ДИПЛОМНА РОБОТА
БАКАЛАВРА

ДРБ. ІП - 67.00.00.000 ПЗ

Група ІП-20-1

Припхан Олександр

2024

Міністерство освіти і науки України
Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

УДК 004.4

ДИПЛОМНА РОБОТА БАКАЛАВРА

Тема: " Побудова студенто-центричного програмного рішення тестування знань "

(назва згідно з наказом ректора)

спеціальність – 121 "Інженерія програмного забезпечення"

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДРБ.ІП - 67.00.00.000 ПЗ

(позначення)

Консультанти:

з питань нормоконтролю

асист. /М.М. Піх/
(посада) (підпис) (дата)(розшифровка підпису)

Студент

ІП-20-1 /О.В.Припхан/
(шифр групи) (підпис) (дата) (розшифровка підпису)

Керівник роботи

к.т.н., доц. /В.В.Бандура/
(посада) (підпис) (дата) (розшифровка підпису)

Допускається до захисту
Завідувач кафедри

к.т.н., доц. /В.В.Бандура/
(посада) (підпис) (дата) (розшифровка підпису)

2024
(рік)

Івано-Франківський національний технічний університет нафти і газу

Інститут, факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Освітньо-кваліфікаційний рівень бакалавр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**
доц. В.В. Бандура
“ ____ ” _____2024 р.

ЗАВДАННЯ

НА ДИПЛОМНУ РОБОТУ БАКАЛАВРА СТУДЕНТОВІ

Припхану Олександру Володимировичу

(прізвище, ім’я, по-батькові)

1. Тема проекту (роботи) “ Побудова студенто-центричного програмного рішення тестування знань”

керівник проекту (роботи) Бандура Вікторія Валеріївна, доцент, к.т.н.

затверджені наказом закладу вищої освіти від “ 07 ” червня 2024 р. № 347/7

2. Строк подання студентом проекту (роботи) 21 червня 2024 р.

3. Вихідні дані до проекту (роботи) Результати і матеріали отримані під час проходження переддипломної практики

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

1. Аналіз предметної області розробки систем автоматизованого тестування знань студентів
2. Аналіз існуючих розробок в області автоматизованого контролю знань
3. Алгоритмічна реалізація та моделювання програмного забезпечення тестування знань
4. Побудова діаграми елементів та UML моделювання
5. Програмна реалізація студенто-центричного рішення тестування знань

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Класифікація тестових завдань (рис. 1.1)
2. Схема взаємодії в тріаді «Викладач - Система тестування - Студент» (рис. 1.2)
3. Схема процесу реалізації відображення вмісту SCORM-пакету (рис. 1.3)
4. Google-форма на тему «Комп’ютерні віруси та антивірусні програми» (рис. 1.4)
5. Тест на тему «Загальна характеристика програми Prolog» (рис. 1.5)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання **22 травня 2024 р.**

Керівник _____
(підпис)

Завдання прийняв до виконання_____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту	Примітка
1	Аналіз предметної області розробки систем автоматизованого тестування знань студентів	26.05.2024	виконано
2	Аналіз існуючих розробок в області автоматизованого контролю знань	29.05.2024	виконано
3	Алгоритмічна реалізація та моделювання програмного забезпечення тестування знань	30.05.2024	виконано
4	Побудова діаграми елементів та UML моделювання	04.06.2024	виконано
5	Програмна реалізація студенто-центричного рішення тестування знань	10.06.2024	виконано
6	Оформлення пояснювальної записки дипломної роботи завідувачем кафедри	21.06.2024	виконано

Студент – дипломник _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Метою дипломної роботи є розробка ефективної та зручної системи тестування знань, орієнтованої на студентів. Це програмне рішення спрямоване на покращення якості освіти шляхом забезпечення індивідуального підходу до оцінювання знань кожного студента, надання зворотного зв'язку та аналітичних інструментів для викладачів. Для досягнення поставленої мети використовуються сучасні технології програмування та розробки програмного забезпечення. Основною мовою програмування обрана C# у поєднанні з платформою .NET, що забезпечує високу продуктивність та масштабованість системи. Для зберігання та обробки даних використовується база даних Microsoft SQL Server.

