організацій, які потребують збору даних або здійснення тестування в Інтернеті.

Його простий інтерфейс та можливості автоматизації роблять його популярним серед користувачів.

1.2.3 Kahoot!

Kahoot! – це популярна інтерактивна платформа, яка використовується для проведення вікторин та тестувань в реальному часі (рис. 1.5).

Вона відома своїм захоплюючим та цікавим підходом до навчання, який залучає студентів та сприяє активному засвоєнню матеріалу [16 – 18].



Рисунок 1.5 – Вигляд емблеми системи тестування знань Kahoot!

Однією з ключових особливостей Kahoot! є можливість створювати власні інтерактивні ігри для навчання.

Викладачі та тренери можуть створювати вікторини на різні теми, додавати питання з різними типами відповідей, зокрема [16]:

- одноразовий вибір завдання;
- множинний вибір завдання;
- правда/брехня тощо

та налаштовувати час на відповідь.

Це дозволяє створювати цікаві та розвиваючі завдання для студентів.

Крім того, Kahoot! надає можливість проведення навчальної гри в реальному часі з використанням мобільних пристроїв та гаджетів.

Учасники можуть підключитися до гри за допомогою своїх смартфонів або планшетів і відповідати на питання прямо під час гри.

Це створює атмосферу змагання та стимулює студентів до активної участі.

Ще однією важливою функцією є можливість аналізу результатів гри за допомогою якою студенти і навчаються, і потім перевіряють рівень засвоєних під час гри знань [17].

Після завершення ігри викладач або тренер може переглянути результати кожного учасника гри (студента), а також отримати зведену статистику про загальні результати гри.

Це дозволяє викладачам та тренерам зрозуміти, наскільки добре студенти засвоїли матеріал, і вносити необхідні корективи до навчального процесу (або ж до навчального контенту).

У цілому, Kahoot! є потужним інструментом для створення захоплюючих та ефективних уроків та тренувань [18].

Вона дозволяє викладачам та тренерам зробити навчання більш цікавим і ефективним, а студентам (які стають учасниками гри) – більш захоплюючим та цікавим.

Аналіз існуючих підходів та програмних засобів для контролю знань студентів показує, що використання інформаційних технологій може значно полегшити та покращити процес контролю знань.

Розробка власної підсистеми контролю знань має великий потенціал для підвищення якості освіти та зручності для представників різних категорій користувачів.

1.2.4 Online-платформа ATutor

Online-платформа ATutor розроблена фахівцями Тернопільського національного технічного університету імені І. Пулюя [25].

Ця online-платформа ATutor є локальною навчальною системою. Її інструментарій забезпечує навчальний процес і вирішення цілий ряд проблем, що виникають внаслідок дистанційного навчання, зокрема, проблеми контролю і оцінювання реального рівня знань студентів [25].

Індивідуальний досвід викладачів свідчить про те, що для усунення наявних проблем можуть бути використано багато різноманітних методів, прийомів та технологій навчання/тестування.

Функціонал онлайн-платформи ATutor надає можливість застосування підходів, які сприяють усуненню несамотійного виконання навчальних завдань.

Наприклад, при виконанні тестів і незалежно від виду контролю (поточний, тематичний, підсумковий) студентам за допомогою конструктора тестів ATutor надається максимально обмежений час для виконання тестових завдань, щоб зробити неможливим використання чиеїсь допомоги, отриманої через месенджери чи з різноманітних Інтернет-ресурсів.

Навіть при випадковому виборі вірної відповіді такий підхід є доволі ефективним.

Для забезпечення мотивації студентів щодо опанування навчального контенту online-курсу підходи, які використовуються в ATutor, є доволі ефективними.

Виконання студентами завдань тесту робиться в режимі індивідуалізації тестування (особистісного підходу до кожного студента і об'єктивну оцінку рівня його знань).

1.2.5 Інтерактивна платформа для створення тестів JustClass

JustClass – безкоштовна українська платформа для створення тестів, яка допомагає викладачам швидко організовувати навчальний процес [26].

JustClass – зручний інструмент сучасного навчання, який допомагає зробити навчальний процес більш ефективним, цікавим та організованим (рис. 1.6).

Платформа JustClass має зручний інтерфейс, дозволяючи легко знайти необхідний навчальний контент за дисциплінами та курсами (класами).

Студенти (учні) можуть виконувати завдання в online-режимі, а результати тестування відображаються викладачеві в режимі реального часу.

Платформа JustClass надає зручний спосіб організації дистанційного/змішаного навчання [26].

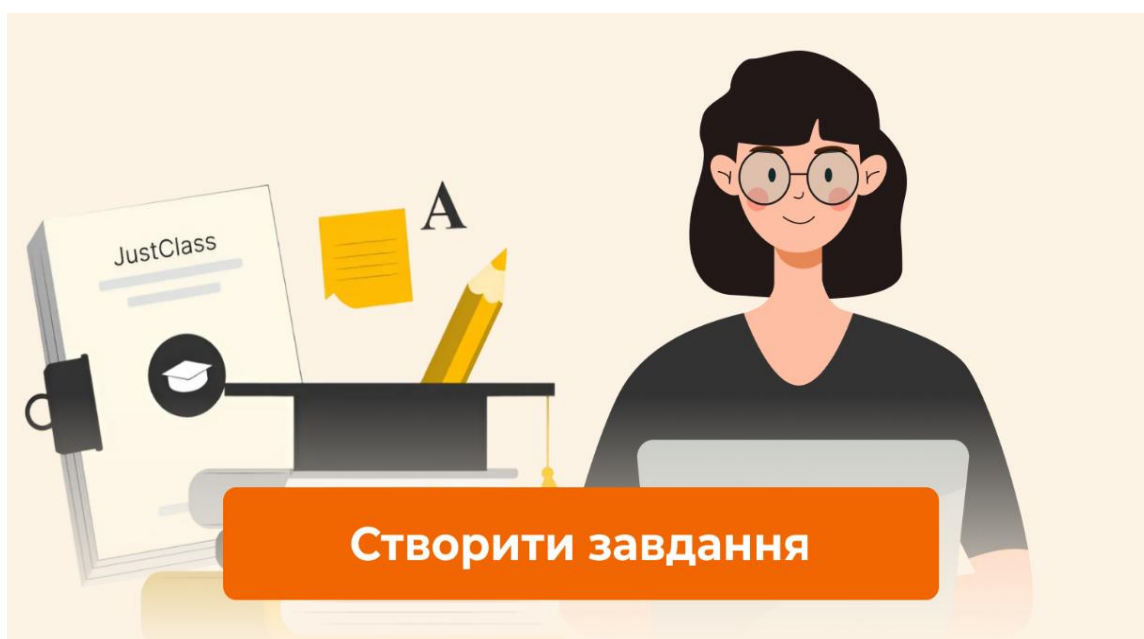


Рисунок 1.6 – Вигляд емблеми інтерактивної платформи JustClass [26]

На платформі зібрано різні завдання (домашні, контрольні, самостійні роботи) та завдання тестів, які відповідають державним стандартам.

Викладач вибирає завдання (з відповідної бази даних), встановлює час виконання завдань тесту.

Завдання, які підтримуються JustClass, можуть містити різноманітні інтерактивні елементи, такі, наприклад, як зображення, відео чи аудіофайли, що робить процес навчання більш цікавим та захоплюючим.

Готові відповіді (рішення завдань) відсутні у загальній мережі, що зазвичай використовується студентами (учнями, курсантами), що мотивує їх самостійно розв'язувати завдання.

Автоматична перевірка результатів економить час викладачів і дозволяє швидко визначити рівень засвоєння студентами тієї чи іншої теми курсу.

На рис. 1.7 продемонстровано приклад створення домашнього завдання для учнів 8-го класу з дисципліни «Алгебра» [26].

Створити нове завдання

Предмет:

Алгебра

Клас:

8

Тип завдання:

Домашні завдання

Відео інструкція

Як створити завдання

[Переглянути >](#)

Тема 1. Раціональні вирази ^

8.1.01А Цілі і дробові вирази. Раціональні вирази. Раціональні дробові

8.1.02А Основна властивість раціонального дробу. Скорочення дробів

8.1.03А Додавання і віднімання раціональних дробів з однаковими знаменниками

8.1.04А Додавання і віднімання раціональних дробів з різними знаменниками

8.1.05А Числові множини

8.1.06А Множення раціональних дробів. Піднесення дробу до степеня

8.1.07А Ділення раціональних дробів

8.1.01А Цілі і дробові вирази. Раціональні вирази. Раціональні дробові

☐ Обрати всі

☐ Пригадай, що таке одночлен та многочлен. Встав пропущені слова.

☐ Визнач, чи є вираз одночленом або многочленом

☐ Спрости вирази та вибери правильні відповіді

☐ Познач зайвий вираз

☐ Визнач, цілим чи дробовим є вираз

☐ Познач правильні відповіді

☐ Склади дробові за умовою та встанови відповідність

☐ Розшифруй закодоване слово

☐ Склади раціональні вирази до задач

☐ Знайди ОЛЗ виразу (Заявляння 1)

Переглянути

Переглянути

Переглянути

Переглянути

Переглянути

Переглянути

Переглянути

Переглянути

Переглянути

Переглянути

Рисунок 1.7 – Приклад створення завдання на платформі JustClass

1.2.6 Online-генератор тестів EasyTestMaker

EasyTestMaker надає можливість створення та керування тестами [27].

EasyTestMaker дозволяє легко формувати різні типи запитань, друкувати альтернативні версії та публікувати їх в Інтернеті для online-тестів. Online-тести оцінюються автоматично.

Функції EasyTestMaker [27]:

- Ідеально відформатовані тести: Створюються питання (завдання) тестів з:
 - вибором однієї правильної відповіді;
 - питання з пропуском місця під відповідь;
 - зіставленням відповідей;
 - короткими відповідями;
 - питання «правда»/«неправда».
- Можливість додавання інструкцій щодо тестування.
- Розділення тесту на кілька підтестів (розділів тесту).
- Звітування про аналіз запитань: Швидкі та прості звіти, щоб побачити, скільки студентів відповіли на конкретні питання правильно, а скільки – неправильно.
- Надання альтернативних версій тесту для зменшення обману: Альтернативні версії та варіанти відповідей автоматично розташовуються в іншому порядку.
- Легке створення тестів для іспитів: Створіть іспит з існуючих тестів можна лише кількома кліками, легко та швидко копіюючи питання з одного тесту до іншого.
- Експорт у файли форматів .docx чи .pdf: Тест, будь-яку його версію, включаючи варіанти відповідей, можна завантажити як документ формату .docx чи .pdf для збереження та друку пізніше.

Online-тести, що вже опубліковані, оцінюються системою EasyTestMaker автоматично.

Викладачі можуть, працюючи з EasyTestMaker [27]:

- переглядати та друкувати результати студентів після проведеного тестування;
- змінювати при необхідності процедуру тестування / оцінювання;
- розробляти:
 - пули запитань;
 - власну базу завдань (питань) тестів;
- опубліковувати свій тест;
- вибирати стільки запитань, скільки потрібно для досягнення (на думку викладача) цілей тестування.

Кожен студент, здійснюючи/проходячи тестування, отримує набір завдань (питань) тесту, який випадково обирається з відповідної таблиці бази тестів.

EasyTestMaker економить час, який викладач витрачає на форматування тестів. Усі тести зберігаються так, що легко можна здійснити їх пошук.

Викладач може вносити зміни або роздруковувати додаткові копії будь-якого створеного ним тесту (чи питання тесту).

Завдяки рандомізації бази слів, що використовуються в тестах, EasyTestMaker також може створити ключ відповідей.

Але EasyTestMaker має й певні недоліки та обмеження у використанні. Крім безкоштовної версії Free, яка надає дуже обмежений функціонал, EasyTestMaker має платні версії Plus та Premium.

Основні функції безкоштовної версії Free для створення тестів:

- простий у використанні інтерфейс тестування;
- доступні всі типи питань;
- варіанти відповідей (так звані «аркуші» відповідей);
- кілька розділів;
- автоматична перенумерація питань під час переміщення питань;

- обмежена кількість тестів – 25 тестів;
- публікація online-тестів з кількістю збережених результатів до 25.

Основні характеристики версії Plus для створення тестів в EasyTestMaker [27]:

- вартість \$59.95/рік;
- всі функції версії Free;
- автоматично згенеровані альтернативні версії тестів;
- конструктор іспитів;
- експорт у файли форматів .docx та .pdf;
- перевірка орфографії текстів тестів;
- виділення жирним шрифтом, виділення, курсив тощо;
- необмежена кількість тестів;
- публікація online-тестів з кількістю збережених результатів до 250.

Основні характеристики версії Premium для створення online-тестів в EasyTestMaker [27]:

- вартість \$79.95/рік;
- всі функції версії Plus;
- завантаження зображень, діаграм або графіків\$
- автоматичне оцінювання результатів;
- встановлення часових обмежень для online-тестів;
- публікація окремих результатів тестів з оцінюваннями;
- експорт результатів у файл MS Excel/CSV;
- публікація тестів для online-тестування, що зберігаються;
- одночасне зберігання до 2500 результатів;
- формування детальних звітів з аналізом результатів тестування;
- перегляд відповідей, які були обрані, та ким були обрані для кожного питання;
- формування зведених звітів;
- надання:
 - розширених параметрів online-тестування;

- тестів з обмеженим часом виконання;
- параметрів завантаження тощо.

Аналіз роботи розглянутих систем вказує на потребу у розробці індивідуалізованої системи для проведення тестування студентів, що, зокрема:

- відповідала б потребам кожного студента;
- стимулювала б студентів до досягнення більш високих результатів у навчанні.

Одним з прикладів такої системи є платформа Knewton [28], яка використовує інтелектуальний аналіз даних та машинне навчання для створення індивідуалізованих навчальних курсів.

Knewton збирає дані про прогрес та відповіді кожного учня, а потім аналізує ці дані для розуміння потреб та здібностей студентів [28].

Knewton використовує ці дані для створення персоналізованих навчальних матеріалів та завдань.

Кожен студент отримує індивідуальний навчальний план, який відповідає його потребам і рівню знань.

Система адаптується до кожного студента, надаючи тестові завдання та допоміжний навчальний контент, що допомагає засвоїти новий матеріал та покращити навички та рівень теоретичних знань.

Крім того, Knewton надає зворотний зв'язок та рекомендації студентам щодо подальшого навчання.

Вона допомагає студентам виявити свої слабкі місця та працювати над ними, а також визначити свої сильні сторони і розвивати їх [28, 29].

На рис. 1.8 продемонстровано основне вікно системи Knewton та її мобільний варіант.

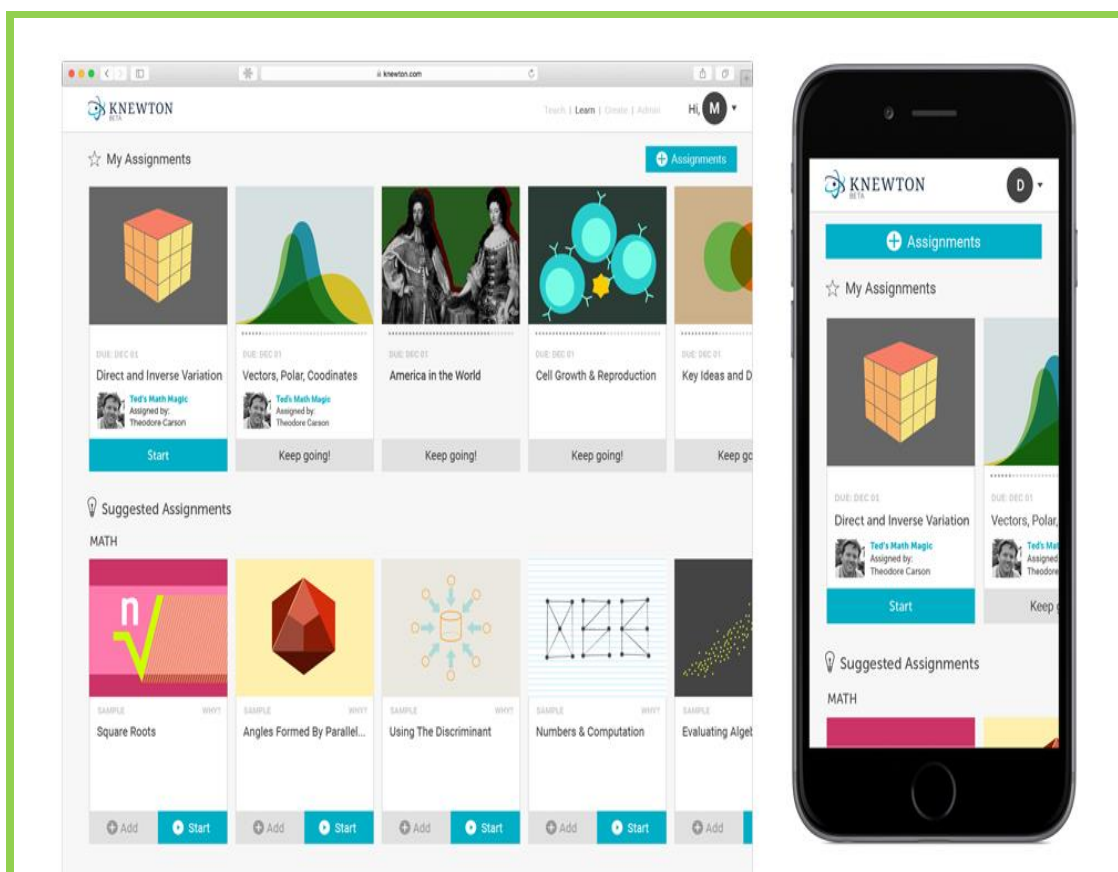


Рисунок 1.8 – Основне вікно Knewton

Ця система є прикладом того, як гейміфікація може бути застосована в індивідуалізованому навчанні та стимулювати учнів до досягнення успіху у навчанні.

Однак, вона має свої обмеження, такі як висока вартості та складність використання [29].

Усі розглянуті приклади веб-орієнтованих систем тестування знань студентів демонструють, як гейміфікація та індивідуалізація навчання можуть бути успішно використані в різних контекстах.

Кожна з цих платформ надає учням можливість навчатися власним темпом, отримувати миттєвий зворотний зв'язок та бути мотивованими досягати нових цілей у навчанні.

2 АНАЛІЗ ТА ШЛЯХИ РОЗВ'ЯЗКУ ПРОБЛЕМИ

2.1 Використання нечіткої логіки для тестування знань студентів

Для реалізації системи контролю знань студентів потрібно розглянути алгоритмічне та математичне забезпечення таких систем.

Для підвищення якості оцінювання знань студентів важливим аспектом є об'єктивність самого процесу.

Для цього доцільно використовувати відповідні програмні рішення (системи, платформи, сервіси), що надають можливість розв'язувати широке коло не тільки простих завдань з тестів, а й вирішувати ще й деякі інтелектуальні задачі.

Завдання тестування знань є завданням віднесення результатів тестування до певного класу оцінок, тому його можна трактувати як класичне інтелектуальне завдання класифікації [30], вирішення якого можна здійснити, використовуючи відомі методи класифікації [31 – 33].

При цьому, результати тестування можуть бути представлені нечіткою слабко структурованою множиною.

Найчастіше для розв'язання подібних завдань тестування знань студентів використовуються:

- методи штучного інтелекту [34];
- штучні нейронні мережі [35];
- метод групового врахування аргументів, що ґрунтується на рекурсивному селективному відборі моделей, на основі яких будуються більш складні моделі [36].

Штучні нейронні мережі потребують багато часу на своє навчання («з вчителем» чи «без вчителя»).

Тому при збільшенні кількості параметрів, що класифікуються при тестуванні, час отримання результату тестування може суттєво зрости.

Все це ускладнює використання штучних нейронних мереж при тестуванні знань студентів (визначення рівня знань з наданого навчального контенту), коли важливо за обмежений проміжок часу:

- оцінити знання студентів;
- урахувати кількість параметрів та характеристик завдань (питань) тестів, що використовуються при проходженні студентами тестування.

Слід також враховувати той факт, що штучні нейронні мережі не можуть враховувати особливості використання опанованого теоретичного навчального контенту до:

- вирішення практичних задач;
- вирішення завдань (надання відповіді на запитання тестів) при проходженні студентами відповідного тесту.

Крім того, цей метод не надає можливості оперувати нечіткими чи слабко структурованими даними.

Крім того, для роботи програмних засобів, що реалізують методи штучного інтелекту, необхідні:

- потужні апаратні ресурси:
 - процесору;
 - оперативної пам'яті;

що накладає певні обмеження на їх використання, наприклад, на мобільних пристроях [37].

Тому проблема розробки підходу до вирішення інтелектуальної задачі тестування знань студентів з використанням методів, систем та засобів нечіткої логіки:

- є актуальною в наш час не до кінця визначених знань;
- надає можливість приймати рішення при оцінювання знань в умовах обмеженої визначеності.

До переваг використання нечіткої логіки, для задачі тестування знань, можна віднести, зокрема, такі [38 – 40]:

- можливість обробляти дані, заданими нечітко: наприклад,

відповідями на запитання, що потребують розгорнутої відповіді природною мовою;

- можливість нечіткої формалізації критеріїв оцінювання знань студентів.

При цьому важливим є забезпечення самотестування знань студентом, щоб він міг, зокрема:

- оцінити складність та повноту тестів з означеної теми;
- отримати пояснення щодо виконаних неправильно завдань тестів;
- отримати рекомендації щодо удосконалення своїх знань відповідно до необхідної деталізації їх вивчення.

Серед існуючих варіантів програмних рішень щодо тестування знань студентів в повній мірі означений функціонал ще не до кінця знаходить своє відображення.

Але використання такого підходу дає можливість оцінювання рівня знань студентів за результатами аналізу розгорнутих відповідей на складні питання тестів.

Алгоритм тестування знань можна продемонструвати діаграмою активності (рис. 2.1), що включатиме:

- здійснення активації застосунка та всіх супутніх сервісів;
- реєстрація користувача (викладача, студента тощо);
- визначення режиму тестування.

Така технологія тестування знань:

- базується на використанні нечіткої логіки;
- дає можливість адекватно оцінити знання студентів, оперуючи даними, що були отримані при тестуванні;

- передбачає:

- самотестування для оцінювання студентом своїх знань;
- отримання рекомендацій щодо їх удосконалення.

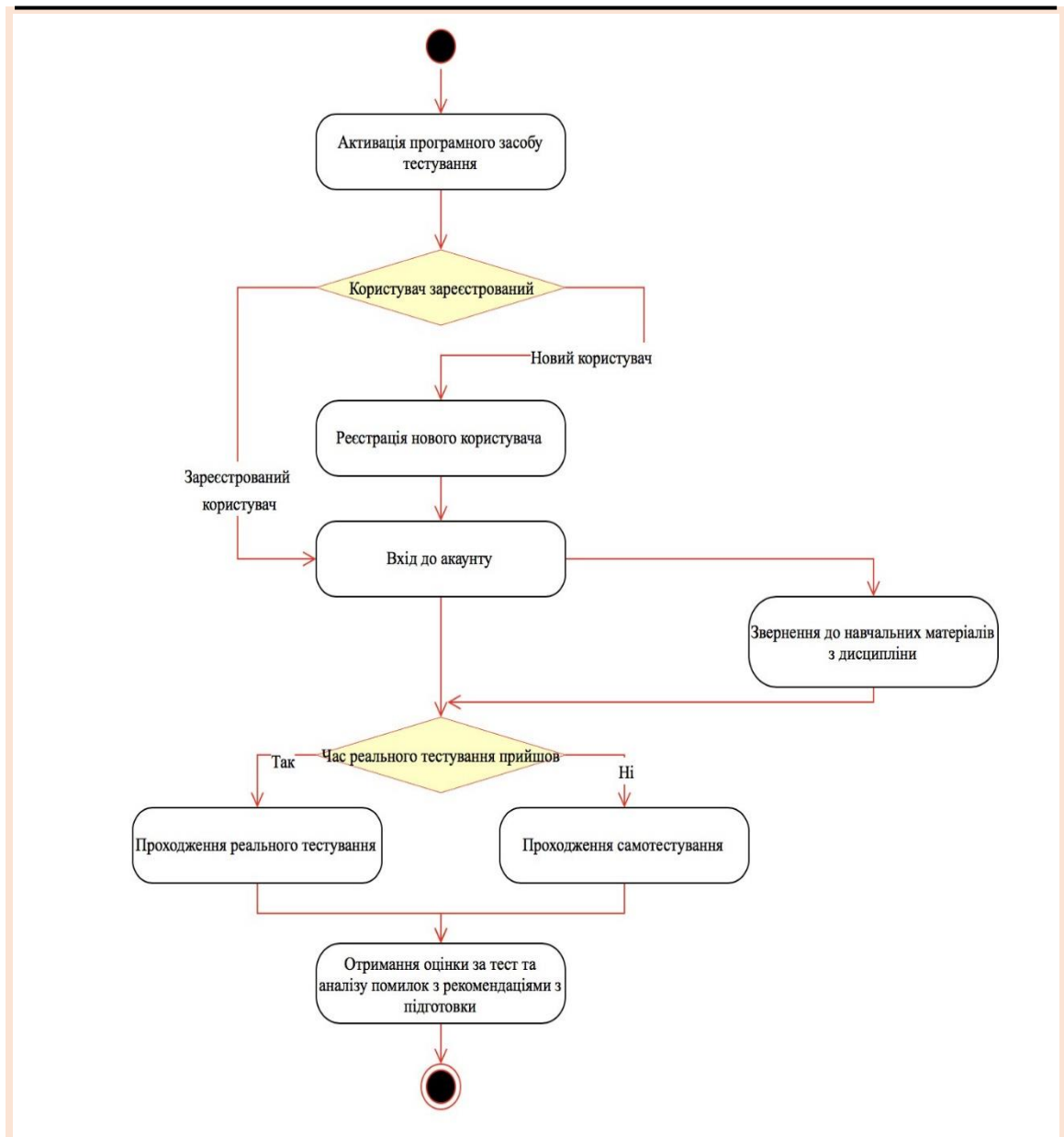


Рисунок 2.1 – Діаграма активності для алгоритму тестування знань

2.2 Онтологічне моделювання процесів тестування знань студентів

2.2.1 Функції системи тестування, що використовує онтологічну модель

Основні функції системи включають авторизацію користувачів, доступ до тестів різних рівнів складності для студентів, та панель викладача з редакторами тестів та питань, а також візуалізацію результатів.

1. *Авторизація користувачів.*

Для авторизації користувачів (студентів та викладачів) потрібно використовувати алгоритм аутентифікації, який перевіряє відповідність введеного логіну та паролю збереженим в базі даних [4].

2. *Доступ до тестів для студентів.*

Студент після авторизації повинен мати доступ до списку тестів з різних дисциплін та рівня складності. Алгоритм відображення тестів повинен включати фільтрацію за дисципліною та рівнем складності.

3. *Отримання результатів тестування студентом.*

Після проходження тесту студент повинен отримати результат на екрані. Це має бути реалізовано через алгоритм, який підраховує бали студента за тест та відображає їх.

4. *Доступ до панелі викладача.*

Викладач після авторизації має мати доступ до панелі викладача, де він може переглядати та редагувати тести, питання та результати студентів. Алгоритм відображення цієї панелі має включати перевірку прав доступу викладача.

5. *Редагування тестів та питань.*

Редактор тестів та питань для викладача має використовувати алгоритми для додавання, видалення та редагування тестів та питань у системі.

6. *Візуалізація результатів.*

Для візуалізації результатів використовується алгоритм, який генерує графіки, таблиці або інші візуальні елементи для відображення успішності студентів, результатів тестів тощо.

2.2.2 Онтологічна модель системи тестування знань студентів

На рис. 2.2 зображена ієрархія класів онтології [41, 42].

Ця онтологія розроблена для ефективного управління та контролю над процесом навчання студентів в середовищі Protégé [43].

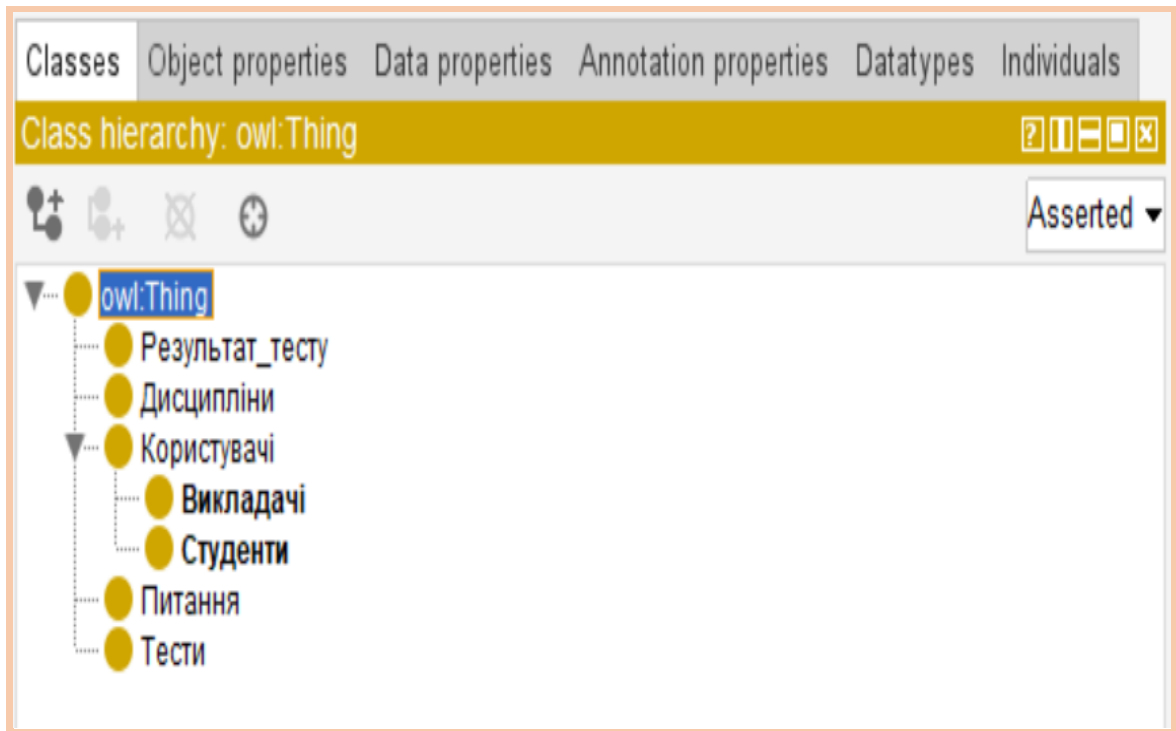


Рисунок 2.2 – Класи онтології

На рис. 2.3 зображені властивості даних цих класів онтології системи тестування знань студентів.

Ця система тестування на основі онтологічного моделювання процесів тестування дозволяє зберігати дані про:

- різних користувачів, таких як викладачі та студенти;
- про дисципліни, за навчальним контентом яких здійснюється тестування;
- тести, що надаються студентам;
- результати тестування студента (групи студентів).

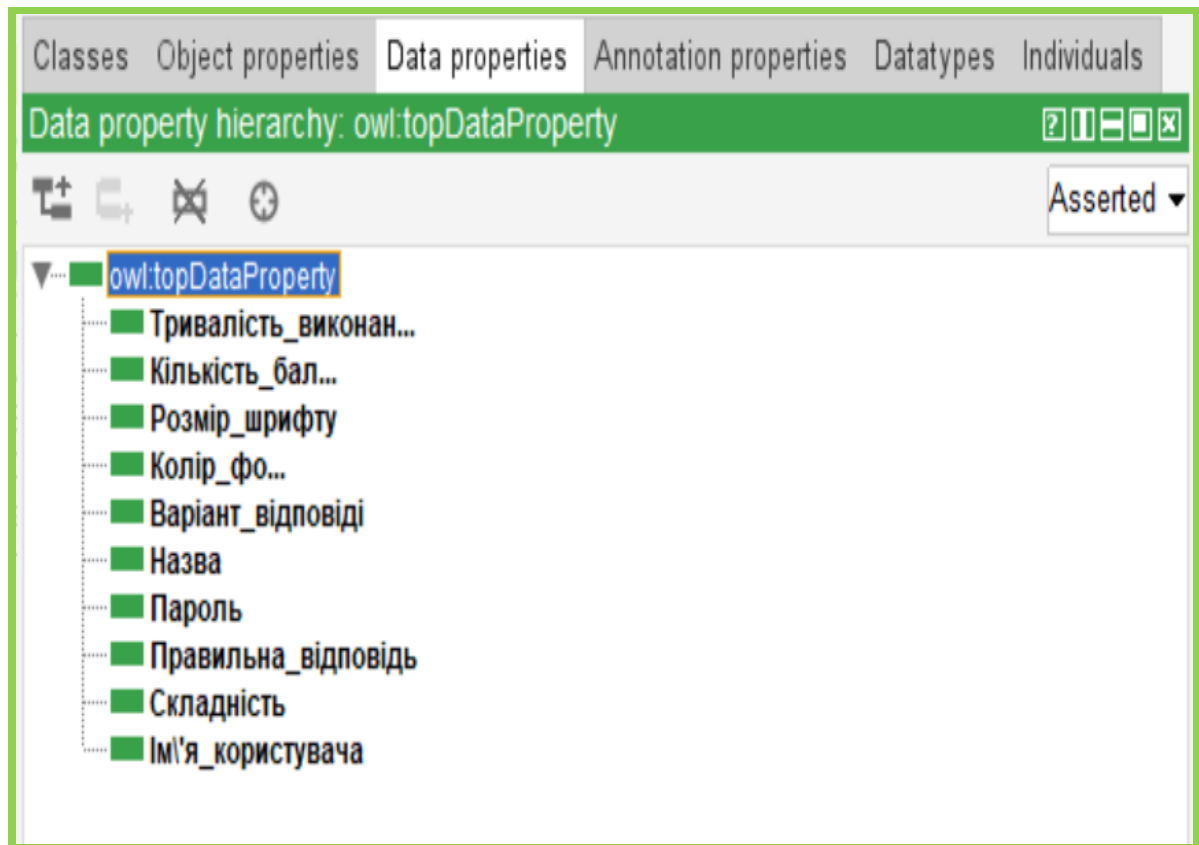


Рисунок 2.3 – Властивості даних онтології

Клас "Результат_тесту" забезпечує можливість фіксувати час виконання тесту, кількість набраних балів та пов'язану з ним дисципліну.

Це дозволяє викладачам аналізувати та оцінювати результати тестування студентів.