В результаті роботи було створено прототип програмного рішення, яке включає наступні функціональні можливості: створення та керування тестами, адаптивне тестування та реальний час зворотного зв'язку.

КЛЮЧОВІ СЛОВА: СТУДЕНТО-ЦЕНТРИЧНЕ НАВЧАННЯ, ТЕСТУВАННЯ ЗНАНЬ, АДАПТИВНЕ ТЕСТУВАННЯ, ЗВОРОТНИЙ ЗВ'ЯЗОК, АНАЛІТИКА, C#, .NET, ОСВІТНІ ТЕХНОЛОГІЇ.

ANNOTATION

The aim of the thesis is to develop an effective and convenient knowledge testing system aimed at students. This software solution is aimed at improving the quality of education by providing an individual approach to assessing the knowledge of each student, providing feedback and analytical tools for teachers. To achieve the goal, modern programming and software development technologies are used. The main programming language is C# in combination with the .NET platform, which ensures high performance and scalability of the system. The Microsoft SQL Server database is used for data storage and processing.

As a result of the work, a prototype of the software solution was created, which includes the following functionalities: creation and management of tests, adaptive testing and real-time feedback.

KEYWORDS: STUDENT-CENTERED LEARNING, KNOWLEDGE TESTING, ADAPTIVE TESTING, FEEDBACK, ANALYTICS, C#, .NET, EDUCATIONAL TECHNOLOGY.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 8
ВСТУП 9

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ РОЗРОБКИ СИСТЕМ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ ЗНАНЬ СТУДЕНТІВ 11
1.1. Основні аспекти застосування тестових технологій для контролю знань студентів 11
1.2 Аналіз існуючих розробок в області автоматизованого контролю знань 13
1.3. Формалізація складових системи тестування знань 33
1.4. Організація формування тестових завдань і процесу тестування знань 36

2. АЛГОРИТМІЧНА РЕАЛІЗАЦІЯ ТА МОДЕЛЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СТУДЕНТО-ЦЕНТРИЧНОГО РІШЕННЯ ТЕСТУВАННЯ ЗНАНЬ 40
2.1. Розробка діаграм використання для акторів розроблюваної системи .. 40
2.2. Розробка прототипів та макетів інтерфейсу користувача 45
2.3. Розробка архітектури системи тестування 49
2.4. Побудова діаграми елементів та UML моделювання 51
2.5. Вибір та опис інструментів для розробки 56

3. ПРОГРАМНА РЕАЛІЗАЦІЯ СТУДЕНТО-ЦЕНТРИЧНОГО РІШЕННЯ ТЕСТУВАННЯ ЗНАНЬ 60
3.1. Розробка діаграм потоків даних 60

3.2. Моделювання структури бази даних з використанням ER-діаграм 63
3.3. Програмна реалізація системи тестування 68
3.4. Розробка та опис інтерфейсу користувача студенто-центричного рішення тестування знань 72

ВИСНОВКИ 76
ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 77
БІБЛІОГРАФІЧНА ДОВІДКА

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

GC (Garbage Collection) – одна із форм автоматичного керування пам’яттю

GUI (Graphical User Interface) – графічний інтерфейс користувача

JVM (Java Virtual Machine) – віртуальна машина для виконання Java програм

PEP (Python Enhancement Proposal) – стандарт написання коду на Python

XML (Extensible Markup Language) – мова для опису даних за допомогою тегів

BSI (British Standards Institution) - Британський інститут стандартів

ВСТУП

Національна програма «Освіта. Україна XXI століття» передбачає розвиток вищої в Україні освіти на основі нових прогресивних концепцій, впровадження в навчальний процес нових інформаційних технологій та науково- методичних досягнень.

Соціально-економічні перетворення, що відбуваються в Україні, в тому числі інформатизація суспільства, вимагають значних змін у всіх сферах діяльності держави. В першу чергу це стосується системи вищої освіти, від ефективності реформування якої суттєво залежить рівень підготовки випускників закладів вищої освіти (ЗВО).