Клас "Дисципліни" відображає назву предмету, за яким створюються тести. Це допомагає в упорядкуванні та категоризації тестів за різними дисциплінами.

Клас "Користувачі" дозволяє користувачам входити до системи з вказанням імені користувача та пароля, а також налаштовувати розмір шрифту та колір фону інтерфейсу.

Це забезпечує комфортні умови користування системою для кожного користувача.

Класи "Викладачі" та "Студенти" наслідують клас "Користувачі" і дозволяють відокремлено зберігати дані про викладачів та студентів.

Це дозволяє кожній групі користувачів мати свої власні права та можливості в системі.

Питання включають в себе варіанти відповідей та правильну відповідь, що робить можливим створення тестів з різними варіантами відповідей.

Це дозволяє викладачам створювати різноманітні тести для перевірки знань студентів.

Клас "Тести" дозволяє вказати складність тесту та пов'язану з ним дисципліну, а також містить список питань, що входять до тесту.

Це дозволяє викладачам легко керувати тестами та їх параметрами.

Для опису зв'язків між об'єктами та їхніх характеристик в контексті конкретної предметної області в онтології створюються об'єктні властивості.

Вони допомагають структурувати та організувати інформацію, що дозволяє ефективно виконувати операції з даними.

На рис. 2.4 наведено об'єктні властивостей у контексті онтології підсистеми контролю знань студентів для інтелектуальної системи «Освітній процес».

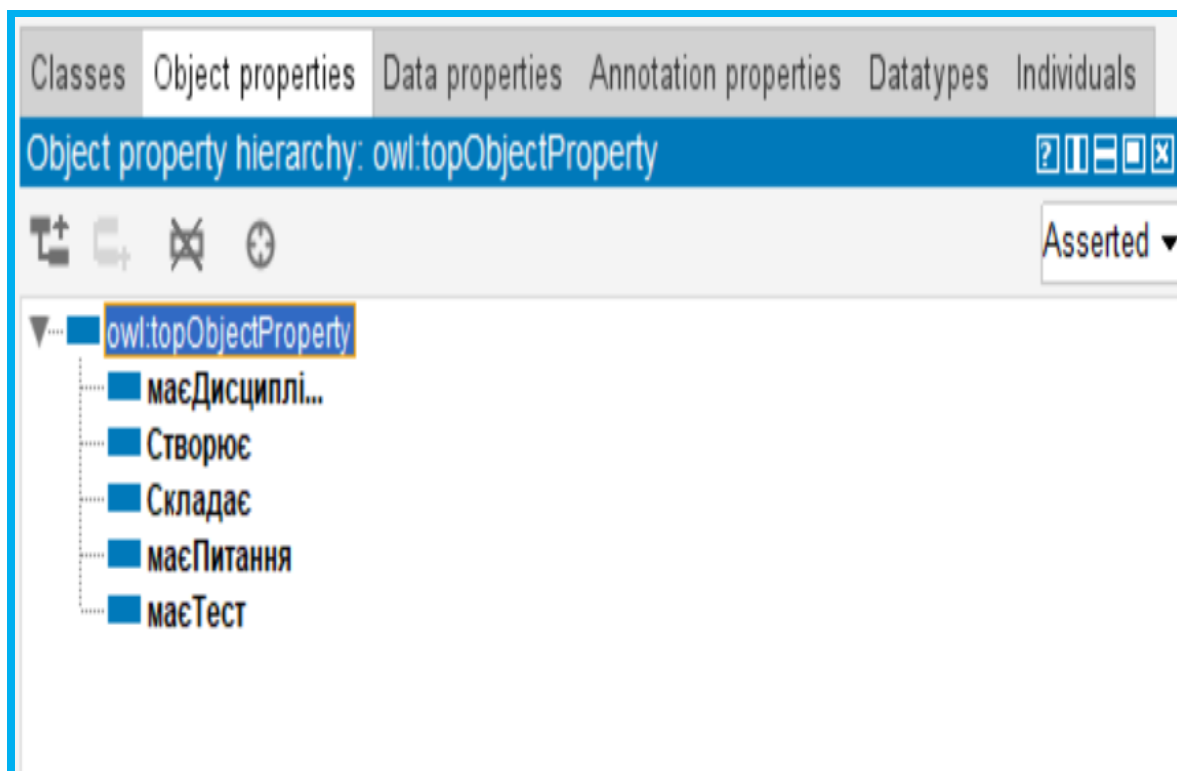


Рисунок 2.4 – Об'єктні властивості онтології

В онтології об'єкти взаємодіють між собою за допомогою ряду властивостей, які визначають їхні зв'язки та функції.

Кожен тест має властивість "Дисципліна", яка вказує на предмет або курс, до якого він відноситься. Це дозволяє групувати тести за дисциплінами, спрощуючи організацію та пошук в системі.

Викладачі можуть створювати тести та питання, що відображається в зв'язку "Створює". Ця властивість вказує на те, що викладачі відповідають за створення тестів та їхніх складових елементів.

Студенти складають тести, що відображається в зв'язку "Складає".

Ця властивість показує, що студенти можуть проходити тести, створені викладачами.

Кожен тест має властивість "МаєПитання", яка вказує на питання, які входять до цього тесту.

Ця властивість дозволяє організувати тести з різними питаннями та відповідями.

Кожне питання також має властивість "МаєТест", яка вказує на тест, до якого воно належить.

Це дозволяє відстежувати зв'язок між питаннями та тестами, спрощуючи редагування тестів та їхніх складових частин.

Завдяки цій онтології (рис. 2.5):

- студенти можуть проходити тестування та отримувати результати;
- викладачі можуть легко створювати тести, вести контроль над навчальним процесом, забезпечуючи ефективність та якість навчання.

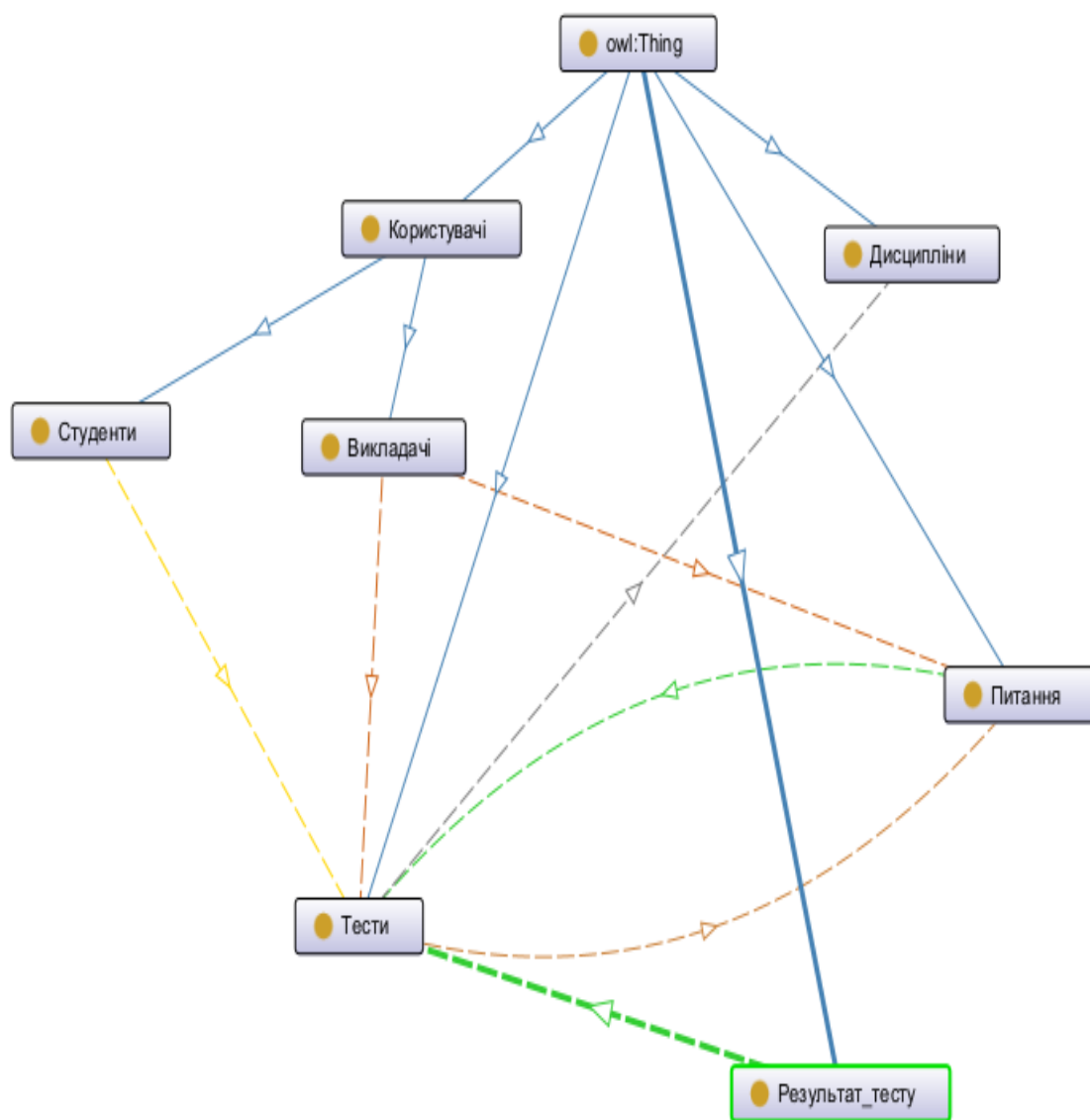


Рисунок 2.5 – Онтологічна модель системи тестування, що пропонується

3 ОПИС ПРОГРАМНОГО ПРОДУКТУ

3.1 Система тестування знань студентів: Інструкція програміста

Розроблена програма системи контролю знань студентів була реалізована за допомогою мови програмування C# [44] та технології Windows Forms (WinForms) [45]. C# є об'єктно-орієнтованою мовою програмування, яка дозволяє ефективно створювати різноманітні застосунки для платформи Windows, включаючи інструменти для роботи з графічним інтерфейсом користувача.

WinForms є одним з найпоширеніших інструментів для створення графічних інтерфейсів у програмах для операційної системи Windows.

Ця технологія надає розробникам зручний спосіб створення віконних додатків із різноманітними елементами керування, такими як кнопки, текстові поля, таблиці тощо [46].

Використання C# та WinForms сприяло розробці програмного продукту, що має зручний інтерфейс користувача та доволі високий рівень повноти функцій, які ним підтримуються.

Крім того, завдяки Windows, програмний продукт може бути інтегрована в інші системи, що використовуються в навчальному процесі.

3.1.1 Класи системи тестування

Клас **Test** слугує для представлення тесту, його організації тестів та забезпечення зручного доступу до їх властивостей та питань.

Кожен екземпляр цього класу має властивості: **Name** – назва тесту, **Subject** – предмет тесту, тобто предметна область знань, до якої він належить, **Difficulty** – складність тесту, виражена цілим числом, і **Questions** – список питань, які входять до цього тесту.

Конструктор класу `Test` дозволяє ініціалізувати ці властивості при створенні нового екземпляру, визначаючи значення його властивостей, та створює порожній список питань.

Метод `AddQuestion` надає можливість додавати нові питання до цього списку. Перевизначений метод `ToString` повертає у вигляді рядка назву тесту. Клас `Test` служить.

Клас `Question` визначає структуру окремого питання у тесті. У цьому класі є наступні властивості: `Name` – рядок, що містить текст питання, `Answers` – список рядків, який містить варіанти відповідей на питання, і `CorrectAnswer` – рядок, що містить правильну відповідь на питання.

Конструктор класу `Question` створює новий екземпляр класу та ініціалізує порожній список `Answers`, для зберігання варіантів відповідей.

Перевизначений метод `ToString` повертає текст питання у вигляді рядка. Клас `Question` дозволяє представляти питання та варіанти відповідей на них (для використання у тестах).

Клас `TestManager` відповідає за завантаження та збереження тестів у файл. Він має два методи: `LoadTests()` та `SaveTests(List<Test> tests)`.

Метод `LoadTests()` перевіряє, чи існує за заданим шляхом файл з тестами. Якщо файлу немає, то метод повертає порожній список тестів. Якщо файл існує, то відбувається його зчитування за допомогою `StreamReader`, потім використовується `JsonSerializer` для десеріалізації об'єктів типу `Test`. Результатом є список завантажених тестів.

Метод `SaveTests(List<Test> tests)` отримує список тестів як параметр і зберігає їх у файл. Він створює новий файл або перезаписує вміст існуючого файлу за допомогою `StreamWriter`. За допомогою `JsonSerializer`, тести серіалізуються у JSON-формат та записуються у файл [47].

Клас `TestResult` представляє результати виконання конкретного тесту. У цьому класі є наступні властивості: `TestName` – містить назву тесту, для якого були надані результати, `Points` – ціле число, що відображає кількість набраних балів за виконання тесту, `Duration` – об'єкт класу `TimeSpan` і вказує

тривалість тестування. Конструктор класу ініціалізує ці властивості при створенні нового екземпляру класу `TestResult`.

Клас `User` представляє користувача системи тестування. У цьому класі є такі властивості: `Username` – рядок з ім'ям користувача, `Password` – рядок, що містить пароль користувача, `FontSize` – ціле число, що відображає розмір шрифту тексту питань (завдань) тесту, `BackgroundColor` – об'єкт класу `Color`, який відображає колір тла тексту питань тестів, `TestResults` – список об'єктів класу `TestResult`, що містить результати тестування студентів. Конструктор класу `User`:

- ініціалізує обов'язкові властивості користувача, такі як ім'я та пароль;
- встановлює значення для `FontSize` та `BackgroundColor`;
- створює порожній список результатів тестування студентів.

Клас `UserManager` відповідає за управління користувачами системи тестування. У цьому класі зберігається список користувачів у властивості `Users`, і шлях до файлу з даними студентів у властивості `FilePath`. Конструктор `userManager()` викликає метод `DeserializeUsers()`, щоб завантажити дані користувачів з файлу при створенні об'єкта `UserManager`.

Метод `DeserializeUsers()` перевіряє, чи існує файл з користувачами за заданим шляхом. Метод `SerializeUsers()` використовується для збереження списку користувачів у файлі.

Метод `AddUser(string username, string password)` додає нового користувача до списку, метод `AuthenticateUser(string username, string password)` перевіряє, чи існує користувач з вказаним ім'ям та паролем. Метод `ReturnUser(string username, string password)` повертає об'єкт користувача за вказаним ім'ям та паролем. `UpdateUserSettings(User user)` оновлює налаштування користувача і зберігає їх у відповідному файлі.

3.1.2 Form1 (Форма тестування)

Форма тестування (Form1) є головним вікном розробленого програмного продукту, що використовується для проведення тестування знань студентів. На цій формі відображаються питання та варіанти відповідей. Студент може вибирати відповіді за допомогою RadioButton. Після завершення тестування виводяться його результати, зокрема кількість набраних балів та час проходження тесту.

Функція InitializeMenuStrip` призначена для ініціалізації головного меню розробленого програмного продукту. Спочатку вона створює новий об'єкт ToolStripMenuItem з назвою «Предмети» і додає його як підпункт до пункту «Тести» (testsToolStripMenuItem).

Для кожного предмету (дисципліни, курсу), знайденого серед тестів в базі даних програмного продукту – системи тестування знань студентів – створюється новий підпункт ToolStripMenuItem з назвою предмету, який додається до підпункту «Предмети». Для кожного тесту з поточного предмету створюється підпункт ToolStripMenuItem з назвою тесту, який призначається, наприклад, обробнику подій Click, зазначеному в методі testToolStripMenuItem_Click, і додається до відповідного підпункту предмету.

Нарешті створюється пункт «Вихід» (exitItem), призначається обробник подій Click exitToolStripMenuItem_Click та додається до головного меню програми (MainMenuStrip). Ця функція створює та налаштовує меню, використовуючи дані з бази даних, та додає кнопку «Вихід» до головного меню програми.

ClearQuestions():

- відповідає за очищення попередніх відповідей та питань;
- забезпечує зняття позначки (unchecked) з RadioButton1, RadioButton2, RadioButton3;
- встановлює порожній рядок для мітки питання (questionLabel.Text), очищаючи елементи інтерфейсу користувача перед наступним питанням.

DisplayQuestion() відповідає за відображення поточного питання тесту на екрані. Спочатку викликається функція ClearQuestions() для очищення попередніх відповідей та питання. Потім вона:

- отримує питання тесту за допомогою змінної currentQuestionIndex;
- оновлює відповідні мітки з текстами питання та варіантами відповідей (questionLabel.Text, RadioButton1.Text, RadioButton2.Text, RadioButton3.Text).

Студент може переглянути питання і вибрати один з варіантів відповіді.

Функція testToolStripMenuItem_Click є обробником подій для події «Click» на об'єкті ToolStripMenuItem з ім'ям testToolStripMenuItem. Вона викликає функцію ClearQuestions() для очищення попередніх даних про питання та відповіді, встановлює значення змінних points та currentQuestionIndex рівними 0.

Потім отримується ім'я предмета і тесту з об'єктів ToolStripMenuItem, потім знаходиться вибраний тест у базі даних. Назви предмета та тесту відображаються у відповідних мітках, також оновлюється назва групи. За допомогою функції ShowAllControls всі елементи у групі стають видимими.

Потім викликається функція DisplayQuestion() для відображення першого питання тесту, і зберігається час початку тестування. Основним для цієї функції є підготовка інтерфейсу для початку нового тесту (очистивши попередні дані), відображення вибраного тесту і першого питання, відстеження часу початку тесту.