Підвищення ефективності освіти можливе при впровадженні в навчальний процес комп'ютеризованих систем навчання, які:

- забезпечують адаптацію процесу навчання до індивідуальних характеристик осіб, що навчаються;
- спрощують процес представлення навчальної інформації і контролю знань;
- сприяють розробці і впровадженню нових методів контролю знань.

Однією з найважливіших задач ЗВО є забезпечення високоякісної підготовки фахівців за всім спектром ліцензованих спеціальностей. Саме тому в систему освіти останнім часом інтенсивно впроваджуються нові інформаційні технології, що забезпечують необхідний рівень автоматизації процесів навчання і якісного контролю знань.

Аналіз традиційно існуючої практики перевірки знань дозволяє виділити ряд недоліків, що в деяких випадках мають місце, зокрема: стихійність, тобто, відсутність систематичності поточного контролю знань; нераціональне використання ефективних методів і форм контролю; відсутність дидактичної цілеспрямованості; не врахування узагальнених та

характерних особливостей навчального матеріалу та специфічних умов роботи в аудиторії тощо. Доцільно також відмітити основні чинники, які впливають на результати контролю та оцінки знань: психологічні фактори; загальна і спеціальна підготовка викладача та особисті якості викладача (принциповість, почуття відповідальності), від яких суттєво залежать помилки великодушності, помилки ореола, помилки центральної тенденції, помилки контрасту, помилки близькості, логічні помилки.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ РОЗРОБКИ СИСТЕМ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ ЗНАНЬ СТУДЕНТІВ

1.1. Основні аспекти застосування тестових технологій для контролю знань студентів

Оцінювання знань в освітніх закладах може здійснюватися у ЗВО традиційними й нетрадиційними методами. До останніх належить комп'ютерне тестування, основними перевагами якого є:

- об'єктивність результатів перевірки, що базується на заздалегідь визначеному еталоні відповідей;
- підвищення ефективності контролю знань студентів з боку викладача за рахунок регулярності тестування;
- можливість автоматизації перевірки результатів тестування знань студентів;
- можливість використання процедур комп’ютерного тестування в системах дистанційної освіти.

В наш час значного поширення на різних фазах навчання набули різні форми тестових опитувань. При цьому спектр застосування тестів досить широкий - від короткого опитування після пояснення поточної теми до підсумкових, випускних або вступних іспитів. Під тестовим контролем в подальшому будемо розуміти спеціально підготовлений контрольний набір завдань, що дозволяє кількісно, надійно і адекватно оцінити знання студентів на основі використання статистичних підходів та методів для вибору тестів та узагальнення і обробки результатів тестування.

Розробка системи тестування - це складний процес, який вимагає спільної роботи багатьох розробників, використання різноманітних технологій, розгляду і врахування психологічно-педагогічних аспектів процесу навчання. Саме останньому повинна приділятися велика увага при

розробці будь-яких програмних компонент для систем комп'ютерного тестування. При створенні систем автоматизованого тестування необхідно обов'язково враховувати всі особливості предметних областей та особливості застосування різних методів та засобів теорії навчання.

Вибір конкретного методу тестування та типу тестових завдань залежить від цільової мети тестового контролю і попередньо обраних показників оцінки рівня знань. Класифікація тестових завдань в залежності від їх характерних ознак представлена на рисунку 1.1.



Рисунок 1.1 - Класифікація тестових завдань

На рисунку 1.2 представлена алгоритмічна схема зв'язків викладача та студента на основі застосування автоматизованої системи тестування.

За допомогою зовнішнього зворотного зв'язку навчальна інформація в даній тріаді (рис. 1.2) надходить до викладача і потім використовується ним для адаптивної корекції діяльності студента і навчальної програми.