Функція testToolStripMenuItem_Click викликається при кліку на пункт меню з тестом і відповідає за ініціалізацію нового тесту. Спочатку вона викликає функцію ClearQuestions() для очищення попередніх відповідей та скидання змінних points і currentQuestionIndex на початкові значення.

Функція nextQuestionButton_Click викликається кнопкою «Наступне питання». Спочатку вона отримує відповідь студента на питання за допомогою функції GetSelectedAnswer. Потім перевіряє правильність відповіді,

порівнюючи її з відповідним еталоном. Якщо відповідь правильна, то кількість балів збільшується на значення, яке відповідає складності тесту.

Далі перевіряється, чи досягнуто останнього питання в тесті. Якщо ні, то індекс поточного питання збільшується на 1, і відображається наступне питання за допомогою функції `DisplayQuestion()`. Якщо так, то виводиться повідомлення про завершення тесту з кількістю балів та часом виконання. Дані про результат тесту зберігаються у базі даних.

Метод `HideAllControls(Control control)` рекурсивно приховує всі елементи у контейнері `control` та всі його дочірні елементи. Він змінює властивість `Visible` на `false` для кожного елемента. Метод `ShowAllControls(Control control)` робить всі елементи у контейнері `control` та всі його дочірні елементи видимими. Він змінює властивість `Visible` на `true` для кожного елемента.

Метод `GetSelectedAnswer(GroupBox groupBox)` повертає текст вибраної відповіді на питання, що містяться у `GroupBox`. Він шукає перший вибраний елемент типу `RadioButton` у `GroupBox` та повертає текст цього елемента. Метод `OnFormClosing(FormClosingEventArgs)` викликається перед закриттям форми і зберігає дані про тести у базі даних.

3.1.3 LoginForm

`LoginForm` відповідає за аутентифікацію та реєстрацію користувачів. При натисканні кнопки для входу, `LoginForm` перевіряє введені дані та, якщо вони вірні, надає доступ користувачу до системи тестування. Крім того, вона дозволяє створювати нові облікові записи користувачів при натисканні кнопки реєстрації, забезпечуючи безпечний та контрольований доступ до системи.

У методі `loginButton_Click` виконується обробка події натискання кнопки для входу користувача. Спочатку отримуються значення імені користувача та пароля з відповідних текстових полів. Після цього

викликається метод `AuthenticateUser` з об'єкта `userManager` для перевірки введених даних.

Якщо користувач з вказаними ім'ям та паролем існує, метод повертає `true`, і в цьому випадку користувача визначається як поточного користувача (`currentUser`), і відображається повідомлення про його успішний вхід до системи.

Якщо користувача з введеними даними не знайдено, то виводиться повідомлення про помилку аутентифікації.

При успішному вході значення `DialogResult` встановлюється на `DialogResult.OK`, щоб позначити успішне завершення роботи з відповідним діалоговим вікном.

У методі `registerButton_Click` виконується обробка події натискання кнопки для реєстрації нового користувача. Спочатку отримуються значення імені користувача та пароля з відповідних текстових полів.

Потім викликається метод `AddUser` з об'єкта `userManager` для додавання нового користувача з введеними даними.

Після цього відображається повідомлення про успішне додавання користувача.

3.1.4 SettingsForm

`SettingsForm` дозволяє користувачам змінювати та зберігати налаштування. При натисканні кнопки «Зберегти», вона зберігає розмір шрифту користувача та оновлює його налаштування. Крім того, вона надає можливість користувачам вибирати колір фону і оновлювати налаштування кольору фону.

У методі `saveButton_Click` відбувається обробка події натискання кнопки для збереження налаштувань поточного користувача. Спочатку змінюється розмір шрифту користувача на значення, що вибрано в

fontSizeNumericUpDown. Потім викликається метод UpdateUserSettings з об'єкта userManager для оновлення налаштувань поточного користувача.

Після цього значення DialogResult встановлюється на DialogResult.OK, щоб позначити успішне завершення опрацювання діалогового вікна, після цього вікно закривається.

У методі button1_Click відбувається обробка події натискання кнопки, що викликає відкриття діалогового вікна вибору кольору. Якщо користувач вибрав колір і підтвердив свій вибір, тоді значення BackgroundColor поточного користувача (студента чи викладача) оновлюється з новим (вибраним) кольором.

3.1.5 QuestionForm

У формі QuestionForm користувач може керувати питаннями для тестів, наприклад, порядком їх надання для тестування. При натисканні кнопки «Додати питання» він може додати нове питання до обраного тесту, вказавши текст питання, варіанти відповідей та правильну відповідь. Крім того, він може редагувати або видаляти існуючі питання. Форма також надає можливість створювати нові тести та видаляти існуючі. Всі зміни зберігаються у базі даних.

У методі addQuestionButton_Click відбувається обробка події натискання кнопки для додавання нового питання до обраного тесту. Спочатку перевіряється, чи був обраний який-небудь тест зі списку testsListBox.

Якщо так, то отримується обраний тест, і створюється новий об'єкт питання за допомогою введених користувачем даних: текст питання, варіанти відповідей та правильну відповідь. Після цього нове питання додається до списку питань обраного тесту, і його представлення додається до списку questionsListBox.

Крім того, поля для введення питання очищаються для можливості додавання наступного питання. На завершення, дані тестів зберігаються за допомогою виклику методу `SaveTests` з об'єкта `testManager`.

У методі `saveQuestionButton_Click` відбувається обробка події натискання кнопки для збереження змін у вибраному питанні тесту.

Спочатку перевіряється, чи був обраний який-небудь тест зі списку `testsListBox`, а потім перевіряється, чи було обрано деяке питання зі списку `questionsListBox`.

Якщо обидві умови виконуються, відбувається оновлення даних обраного питання на основі введених користувачем даних: текст питання, варіанти відповідей та правильну відповідь.

Після цього відбувається оновлення відображення у списку тестів та у списку питань, щоб відобразити оновлені дані. На завершення, дані тестів зберігаються за допомогою виклику методу `SaveTests` з об'єкта `testManager`.

У методі `deleteQuestionButton_Click` відбувається обробка події натискання кнопки для видалення обраного питання з тесту. Спочатку отримуються обраний тест і обране питання з відповідних списків `testsListBox` і `questionsListBox`.

Потім обране питання видаляється зі списку питань обраного тесту за допомогою методу `Remove`, і видаляється його відображення з `questionsListBox`.

На завершення, дані тестів зберігаються за допомогою виклику методу `SaveTests` з об'єкту `testManager`.

У методі `testsListBox_SelectedIndexChanged` відбувається обробка події зміни вибраного елементу зі списку тестів (`testsListBox`).

Спочатку очищається вміст списку питань (`questionsListBox`). Потім отримується обраний тест зі списку тестів, і якщо тест не є нульовим, відбувається перебір усіх питань обраного тесту.

Кожне питання додається до списку питань (questionsListBox) для відображення його назви. Це забезпечує оновлення списку питань у відповідності до обраного тесту.

У методі questionsListBox_SelectedIndexChanged відбувається обробка події зміни вибраного елемента у списку питань (questionsListBox). Спочатку отримується обране питання зі списку питань.

Якщо питання не є нульовим, то дані про назву питання, варіанти відповідей і правильну відповідь цього питання заповнюють відповідні текстові поля (questionTextBox, firstAnswerTextBox, secondAnswerTextBox, thirdAnswerTextBox і correctAnswerTextBox).

Це забезпечує відображення даних обраного питання у відповідних текстових полях для можливості їх редагування чи перегляду.

У методі deleteTestButton_Click відбувається обробка події натискання кнопки для видалення обраного тесту зі списку тестів. Спочатку перевіряється, чи був обраний який-небудь тест зі списку testsListBox. Якщо так, то обраний тест видаляється зі списку тестів за допомогою методу Remove, і також видаляється його відображення з testsListBox.

Крім того, список питань (questionsListBox) очищається, бо питання, які належать до видаленого тесту, також стають неактуальними. Дані щодо тестів зберігаються за допомогою виклику методу SaveTests з об'єкту testManager.

За допомогою методу newTestButton_Click відбувається обробка події натискання кнопки для створення нового тесту.

Спочатку отримуються дані для нового тесту з текстових полів subjectNewTestTextbox, nameNewTestTextBox та значення difficultyNumericUpDown з відповідного вікна системи. Обов'язково перевіряється, щоб жодне з полів не було порожнім.

Якщо хоча б одне з полів порожнє, виводиться повідомлення про необхідність введення предмету і назви тесту, а виконання методу припиняється. Якщо всі поля заповнені, створюється новий об'єкт тесту за

допомогою введених даних, і цей тест додається до списку тестів (DataBank.tests) та відображається у списку тестів (testsListBox).

На завершення всі дані тестів зберігаються у базі даних за допомогою виклику методу SaveTests з об'єкту testManager.

3.1.6 ChartForm

Форма ChartForm системи тестування дозволяє візуалізувати дані щодо тестування – відображати результати тестування у вигляді графіків та таблиць.

Користувач може переглядати результати проходження тестів за допомогою гістограм чи таблиць, які відображають кількість балів, отриманих у різних тестах.

Графіки автоматично масштабуються для більш кращої візуалізації даних (видимості), а таблиці надають детальну інформацію про результати тестування для кожного користувача та за кожен предмет чи конкретну тему предмету [48].

У методі InitializePictureBox створюється новий екземпляр PictureBox з ім'ям pictureBox1. Цей елемент управління призначений для відображення зображень.

Параметр Dock встановлюється на значення DockStyle.Fill, щоб зображення займало всю доступну площу форми.

Параметр SizeMode встановлюється на значення PictureBoxSizeMode.Zoom, щоб зображення автоматично масштабувалося для вміщення у розміри PictureBox, зберігаючи відповідні пропорції.

Потім PictureBox додається на форму, щоб її було видно на екрані.

Метод DrawPointsChart спочатку викликає метод InitializePictureBox, який ініціалізує новий елемент PictureBox. Після цього створюється новий об'єкт класу Chart, який буде відображати графік. Розмір графіка встановлюється таким же самим, як і розмір PictureBox.

Потім додається область для графіка (що відображає результати тестування), яка називається «Points». Це надає можливість налаштування параметрів відображення графіка.

Потім додається сюди й серія даних для вибраного тесту.

Для кожного користувача, крім адміністратора, обчислюється загальна кількість балів, отриманих у вибраному тесті. Ці дані додаються до серії, що дозволяє побудувати гістограму кількості балів за кожного користувача.

Також виконуються налаштування відображення графіка, такі як:

- відключення великої сітки для обох вісей;
- встановлення інтервалу між значеннями вісі X.
- додавання заголовку для графіка, який відображає назву вибраного тесту.

Нарешті, створюється зображення з графіка у форматі Bitmap, щоб можна було відобразити його у PictureBox. Після цього зображення встановлюється у PictureBox для відображення зображення цього графіку.

Важливим тут є те, що після використання графіка користувачем (перегляду, аналізу, тощо), він вивільняється (видаляється) за допомогою методу Dispose, щоб вивільнити ресурси системи тестування.

У методі SubjectChart спочатку викликається метод InitializePictureBox, який ініціалізує новий елемент PictureBox. Після цього створюється новий об'єкт Chart, який буде відображати графік.

Цьому об'єкту надається розмір такий же самий, як і у PictureBox. Далі додається область «Points» для графіка. Потім створюється серія даних для вибраного предмету (дисципліни, курсу).

Для кожного користувача, крім адміністратора, обчислюється загальна кількість балів, отриманих студентом (чи максимально можливих при правильному виконанні завдань тесту) за тести за вибраним предметом (дисципліною, курсом), і ці дані додаються до серії.

Налаштування відображення графіка такі ж самі, як і у попередньому методі. Зображення з діаграми створюється у форматі Bitmap і встановлюється у PictureBox. Після використання графіка його ресурси звільняються.

Метод CreateResultTable створює об'єкт DataGridView [49] і налаштовує його для відображення таблиці результатів. Він додає цей об'єкт DataGridView на форму і налаштовує його так, щоб він максимально займав всю доступну (вільну) площу форми і автоматично підлаштовував розмір стовпців таблиці.

Потім він додає стовпці таблиці для кожного користувача і кожного тесту. Додається також стовець для сумарної кількості балів. Для кожного користувача, крім адміністратора, обчислюється сумарна кількість балів за кожен виконаний (пройдений) тест та загальна кількість балів.

Результати щодо тестування заповнюються у відповідні комірки таблиці. Налаштування таблиці включають приховання заголовків рядків, вирівнювання заголовків стовбців по центру, заборону додавання та видалення рядків, і встановлення таблиці як «тільки для читання».

Метод CreateSubjectResultTable:

- створює об'єкт DataGridView;
- налаштовує об'єкт DataGridView для відображення таблиці результатів за предметами (дисциплінами, курсами, темами, розділами курсу тощо);
- додає об'єкт DataGridView на форму;
- налаштовує об'єкт DataGridView так, щоб він займав всю доступну (вільну) площу форми;
- автоматично підлаштовує розмір стовпців таблиці.

Потім метод CreateSubjectResultTable отримує всі унікальні предмети (дисципліни, курси, теми, розділи курсу тощо) з об'єктів класу Test і додає стовпці для користувачів і для кожного унікального предмета (дисципліни, курсу, тощо).

Додаються рядки для кожного користувача (за винятком адміністратора системи тестування) і розраховується відсоток набраних балів від максимально можливої кількості балів за кожен предмет.

Результати (дані щодо тестування) заповнюються у відповідні комірки таблиці. Налаштування таблиці включають:

- приховування заголовків рядків;
- вирівнювання заголовків стовбців по центру;
- заборону додавання та видалення рядків;
- встановлення таблиці в режим «тільки для читання».

3.2 Інструкція користувача системи тестування

3.2.1 Інструкція студента

Система тестування зустрічає користувача (студента) відповідним вікном для авторизації, в якому вказується прізвище, ім'я та пароль студента.

Якщо авторизація пройдена успішно, то система тестування відреагує так, як показано на рис. 3.1.

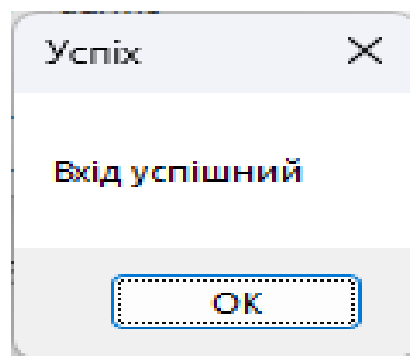


Рисунок 3.1 – Успішна авторизація користувача системи тестування

Після успішної авторизації користувач перенаправляється у вікно інтелектуального діалогу, де йому надається можливість налаштувати під себе розмір шрифту (рис. 3.2) та колір фону (рис. 3.3).

Зміни в інтерфейсі будуть збережені саме для цього користувача (студента).

В подальшому, працюючи з системою тестування, студент буде отримувати саме той колір фону та розмір шрифту, яким буде надаватися текст питань тесту, що він налаштував для себе.

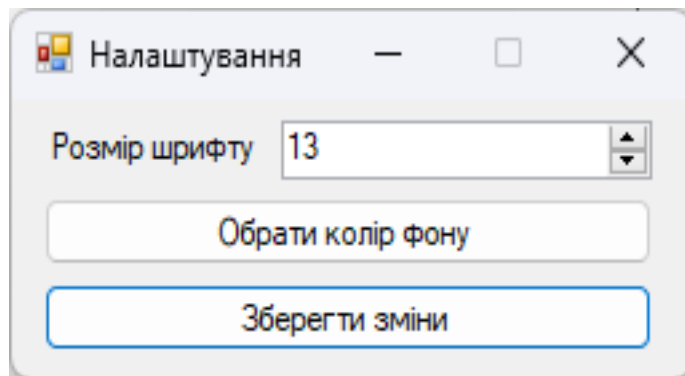


Рисунок 3.2 – Налаштування інтерфейсу в системі тестування

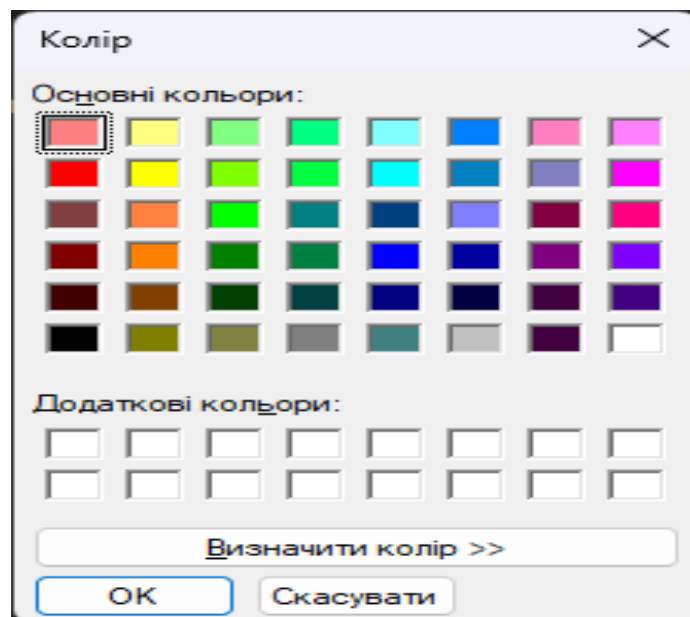


Рисунок 3.3 – Вибір кольору фону тесту(з палітри кольорів) при налаштуванні системи тестування

Після налаштувань інтерфейсу користувач перенаправляється на форму самого тестування (рис. 3.4).

Тут йому доступні на вибір тести з усіх дисциплін різного рівня складності.

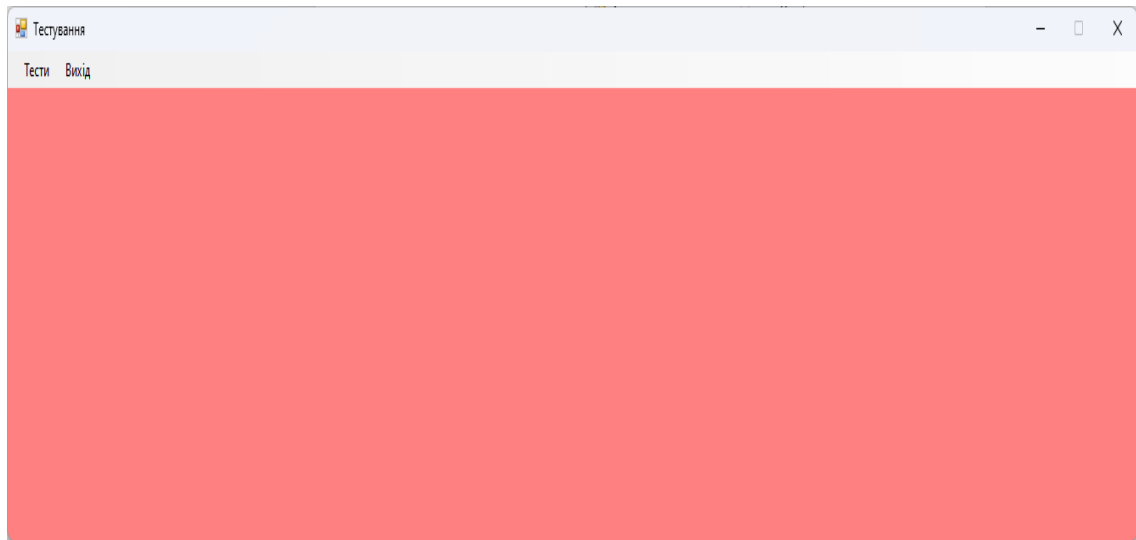


Рисунок 3.4 – Головне меню системи тестування

Для того, щоб обрати тест необхідно обрати вкладинку «Тести» у меню системи тестування (рис. 3.5).

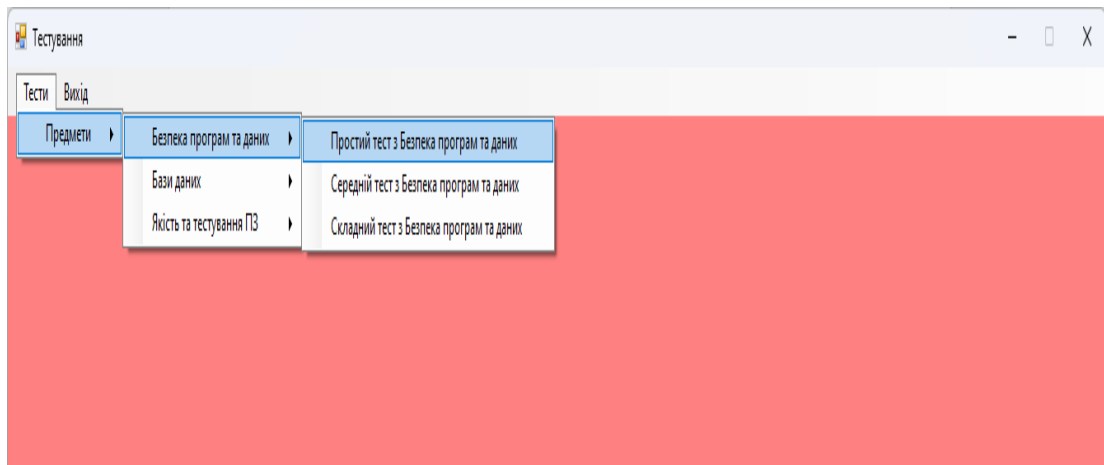


Рисунок 3.5 – Вкладка «Тести»

Студент може здійснити вибір тесту (рис. 3.6). В наведеному прикладі студент обирає легкий тест з дисципліни «Бази даних».

Для цього йому необхідно знайти дисципліну «Бази даних» у вкладці «Тести» головного меню та обрати рівень складності тесту.

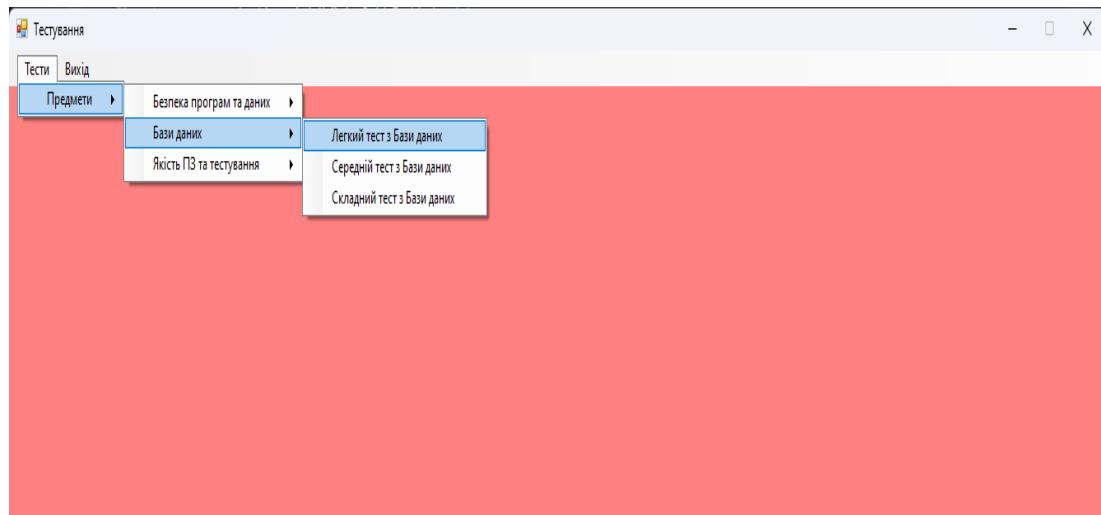


Рисунок 3.6 – Вибір тесту та його рівня

Після обрання тесту на екрані, перед студентом з'являються питання цього тесту. Для переходу до наступного питання користувач (студент) повинен вибрати один з варіантів відповідей та натиснути кнопку «Наступне питання» (рис. 3.7).

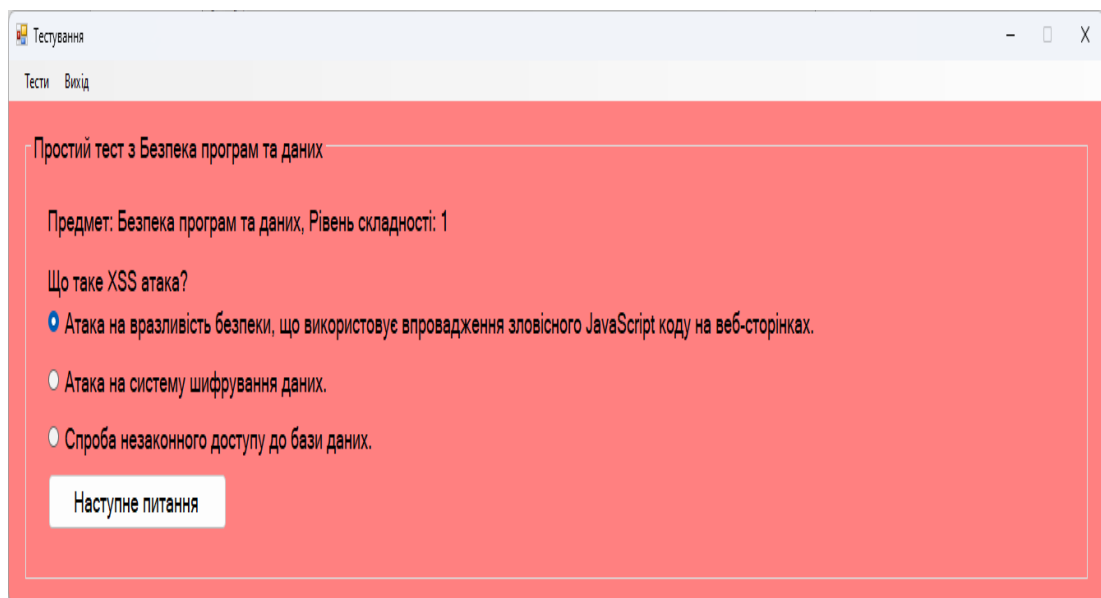


Рис. 3.7. Процес проходження простого тесту з дисципліни
«Безпека програм та даних»

Відповівши на всі питання тесту, студент отримує (йому надається) повідомлення з результатами тестування (рис. 3.8).

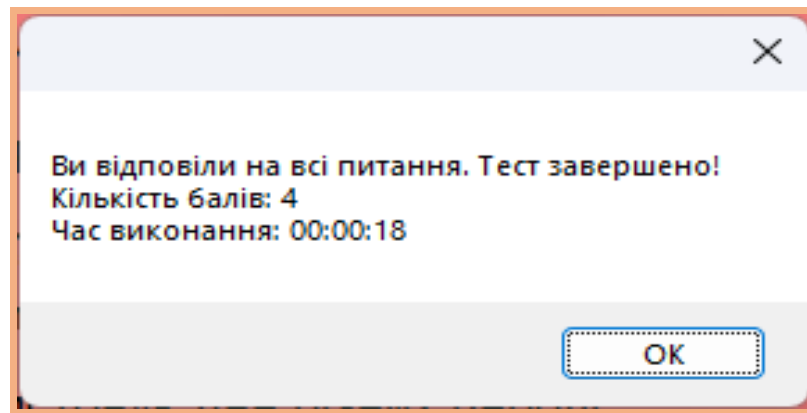


Рисунок 3.8 – Повідомлення студенту про результат тестування з дисципліни «Безпека програм та даних»

Після отримання повідомлення про результат проходження студентом відповідного тестування, форма з надання тестів повертається в початковий стан та система надає користувачеві (студентові) можливість обрати інші тести з тієї ж теми чи іншого предмету (дисципліни).

Обравши тест з дисципліни «Бази даних», студент переходить до виконання тесту. На цьому кроці студент може ознайомитись з питанням та обрати одну з запропонованих відповідей.

Зробивши свій вибір варіанту відповіді, студенту потрібно натиснути на кнопку «Наступне питання» і повторювати ці дії до завершення всіх питань тесту з обраної дисципліни «Бази даних» (рис. 3.9).

Про завершення тестування студент дізнається з відповідного повідомлення, яку надається у вікні системи тестування (рис. 3.10), на якому зображено:

- кількість набраних балів;
- час виконання тесту.

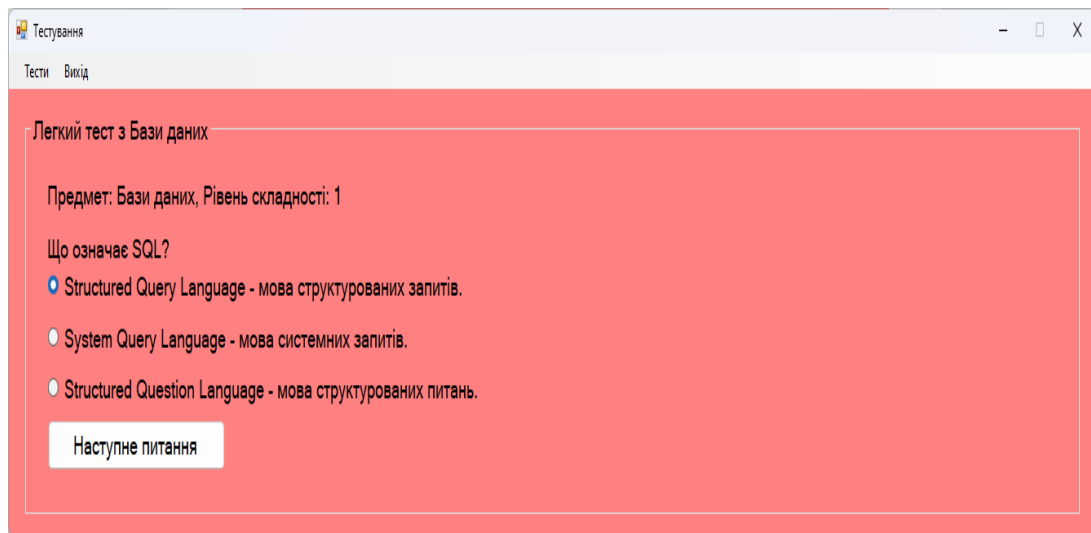


Рисунок 3.9 – Процес проходження тесту з дисципліни «Бази даних»

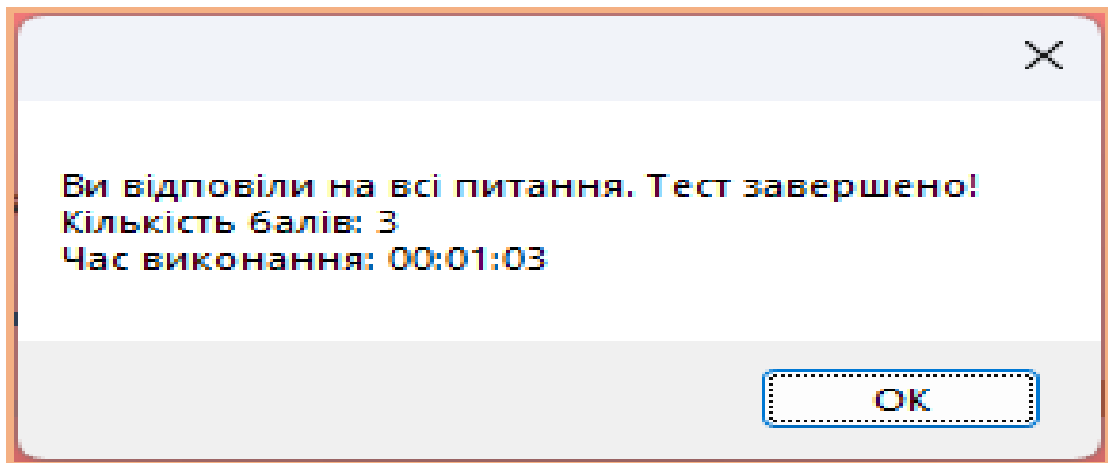


Рисунок 3.10 – Результат тестування студента з дисципліни «База даних»

3.2.2 Інструкція викладача

Авторизація в системі користувача з ім'ям «admin» надає йому доступ до панелі викладача (рис. 3.11).

Панель викладача складається наступних блоків (компонентів, елементів відповідної форми системи тестування):

- з редактору тестів,
- редактору питань,
- візуалізації результатів.

Розглянемо ці компоненти більш детально:

1. *Редактор тестів.* Для того щоб створити новий тест необхідно заповнити всі поля («Назва», «Предмет», «Складність») в редакторі тестів і натиснути кнопку «Створити новий тест».

Для видалення тесту необхідно обрати тест зі списку тестів та натиснути на кнопку «Видалити тест».

2. *Редактор питань.* Обравши зі списку тест та питання в редакторі питань автоматично заповнюються поля відповідно до обраного питання (рис. 3.12).

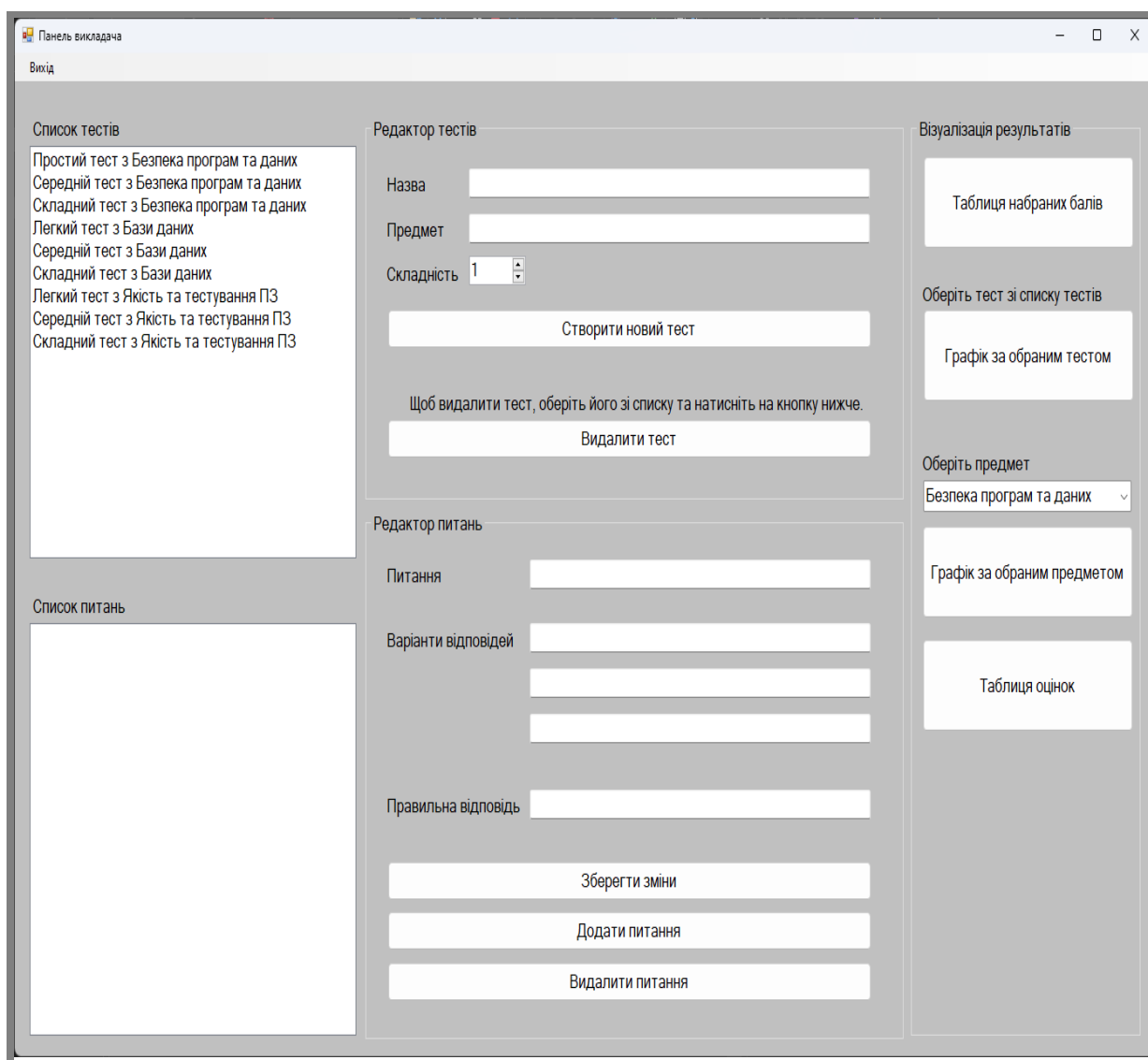


Рисунок 3.11 – Панель викладача в системі тестування

Викладач може змінити будь-яке з полів і натиснути на кнопку «Зберегти зміни».

Після цього питання буде збережено з новими параметрами.

За схожим принципом працює і додавання питання.

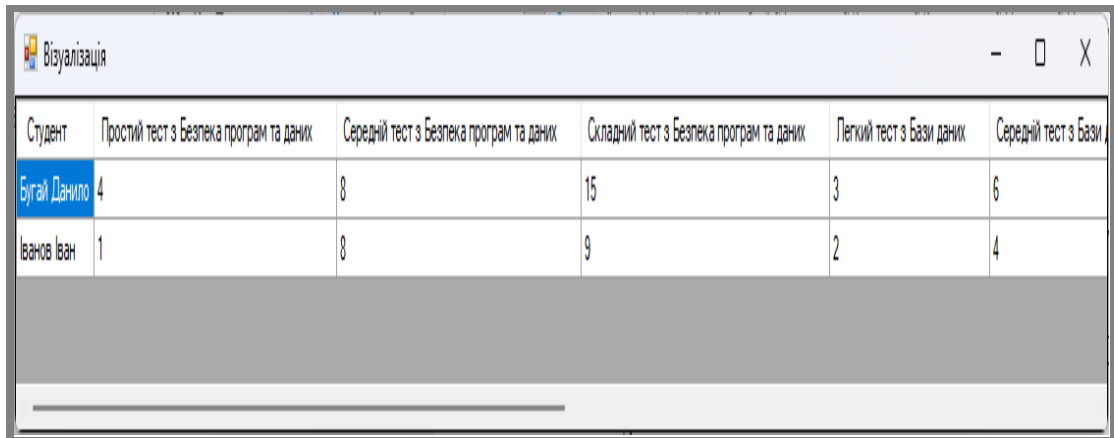
Рисунок 3.12 – Процес редагування питань

Натиснувши кнопку «Додати питання» в обраному тесті буде створено питання з попередньо введеними викладачем параметрами.

Для видалення питання потрібно обрати питання зі списку та натиснути на кнопку «Видалити питання».

3. *Візуалізація результатів.* Після натискання на кнопку «Таблиця набраних балів» відкривається таблиця.

В цій таблиці відображається список студентів та отримані ними бали по всіх тестах зі всіх предметів (рис. 3.13).



Студент	Простий тест з Безпека програм та даних	Середній тест з Безпека програм та даних	Складний тест з Безпека програм та даних	Легкий тест з Бази даних	Середній тест з Бази даних
Бугай Данило	4	8	15	3	6
Іванов Іван	1	8	9	2	4

Рисунок 3.13 – Таблиця набраних студентами балів після проходження тестування

Також в кінці таблиці відображається сумарна кількість балів набраних студентом з усіх тестів.

Обравши тест зі списку і натиснувши на кнопку «Графік за обраним тестом» з'являється графік по обраному простому тесту з дисципліни (предмету) «Безпека програм та даних» (рис. 3.14).

Вісь X відображає студентів групи, а вісь Y – кількість набраних ними балів з обраного простого тесту.

Обравши предмет зі списку предметів, тестування з яких забезпечує розроблена система тестування, і натиснувши на кнопку «Графік за обраним предметом», можна отримати графік за обраним предметом «Безпека програм та даних» для ситуації, коли питання тесту мають різні коефіцієнти складності (рис. 3.15).

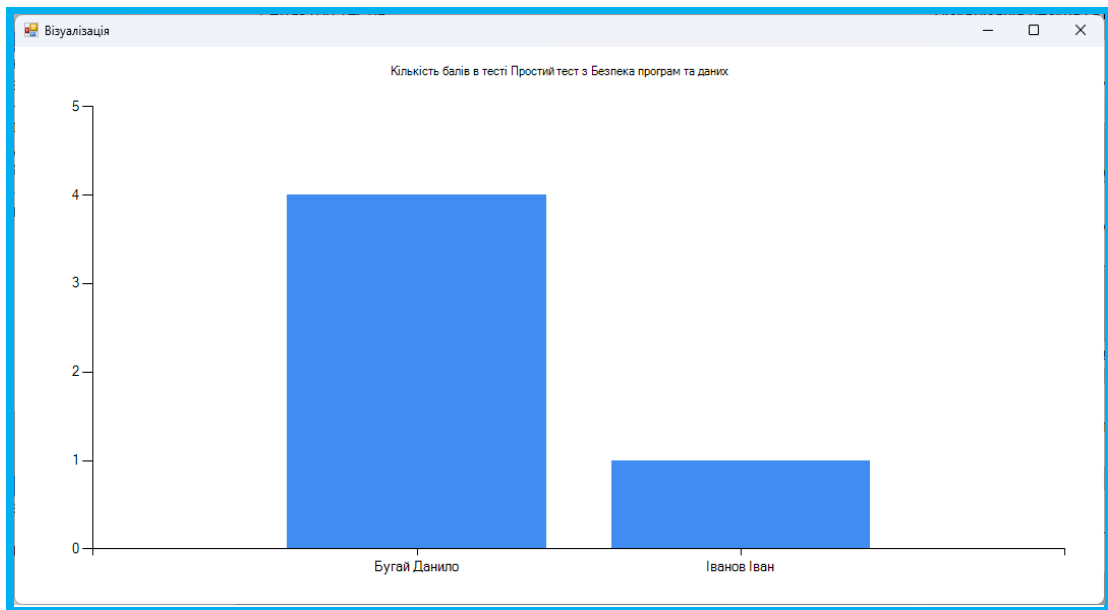


Рисунок 3.14 – Графік результатів тестування студентів групи за обраним тестом з дисципліни «Безпека програм та даних»

Вісь X відображає студентів групи, а вісь Y – кількість набраних ними балів з обраного не простого тесту, тесту, коли його питання мають різні коефіцієнти складності.

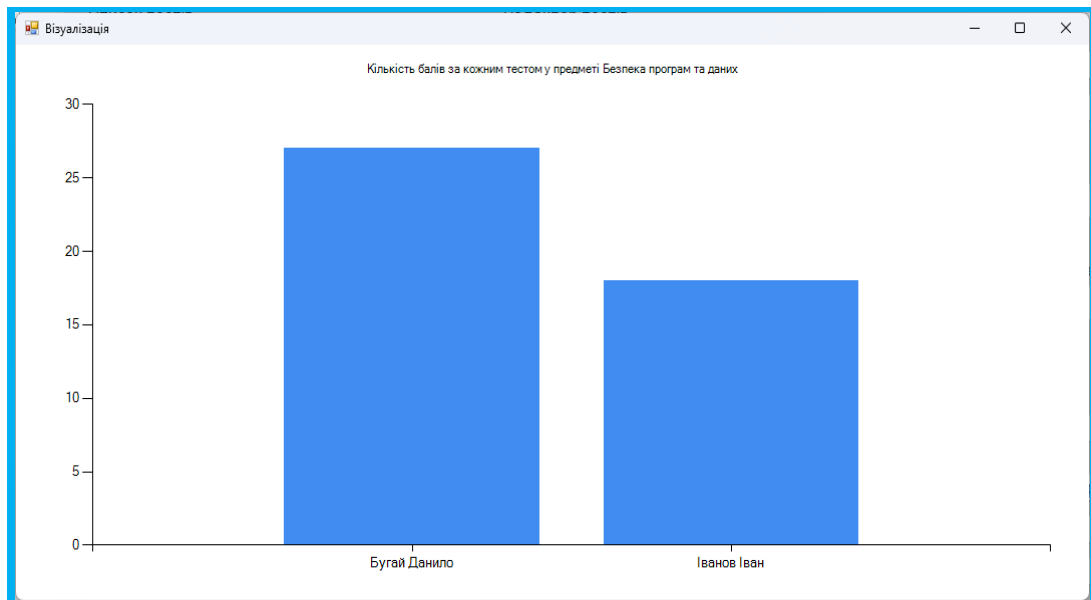
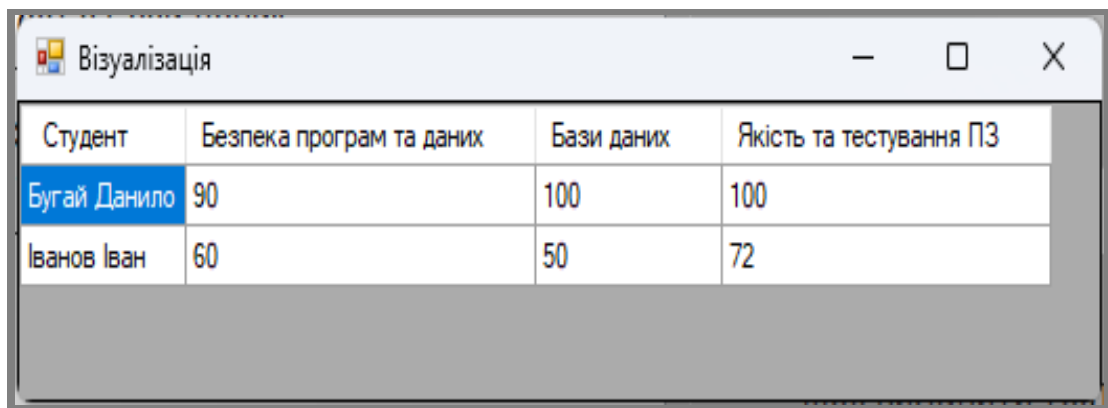


Рисунок 3.15 – Графік результатів тестування студентів групи за обраним (не простим) тестом з дисципліни «Безпека програм та даних»

Після натискання на кнопку «Таблиця оцінок» відкривається таблиця, в якій відображається список студентів групи та оцінки, що отримані ними по всім обраним предметам «Безпека програм та даних», «Бази даних», «Якість та тестування програмного забезпечення» (рис. 3.16).

Оцінка рахується на основі кількості набраних балів з усіх тестів обраного студентом предмету.



Студент	Безпека програм та даних	Бази даних	Якість та тестування ПЗ
Бугай Данило	90	100	100
Іванов Іван	60	50	72

Рис. 3.16. Таблиця оцінок студентів з проведеного тестування за трьома дисциплінами «Безпека програм та даних», «Бази даних», «Якість та тестування програмного забезпечення»

Для оцінки роботи розробленого програмного продукту продемонструємо ще приклад роботи викладача в системі тестування.

Викладачеві має бути надано можливість виконання наступних функцій:

- видалення тесту;
- створення тесту;
- редагування питань;
- перевірка результатів студентів.

Перш за все викладу необхідно авторизуватися/zareєструватися (рис. 3.17). Авторизація (реєстрація, аутентифікація) викладача відбувається за допомогою спеціального облікового запису, коли вказується ім'я користувача admin.

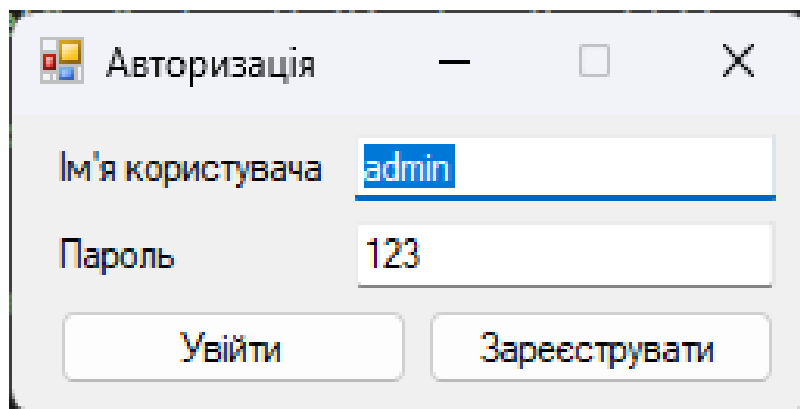


Рисунок 3.17 – Авторизація викладача в системі тестування

Для перевірки функції створення тесту викладач додає новий легкий тест з дисципліни «Бази даних» в систему тестування.

Далі викладач заповнює поля редактору тестів наступним чином:

- Назва – **Легкий тест з Баз даних (2)**.
- Предмет – **Бази даних**.
- Складність – **1**.

Потім викладач має натиснути на кнопку «Створити новий тест».

Після цього натискання тест з назвою **Легкий тест з Баз даних (2)** з'являється в списку тестів.

Це дозволить викладачу в подальшому:

- редагувати його питання;
- за необхідності видалити непотрібні /некоректні/неадекватні питання (рис. 3.18).

На даному етапі в тесті ще немає питань, тому викладач переходить до редактору питань і заповнює всі необхідні його поля.

На рис. 3.19 зображено приклад створення питання для тесту з назвою **Легкий тест з Баз даних (2)**

Панель викладача

Вихід

Список тестів

- Простий тест з Безпека програм та даних
- Середній тест з Безпека програм та даних
- Складний тест з Безпека програм та даних
- Легкий тест з Бази даних
- Середній тест з Бази даних
- Складний тест з Бази даних
- Легкий тест з Якість ПЗ та тестування
- Середній тест з Якість ПЗ та тестування
- Складний тест з Якість ПЗ та тестування
- Легкий тест з Бази даних (2)**

Редактор тестів

Назва: Легкий тест з Бази даних (2)

Предмет: Бази даних

Складність: 1

Створити новий тест

Рисунок 3.18 – Створення тесту з назвою **Легкий тест з Бази даних (2)**

Редактор питань

Питання: Скільки агрегатних функцій в SQL

Варіанти відповідей: 5, 6, 7

Правильна відповідь: 7

Зберегти зміни

Додати питання

Видалити питання

Список питань

- Скільки агрегатних функцій в SQL**

Рисунок 3.19 – Створення питання для тесту з назвою **Легкий тест з Бази даних (2)**

Після того, як викладач заповнить всі поля і натисне на кнопку «Додати питання», нове питання з'явиться в списку питань тесту.

Це дозволить викладачеві в подальшому його редагувати або за необхідності видалити.

Для перевірки функції перегляду результатів тестування студентів у вкладинці «Візуалізація результатів» викладач, натиснувши на кнопку «Таблиця набраних балів», отримає таблицю з результатами всіх тестувань студентів групи (як попередніх, так і з нового тесту **Легкий тест з Баз даних (2)**) (рис. 3.20).

Візуалізація					
Студент	Простий тест з Безпека програм та даних	Середній тест з Безпека програм та даних	Складний тест з Безпека програм та даних	Легкий тест з Баз даних	
Бугай Данило	4	8	15	3	6
Іванов Іван	1	8	9	2	4

Рисунок 3.20 – Таблиця набраних балів студентами за результатами проходження тестування з обраних предметів

ВИСНОВКИ

Суть проблеми, що вирішується в даній магістерській роботі, полягає у розробці програмної системи, яка б забезпечувала можливість проведення тестувань, а також зберігала результати тестувань для подальшого аналізу та використання в освітньому процесі.

В магістерській роботі:

- проаналізовано предметну область – тестування знань студентів, визначено види тестів, методи та підходи до тестування студентів;
- розглянуто сучасні поширені інформаційні системи та платформи, які підтримують комп'ютерне тестування знань студентів, визначено їх переваги та недоліки;
- розроблено онтологічну модель системи тестування знань студентів.

У магістерській роботі:

- проведено аналіз вимог до системи тестування знань студентів;
- розроблено архітектуру та інтерфейс користувача системи тестування;
- реалізовано необхідний функціонал системи тестування;
- розроблено систему тестування знань студентів;
- протестовано систему тестування знань студентів.

Розроблена система тестування:

- сприяє підвищенню ефективності контролю знань студентів;
- дозволяє викладачам швидко та об'єктивно оцінювати рівень знань студентів та розуміння ними навчального матеріалу (навчального контенту);
- спрощує процес підготовки тестів та проведення тестування;
- зменшує час, необхідний для оцінювання робіт студентів;
- підвищує доступність інформації про досягнення кожного студента;
- має зручний і комфортний інтерфейс.

Результати магістерської роботи були:

- представлені на VI Міжнародної наукової конференції «Актуальні питання розвитку галузей», яка відбулась у Вінниці 31 жовтня 2025 року,
- опубліковані в науковій статті у фаховому журналі «Наука і техніка сьогодні», 2025. № 10(51)

Розроблена система тестування може бути використана при організації перевірки знань студентів, що навчаються за спеціальністю «Інженерія програмного забезпечення».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Франчук В.М., Франчук Н.П. Організація електронного навчального курсу для підтримки аудиторного навчального процесу URL: <http://2017.moodlemoot.in.ua/course/view.php?id=119>
2. Ковальчук А. Тестові технології оцінювання якості. URL: <http://osvita.ua/school/theory/5919>.
3. Рудюк Т.Я. Методичні підходи до створення валідних тестових завдань з природничих наук. *Тестування і моніторинг в освіті*, 2007. № 12. С. 34-36.
4. Frary R.B. Formula Scoring of Multiple-Choice Tests (Correction for Guessing). *Educational measurement: Issues and Practice*. 2008. 7 (2). P. 33-38.
5. Tecla L., Eggen T. Innovative Item Types in Computer Based Testing: Scoring of Multiple Response. Citogroep. Netherland. 2003. 9.
6. Moodle.org: open-source community-based tools for learning [Electronic Resource]. URL: <http://moodle.org>.
7. ILIAS E-Learning [Electronic Resource]. URL: <http://www.ilias.de>.
8. Test creation and e-testing software – SunRav TestOfficePro. URL: <https://www.sunrav.com/testofficepro.html>
9. WebQuiz – твій квіз онлайн. URL: <https://webquiz.net>
10. Want to learn how to create videos that connect, captivate and get results? URL: <https://colabcreation.co.uk/learning-lab/course-lab>
11. Learning Design Supercharged. URL: <https://www.adobe.com/products/captivate.html>
12. MyTest X. URL https://teach-inf.com.ua/load/programi/testi/mytest_x/16-1-0-35
13. AD Tester. URL: <https://www.adtester.org>
14. Google Forms. Google Tools Meets Middle School. 2455 Teller Road, Thousand Oaks California 91320, 2018. С. 127–142. URL: <https://doi.org/10.4135/9781506360188.n7>.

15. Google Forms: what is it and how to create it? URL: <https://wedex.com.ua/en/blog/google-forms-what-is-it-and-how-to-create-it>
16. Kahoot!. 50 Ways to Use Technology Enhanced Learning in the Classroom: Practical strategies for teaching. 1 Oliver's Yard, 55 City Road London EC1Y 1SP, 2018. С. 206–210. URL: <https://doi.org/10.4135/9781529793550.n40>
17. Kahoot! URL: <https://kahoot.it>
18. Professional Development for Educators. URL: <https://create.kahoot.it/profiles/d3865a1d-9745-462f-a895-6022ed3020b1>
19. Quizizz is now wayground. URL: <https://wayground.com/?lng=en>
20. Learning Apps. URL: <https://learningapps.org>
21. Морзе Н.В., Глазунова О.Г., Мокрієв М.В. Методика створення електронного навчального курсу (на базі платформи дистанційного навчання Moodle 3): навчальний посібник. Київ: 2016. 240 с.
22. Moodle LMS. URL: <https://moodle.org/course>
23. Download Moodle LMS. URL: <https://download.moodle.org>
24. Rice W., Nash S. S. Moodle E-Learning Course Development. Packt Publishing, Limited, 2022.
25. About ATutor. URL: <https://dl.tntu.edu.ua/about.php>
26. JustClass. URL: <https://justclass.com.ua/blog/6-platform-dlya-stvorenniya-testiv-na-urok>
27. EasyTestMaker. URL: <https://www.easytestmaker.com>
28. Адаптивні технології: досвід Knewton. URL: <http://blog.ed-era.com/knewton/>
29. Ткаченко К.О., Ісаченко Б.О. Використання гейміфікації в інтелектуальних навчальних системах та їх вебдизайні. *Водний транспорт*, No 2(38) 2023. С. 299-311. doi.org/10.33298/2226-8553.2023.2.38.32
30. Huajin Tang, Kay Chen Tan, Zhang Yi. Neural Networks: Computational Models and Applications. Springer Science & Business Media, 2007. 300 p.
31. Crina Grosan, Ajith Abraham. Intelligent Systems: A Modern Approach. Berlin: Springer-Verlag Berlin Heidelberg, 2011. 450 p.

32. What is data mining? URL: <https://www.ibm.com/think/topics/data-mining>
33. Twin A., Yashina N. What Is Data Mining? How It Works, Benefits, Techniques, and Examples. URL: <https://www.investopedia.com/terms/d/datamining.asp>
34. Удовик І.М., Коротенко Г.М., Коротенко Л.М., Трусів В.О., Харь А.Т. Методи та системи штучного інтелекту: навч. посіб. Дніпро: Державний ВНЗ «Національний гірничий університет», 2017. 105 с.
35. Sumi Helal, Wendong Li, Raja Bose. Mobile Platforms and Development Environments (Synthesis Lectures on Mobile and Pervasive Computing) Morgan & Claypool Publishers, 2012. 120 p.
36. Timothy J. Ross. Fuzzy Logic with Engineering Applications New Mexico : Wiley, 2010. 606 p.
37. Басюк Л.В., Лисенко Т.А. Носач І.В. Аналіз ефективності методів штучного інтелекту для обробки та інтерпретації даних в освітніх дослідженнях. *Педагогічна академія: Інформаційно-комунікаційні технології в освіті*, 2024. <https://doi.org/10.5281/zenodo.13327035>
38. Voskoglou M. Fuzzy Sets, Fuzzy Logic and Their Applications. 2020. 366 p.
39. Mohan C. An introduction to fuzzy set theory and fuzzy logic. 2019. 392 p.
40. Carter J., Chiclana F., Singh Khuman A., Chen T. Fuzzy Logic: Recent Applications and Developments. 2021. 385 p.
41. Jacquette D. Ontology. Routledge, 2014. URL: <https://doi.org/10.4324/9781315710655>.
42. What Is an Ontology?. Building Ontologies with Basic Formal Ontology. 2015. URL: <https://doi.org/10.7551/mitpress/8743.003.0004>).
43. Open-source ontology editor Protégé. URL: <https://protege.stanford.edu/>
44. Sharp D. Mastering C# Intermediate : Classes, Interfaces and OOP: The Ultimate, Step by Step Guide to Classes, Interfaces and Object Oriented Programming with C# Intermediate C# Programming Level. Independently Published, 2019.

45. Gibbons P. Using WinForms. .NET Development for Java Programmers. Berkeley, CA, 2002. С. 67–95. URL: https://doi.org/10.1007/978-1-4302-1106-8_4.
46. Луценко В. М. Комплексні системи захисту інформації довільної складності. *Ukrainian Information Security Research Journal*. 2012. Т. 14, № 2 (55). URL: <https://doi.org/10.18372/2410-7840.14.2176>
47. Fattah H. Data Serialization and Deserialization. LTE™ Cellular Narrowband Internet of Things (NB-IoT). 2021. С. 69–86. URL: <https://doi.org/10.1201/9781003120018-6>.
48. Chart Class URL: <https://learn.microsoft.com/en-gb/dotnet/api/system.windows.forms.datavisualization.charting.chart?view=netframework-4.8.1>
49. DataGridView Class URL: <https://learn.microsoft.com/en-us/dotnet/api/system.windows.forms.datagridview?view=windowsdesktop-8.0>
50. Ткаченко К.О., Носов І.С., Гостіщев А.С. Розроблення інформаційних систем управління та організації навчального процесу: онтологічний підхід: збірник наукових праць з матеріалами VI Міжнародної наукової конференції *Актуальні питання розвитку галузей* (м. Вінниця, 31 жовтня, 2025). Вінниця: ТОВ «УКРЛОГОС Груп, 2025. С.428 – 432.
51. Ткаченко К.О., Ткаченко О.А., Носов І.С., Гостіщев А.С. Розробка інформаційних систем тестування знань студентів: онтологічний підхід. *Наука і техніка сьогодні*, 2025. № 10(51). С. 2039-2052. [https://doi.org/10.52058/2786-6025-2025-10\(51\)-2039-2052](https://doi.org/10.52058/2786-6025-2025-10(51)-2039-2052)

ДОДАТКИ

ДОДАТОК А. Програмний код

SettingsForm:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace StudentTest
{
    public partial class SettingsForm : Form
    {
        public SettingsForm()
        {
            InitializeComponent();
            fontSizeNumericUpDown.Value =
DataBank.currentUser.FontSize;
        }
        private void saveButton_Click(object sender, EventArgs
e)
        {
            DataBank.currentUser.FontSize =
(int)fontSizeNumericUpDown.Value;

DataBank.userManager.UpdateUserSettings(DataBank.currentUser);
            DialogResult = DialogResult.OK;
            Close();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            if (colorDialog1.ShowDialog() == DialogResult.OK)
            {
                DataBank.currentUser.BackgroundColor =
colorDialog1.Color;
            }
        }
    }
}

```

LoginForm:

```

using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Windows.Forms;
namespace StudentTest
{
    public partial class LoginForm : Form
    {
        public LoginForm()
        {
            InitializeComponent();
            DataBank.userManager = new UserManager();
            DataBank.testManager = new TestManager();
            DataBank.tests = DataBank.testManager.LoadTests();
        }
        public class TestResult
        {
            public string TestName { get; set; }
            public int Points { get; set; }
            public TimeSpan Duration { get; set; }
            public TestResult(string testName, int points,
TimeSpan duration)
            {
                TestName = testName;
                Points = points;
                Duration = duration;
            }
        }
        [Serializable]
        public class User
        {
            public string Username { get; set; }
            public string Password { get; set; }
            public int FontSize { get; set; }
            public Color BackgroundColor { get; set; }
            public List<TestResult> TestResults { get; set; }
            public User(string username, string password)
            {
                Username = username;
                Password = password;
                FontSize = 12;
                BackgroundColor = Color.White;
            }
        }
    }
}

```

```

        TestResults = new List<TestResult>();
    }
}
public class UserManager
{
    public List<User> users;
    private string filePath = "users.json";
    public UserManager()
    {
        users = DeserializeUsers();
    }
    private List<User> DeserializeUsers()
    {
        if (!File.Exists(filePath))
        {
            return new List<User>();
        }
        try
        {
            string json = File.ReadAllText(filePath);
            return
JsonConvert.DeserializeObject<List<User>>(json);
        }
        catch (Exception e)
        {
            MessageBox.Show("Помилка завантаження
користувачів: " + e.Message, "Помилка");
            return new List<User>();
        }
    }
    private void SerializeUsers()
    {
        try
        {
            string json =
JsonConvert.SerializeObject(users, Formatting.Indented);
            File.WriteAllText(filePath, json);
        }
        catch (Exception e)
        {
            MessageBox.Show("Помилка збереження
користувачів: " + e.Message, "Помилка");
        }
    }
    public void AddUser(string username, string
password)

```

```

        {
            users.Add(new User(username, password));
            SerializeUsers();
        }
        public bool AuthenticateUser(string username, string
password)
        {
            return users.Any(u => u.Username == username &&
u.Password == password);
        }
        public User ReturnUser(string username, string
password)
        {
            return users.FirstOrDefault(u => u.Username ==
username && u.Password == password);
        }
        public void UpdateUserSettings(User user)
        {
            User existingUser = users.FirstOrDefault(u =>
u.Username == user.Username);
            if (existingUser != null)
            {
                existingUser.FontSize = user.FontSize;
                existingUser.BackgroundColor =
user.BackgroundColor;
                existingUser.TestResults = user.TestResults;
                SerializeUsers();
            }
        }
    }
    private void loginButton_Click(object sender, EventArgs
e)
    {
        string username = usernameTextBox.Text;
        string password = passwordTextBox.Text;
        if (DataBank.userManager.AuthenticateUser(username,
password))
        {
            DataBank.currentUser =
DataBank.userManager.ReturnUser(username, password);
            MessageBox.Show("Вхід успішний", "Успіх");
            DialogResult = DialogResult.OK;
        }
        else
        {

```

```

        MessageBox.Show("Неправильне ім'я користувача
або пароль", "Помилка входу");
    }
}
private void registerButton_Click(object sender,
EventArgs e)
{
    string username = usernameTextBox.Text;
    string password = passwordTextBox.Text;
    DataBank.userManager.AddUser(username, password);
    MessageBox.Show("Користувача успішно додано",
"Успіх");
}
}
}

```

QuestionForm:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;
using static StudentTest.LoginForm;
namespace StudentTest
{
    public partial class QuestionForm : Form
    {
        public QuestionForm()
        {
            InitializeComponent();
            InitializeTestListBox();
            PopulateSubjectsComboBox();
            subjectComboBox.SelectedIndex = 0;
            ToolStripMenuItem exitItem = new
ToolStripMenuItem("Вихід");
            exitItem.Click += exitToolStripMenuItem_Click;
            this.MainMenuStrip.Items.Add(exitItem);
            this.Font = new Font(this.Font.FontFamily,
DataBank.currentUser.FontSize);
            this.BackColor =
DataBank.currentUser.BackgroundColor;
        }
    }
}

```

```

        private void exitToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            System.Windows.Forms.Application.Restart();
            System.Windows.Forms.Application.Exit();
        }
        public void InitializeTestListBox()
        {
            foreach (var test in DataBank.tests)
            {
                testsListBox.Items.Add(test);
            }
        }
        private void addQuestionButton_Click(object sender,
EventArgs e)
        {
            if (testsListBox.SelectedIndex != -1)
            {
                Test selectedTest =
                (Test)testsListBox.SelectedItem;
                // Створюємо нове питання з тексту, який ввів
                користувач
                Question newQuestion = new Question
                {
                    Name = questionTextBox.Text,
                    Answers = new List<string>
                    {
                        firstAnswerTextBox.Text,
                        secondAnswerTextBox.Text,
                        thirdAnswerTextBox.Text
                    },
                    CorrectAnswer = correctAnswerTextBox.Text
                };
                // Додаємо питання до вибраного тесту
                selectedTest.Questions.Add(newQuestion);
                questionsListBox.Items.Add(newQuestion);
                // Очищаємо поля для введення питання
                questionTextBox.Clear();
                firstAnswerTextBox.Clear();
                secondAnswerTextBox.Clear();
                thirdAnswerTextBox.Clear();
                correctAnswerTextBox.Clear();
            }
            // Зберегти тести
            DataBank.testManager.SaveTests(DataBank.tests);
        }
    }

```

```

        private void saveQuestionButton_Click(object sender,
EventArgs e)
        {
            Test selectedTest = (Test)testsListBox.SelectedItem;
            if (selectedTest != null)
            {
                Question selectedQuestion =
                (Question)questionsListBox.SelectedItem;
                if (selectedQuestion != null)
                {
                    // Змінюємо дані у вибраному питанні
                    selectedQuestion.Name =
questionTextBox.Text;
                    selectedQuestion.Answers[0] =
firstAnswerTextBox.Text;
                    selectedQuestion.Answers[1] =
secondAnswerTextBox.Text;
                    selectedQuestion.Answers[2] =
thirdAnswerTextBox.Text;
                    selectedQuestion.CorrectAnswer =
correctAnswerTextBox.Text;
                    // Оновлюємо відображення у списку тестів
                    int testIndex = testsListBox.SelectedIndex;
                    testsListBox.ClearSelected();
                    testsListBox.SelectedIndex = testIndex;
                    // Оновлюємо відображення у списку питань
                    int selectedIndex =
questionsListBox.SelectedIndex;
                    questionsListBox.ClearSelected();
                    questionsListBox.SelectedIndex =
selectedIndex;
                }
            }
            // Зберегти тести
            DataBank.testManager.SaveTests(DataBank.tests);
        }
        private void deleteQuestionButton_Click(object sender,
EventArgs e)
        {
            // Видаляємо питання з відповідного тесту
            Test selectedTest = (Test)testsListBox.SelectedItem;
            Question selectedQuestion =
            (Question)questionsListBox.SelectedItem;
            selectedTest.Questions.Remove(selectedQuestion);
            // Видаляємо виділене питання з лістбокса
            questionsListBox.Items.Remove(selectedQuestion);
        }
    }
}

```

```

        // Зберегти тести
        DataBank.testManager.SaveTests(DataBank.tests);
    }
    private void testsListBox_SelectedIndexChanged(object
sender, EventArgs e)
    {
        questionsListBox.Items.Clear();
        Test selectedTest = (Test)testsListBox.SelectedItem;
        if (selectedTest != null)
        {
            foreach (var question in selectedTest.Questions)
            {
                questionsListBox.Items.Add(question);
            }
        }
    }
    private void
questionsListBox_SelectedIndexChanged(object sender, EventArgs
e)
    {
        Question selectedQuestion =
        (Question)questionsListBox.SelectedItem;
        if (selectedQuestion != null)
        {
            questionTextBox.Text = selectedQuestion.Name;
            firstAnswerTextBox.Text =
selectedQuestion.Answers[0];
            secondAnswerTextBox.Text =
selectedQuestion.Answers[1];
            thirdAnswerTextBox.Text =
selectedQuestion.Answers[2];
            correctAnswerTextBox.Text =
selectedQuestion.CorrectAnswer;
        }
    }
    private void deleteTestButton_Click(object sender,
EventArgs e)
    {
        if (testsListBox.SelectedIndex != -1)
        {
            // Видаляємо тест з лістбокса
            Test selectedTest =
            (Test)testsListBox.SelectedItem;
            DataBank.tests.Remove(selectedTest);
            testsListBox.Items.Remove(selectedTest);
            // Очищуємо лістбокс з питаннями

```

```

        questionsListBox.Items.Clear();
        // Зберегти тести
        DataBank.testManager.SaveTests(DataBank.tests);
    }
}
private void newTestButton_Click(object sender,
EventArgs e)
{
    string subject = subjectNewTestTextbox.Text;
    string name = nameNewTestTextBox.Text;
    int difficulty = (int)difficultyNumericUpDown.Value;
    if (string.IsNullOrEmpty(subject) ||
string.IsNullOrEmpty(name))
    {
        MessageBox.Show("Будь ласка, введіть предмет і
назву тесту.");
        return;
    }
    Test newTest = new Test(name, subject, difficulty);
    DataBank.tests.Add(newTest);
    testsListBox.Items.Add(newTest);
    // Зберегти тести
    DataBank.testManager.SaveTests(DataBank.tests);
}
private void button1_Click(object sender, EventArgs e)
{
    DataBank.selectedTest =
(Test)testsListBox.SelectedItem;
    if (DataBank.selectedTest != null)
    {
        ChartForm chartForm = new ChartForm();
        chartForm.DrawPointsChart();
        chartForm.Show();
    }
}
private void PopulateSubjectsComboBox()
{
    subjectComboBox.Items.Clear();
    var subjects = DataBank.tests.Select(t =>
t.Subject).Distinct();
    foreach (var subject in subjects)
    {
        subjectComboBox.Items.Add(subject);
    }
}
private void button2_Click(object sender, EventArgs e)

```

```

        {
            DataBank.selectedSubject = subjectComboBox.Text;
            if (DataBank.selectedSubject != null)
            {
                ChartForm chartForm = new ChartForm();
                chartForm.SubjectChart();
                chartForm.Show();
            }
        }
        private void button3_Click(object sender, EventArgs e)
        {
            ChartForm chartForm = new ChartForm();
            chartForm.CreateResultTable();
            chartForm.Show();
        }
        private void button4_Click(object sender, EventArgs e)
        {
            ChartForm chartForm = new ChartForm();
            chartForm.CreateSubjectResultTable();
            chartForm.Show();
        }
    }
}

```

ChartForm:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;
using static StudentTest.LoginForm;
namespace StudentTest
{
    public partial class ChartForm : Form
    {
        PictureBox pictureBox1;
        public ChartForm()
        {
            InitializeComponent();
        }
        private void InitializePictureBox()
        {

```

```

        pictureBox1 = new PictureBox();
        pictureBox1.Dock = DockStyle.Fill; // Займає всю
доступну площу форми
        pictureBox1.SizeMode = PictureBoxSizeMode.Zoom; //
Масштабує зображення
        Controls.Add(pictureBox1); // Додає PictureBox на
форму
    }
    public void DrawPointsChart()
    {
        InitializePictureBox();
        // Створити новий графік
        Chart chart = new Chart();
        chart.Size = pictureBox1.Size;
        // Додати область для графіка
        ChartArea chartArea = new ChartArea("Points");
        chart.ChartAreas.Add(chartArea);
        // Додати серію для вибраного тесту
        Series series = new
Series(DataBank.selectedTest.Name);
        series.ChartType = SeriesChartType.Column;
        foreach (var user in
DataBank.userManager.users.Where(u => u.Username != "admin"))
        {
            int totalPoints = user.TestResults.Where(tr =>
tr.TestName == DataBank.selectedTest.Name).Sum(tr => tr.Points);
            series.Points.AddXY(user.Username, totalPoints);
        }
        chart.Series.Add(series);
        // Налаштувати відображення графіка
        chartArea.AxisX.MajorGrid.Enabled = false;
        chartArea.AxisY.MajorGrid.Enabled = false;
        chartArea.AxisX.Interval = 1;
        chart.Titles.Add("Кількість балів в тесті " +
DataBank.selectedTest.Name);
        // Створити зображення з діаграми
        Bitmap bmp = new Bitmap(chart.Width, chart.Height);
        chart.DrawToBitmap(bmp, new Rectangle(0, 0,
bmp.Width, bmp.Height));
        // Встановити зображення у PictureBox
        pictureBox1.Image = bmp;
        // Важливо: звільнити ресурси графіка
        chart.Dispose();
    }
    public void SubjectChart()
    {

```

```

InitializePictureBox();
// Створити новий графік
Chart chart = new Chart();
chart.Size = pictureBox1.Size;
// Додати область для графіка
ChartArea chartArea = new ChartArea("Points");
chart.ChartAreas.Add(chartArea);
// Додати точку до серії з ім'ям користувача
Series series = new
Series(DataBank.selectedSubject);
series.ChartType = SeriesChartType.Column;
// Додати серію для кожного користувача за обраним
предметом
foreach (var user in
DataBank.userManager.users.Where(u => u.Username != "admin"))
{
    int totalPoints = user.TestResults
        .Where(tr => DataBank.tests
            .Any(test => test.Subject ==
DataBank.selectedSubject && test.Name == tr.TestName))
        .Sum(tr => tr.Points);
    series.Points.AddXY(user.Username, totalPoints);
}
chart.Series.Add(series);
// Налаштувати відображення графіка
chartArea.AxisX.MajorGrid.Enabled = false;
chartArea.AxisY.MajorGrid.Enabled = false;
chartArea.AxisX.Interval = 1;
// Додати заголовок графіка
chart.Titles.Add("Кількість балів за кожним тестом у
предметі " + DataBank.selectedSubject);
// Створити зображення з діаграми
Bitmap bmp = new Bitmap(chart.Width, chart.Height);
chart.DrawToBitmap(bmp, new Rectangle(0, 0,
bmp.Width, bmp.Height));
// Встановити зображення у PictureBox
pictureBox1.Image = bmp;
// Важливо: звільнити ресурси графіка
chart.Dispose();
}
public void CreateResultTable()
{
    // Створення DataGridView
    DataGridView dataGridView = new DataGridView();
    dataGridView.Dock = DockStyle.Fill; // Займає всю
доступну площу форми

```

```

        dataGridView.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.AllCells; // Автоматично
підлаштовує розмір стовбців
        // Додавання DataGridView на форму
Controls.Add(dataGridView);
        // Додавання стовбців для користувачів
dataGridView.Columns.Add("User", "Студент");
        // Додавання стовбців для тестів
foreach (var test in DataBank.tests)
{
            dataGridView.Columns.Add(test.Name, test.Name);
        }
        // Додавання колонки для сумарної кількості балів
dataGridView.Columns.Add("Total", "Сумарна кількість
балів");
        // Додавання рядків для користувачів та заповнення
значеннями
        foreach (var user in
DataBank.userManager.users.Where(u => u.Username != "admin"))
        {
            // Додаємо новий рядок
            int rowIndex = dataGridView.Rows.Add();
            // Встановлюємо значення для першого стовбця
            (ім'я користувача)
            dataGridView.Rows[rowIndex].Cells[0].Value =
user.Username;
            // Заповнюємо значення для кожного тесту
            foreach (var test in DataBank.tests)
            {
                int totalPoints = user.TestResults
                    .Where(tr => tr.TestName == test.Name)
                    .Sum(tr => tr.Points);

                dataGridView.Rows[rowIndex].Cells[test.Name].Value =
totalPoints;
            }
            // Підрахунок і встановлення сумарної кількості
            балів
            int totalPointsForUser = user.TestResults.Sum(tr
=> tr.Points);
            dataGridView.Rows[rowIndex].Cells["Total"].Value
= totalPointsForUser;
        }
        // Налаштування DataGridView
dataGridView.RowHeadersVisible = false;

```

```

        dataGridView.ColumnHeadersDefaultCellStyle.Alignment
= DataGridViewContentAlignment.MiddleCenter;
        dataGridView.AllowUserToAddRows = false;
        dataGridView.AllowUserToDeleteRows = false;
        dataGridView.ReadOnly = true;
    }
    public void CreateSubjectResultTable()
    {
        // Створення DataGridView
        DataGridView dataGridView = new DataGridView();
        dataGridView.Dock = DockStyle.Fill; // Займає всю
доступну площу форми
        dataGridView.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.AllCells; // Автоматично
підлаштовує розмір стовбців
        // Додавання DataGridView на форму
        Controls.Add(dataGridView);
        // Отримання всіх унікальних предметів з об'єктів
Test з файлу JSON
        var uniqueSubjects = DataBank.tests.Select(t =>
t.Subject).Distinct();
        // Додавання стовпців для користувачів
        dataGridView.Columns.Add("User", "Студент");
        // Додавання стовпців для всіх предметів
        foreach (var subject in uniqueSubjects)
        {
            dataGridView.Columns.Add(subject, subject);
        }
        // Додавання рядків для користувачів та заповнення
значеннями
        foreach (var user in
DataBank.userManager.users.Where(u => u.Username != "admin"))
        {
            // Додаємо новий рядок з ім'ям користувача
            int rowIndex =
dataGridView.Rows.Add(user.Username);
            // Заповнюємо значення для кожного предмету
            foreach (var subject in uniqueSubjects)
            {
                int totalPoints = 0;
                int maxPoints = 0;
                foreach (var test in DataBank.tests.Where(t
=> t.Subject == subject))
                {
                    int testTotalPoints = user.TestResults

```

```

        .Where(tr => tr.TestName ==
test.Name)
        .Sum(tr => tr.Points);
        // Отримання максимально можливої
кількості балів за тест
        int testMaxPoints = test.Difficulty *
test.Questions.Count;
        totalPoints += testTotalPoints;
        maxPoints += testMaxPoints;
    }
    // Розрахунок відсотка набраних балів від
максимально можливої кількості балів за предмет
    double percentage = (double)totalPoints /
maxPoints * 100;
    // Додавання комірки з оцінкою на перетині
стовпця предмету та рядка студента

dataGridView.Rows[rowIndex].Cells[subject].Value
=Math.Round(percentage);
    }
}
// Налаштування DataGridView
dataGridView.RowHeadersVisible = false;
dataGridView.ColumnHeadersDefaultCellStyle.Alignment
= DataGridViewContentAlignment.MiddleCenter;
dataGridView.AllowUserToAddRows = false;
dataGridView.AllowUserToDeleteRows = false;
dataGridView.ReadOnly = true;
    }
}
}

```

Form1:

```

using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Windows.Forms;
using static StudentTest.LoginForm;
using static System.Net.Mime.MediaTypeNames;
namespace StudentTest
{
    public partial class Form1 : Form
    {
        private Test selectedTest;
    }
}

```

```

private int currentQuestionIndex;
public int points;
public DateTime startTime = DateTime.Now;
public Form1()
{
    InitializeComponent();
    InitializeMenuStrip();
    HideAllControls(groupBox1);
    this.Font = new Font(this.Font.FontFamily,
DataBank.currentUser.FontSize);
    this.BackColor =
DataBank.currentUser.BackgroundColor;
}
private void InitializeMenuStrip()
{
    ToolStripMenuItem subjectsMenu = new
ToolStripMenuItem("Предмети");

testsToolStripMenuItem.DropDownItems.Add(subjectsMenu);
    foreach (var subject in DataBank.tests.Select(test
=> test.Subject).Distinct())
    {
        ToolStripMenuItem subjectItem = new
ToolStripMenuItem(subject);
        subjectsMenu.DropDownItems.Add(subjectItem);
        foreach (var test in DataBank.tests.Where(test
=> test.Subject == subject))
        {
            ToolStripMenuItem testItem = new
ToolStripMenuItem(test.Name);
            testItem.Click +=
testToolStripMenuItem_Click;
            subjectItem.DropDownItems.Add(testItem);
        }
    }
    // Додати кнопку "Вихід"
    ToolStripMenuItem exitItem = new
ToolStripMenuItem("Вихід");
    exitItem.Click += exitToolStripMenuItem_Click;
    this.MainMenuStrip.Items.Add(exitItem);
}
private void exitToolStripMenuItem_Click(object sender,
EventArgs e)
{
    System.Windows.Forms.Application.Restart();
    System.Windows.Forms.Application.Exit();
}

```

```

    }
    public void testToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        ClearQuestions();
        points = 0;
        currentQuestionIndex = 0;
        ToolStripMenuItem selectedItem =
(ToolStripMenuItem)sender;
        ToolStripMenuItem subjectItem =
(ToolStripMenuItem)selectedItem.OwnerItem;
        string subjectName = subjectItem.Text;
        string testName = selectedItem.Text;
        selectedTest = DataBank.tests.Find(test => test.Name
== testName && test.Subject == subjectName);
        label1.Text = "Предмет: " + selectedTest.Subject +
", Рівень складності: " + selectedTest.Difficulty;
        groupBox1.Text = selectedTest.Name;
        ShowAllControls(groupBox1);
        DisplayQuestion();
        startTime = DateTime.Now;
    }
    private void ClearQuestions()
    {
        radioButton1.Checked = false;
        radioButton2.Checked = false;
        radioButton3.Checked = false;
        questionLabel.Text = "";
    }
    private void DisplayQuestion()
    {
        ClearQuestions();
        // Вивести поточне питання на екран
        Question currentQuestion =
selectedTest.Questions[currentQuestionIndex];
        questionLabel.Text = currentQuestion.Name;
        radioButton1.Text = currentQuestion.Answers[0];
        radioButton2.Text = currentQuestion.Answers[1];
        radioButton3.Text = currentQuestion.Answers[2];
    }
    private void nextQuestionButton_Click(object sender,
EventArgs e)
    {
        Question currentQuestion =
selectedTest.Questions[currentQuestionIndex];
        string answer = GetSelectedAnswer(groupBox1);

```

```

        if (currentQuestion.CorrectAnswer == answer)
        {
            points += selectedTest.Difficulty;
        }
        // Перевірити, чи не досягли ми останнього питання
        if (currentQuestionIndex <
selectedTest.Questions.Count - 1)
        {
            currentQuestionIndex++;
            DisplayQuestion();
        }
        else
        {
            TimeSpan dur = DateTime.Now - startTime;
            // Якщо досягнуто останнього питання, показати
повідомлення про завершення тесту
            MessageBox.Show("Ви відповіли на всі питання.
Тест завершено!\nКількість балів: " + points + "\nЧас виконання:
" + $"{dur.Hours:00}:{dur.Minutes:00}:{dur.Seconds:00}");
            HideAllControls(groupBox1);
            // Перевірити, чи користувач вже проходив цей
тест
            TestResult existingResult =
DataBank.currentUser.TestResults.FirstOrDefault(r => r.TestName
== selectedTest.Name);
            if (existingResult != null)
            {
                existingResult.Points = points; // Оновити
бали
                existingResult.Duration = dur; // Оновити
тривалість
            }
            else
            {
                // Додати новий результат, якщо користувач
ще не проходив цей тест
                DataBank.currentUser.TestResults.Add(new
TestResult(selectedTest.Name, points, dur));
            }

DataBank.userManager.UpdateUserSettings(DataBank.currentUser);
    }
}
private void HideAllControls(Control control)
{

```

```

        control.Visible = false;
        foreach (Control c in control.Controls)
        {
            c.Visible = false;
            HideAllControls(c);
        }
    }
    private void ShowAllControls(Control control)
    {
        control.Visible = true;
        foreach (Control c in control.Controls)
        {
            c.Visible = true;
            ShowAllControls(c);
        }
    }
    private string GetSelectedAnswer(GroupBox groupBox)
    {
        RadioButton selectedRadioButton =
groupBox.Controls.OfType<RadioButton>().FirstOrDefault(r =>
r.Checked);
        return selectedRadioButton?.Text;
    }
    protected override void
OnFormClosing(FormClosingEventArgs e)
    {
        DataBank.testManager.SaveTests(DataBank.tests);
        base.OnFormClosing(e);
    }
}
public class Test
{
    public string Name { get; set; }
    public string Subject { get; set; }
    public int Difficulty { get; set; }
    public List<Question> Questions { get; set; }
    public Test(string name, string subject, int difficulty)
    {
        Name = name;
        Subject = subject;
        Difficulty = difficulty;
        Questions = new List<Question>();
    }
    public void AddQuestion(Question question)
    {
        Questions.Add(question);
    }
}

```

```

    }
    public override string ToString()
    {
        return Name;
    }
}
public class Question
{
    public string Name { get; set; }
    public List<string> Answers { get; set; }
    public string CorrectAnswer { get; set; }
    public Question()
    {
        Answers = new List<string>();
    }
    public override string ToString()
    {
        return Name;
    }
}
public class TestManager
{
    private readonly string filePath = "tests.json";
    public List<Test> LoadTests()
    {
        if (!File.Exists(filePath))
        {
            return new List<Test>();
        }
        using (StreamReader file = File.OpenText(filePath))
        {
            JsonSerializer serializer = new
JsonSerializer();
            return (List<Test>)serializer.Deserialize(file,
typeof(List<Test>));
        }
    }
    public void SaveTests(List<Test> tests)
    {
        using (StreamWriter file =
File.CreateText(filePath))
        {
            JsonSerializer serializer = new
JsonSerializer();
            serializer.Serialize(file, tests);
        }
    }
}

```

}
}
}