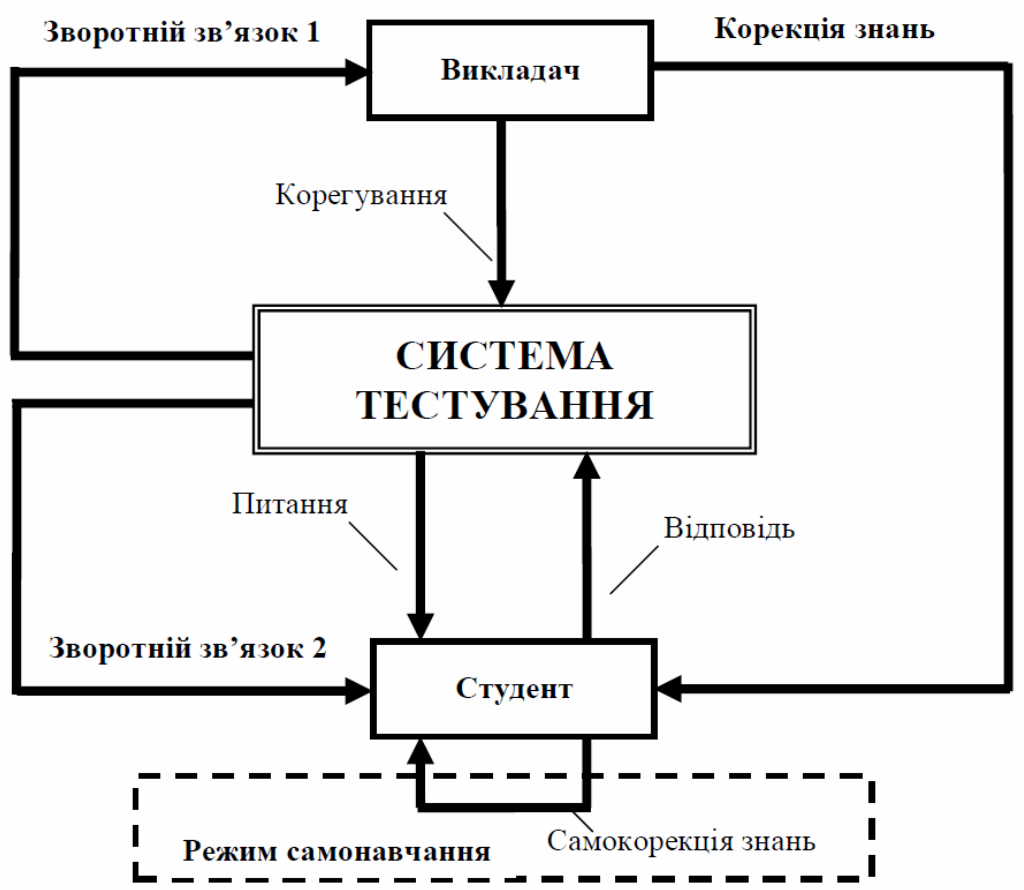


Рисунок 1.2 - Схема взаємодії в тріаді «Викладач - Система тестування - Студент»

1.2 Аналіз існуючих розробок в області автоматизованого контролю знань

Крім систем онлайн навчання у ЗВО, існує успішний досвід застосування дистанційного навчання в сфері інформаційних технологій такими компаніями як Microsoft і CISCO Network. Дані компанії пропонують дистанційне навчання для фахівців в області інформаційних технологій з метою підвищення свого професійного рівня, а так само надають можливість отримання міжнародних сертифікатів, що підтверджують знань в будь-якій країні світу. Залежно від засобів дистанційного навчання та форм комунікації можна виділити три різновиди технологічної організації ДН]: одинична, мультимедіа, гіпермедіа.

Модель одиничної медіа означає використання будь-якого засобу придбання знань і каналу передачі інформації, наприклад навчання через чат, навчальні радіо- чи телепередачі. В даному випадку основним засобом навчання є, друкований текст. Практично відсутня обопільна комунікація, що наближає цю модель ДН до традиційного заочного навчання.

У мультимедійному підході до ДН використовують засоби навчання - навчальні посібники в друкованому вигляді, комп'ютерні навчальні програми на різних носіях, аудіо- та відеозаписи і т.д. Однак переважає при цьому передача навчальної інформації в «одну сторону». При необхідності використовуються частини очного навчання - мітинги навчаються і викладачів, підведення підсумкових навчальних семінарів або консультацій, очні іспити і т. П..

Моделлю ДН нового покоління є гіпермедіа, що передбачає використання ІТ при переважній ролі телекомунікацій. Найбільш простою формою при цьому є використання електронної пошти і відеозв'язку, а також аудіонавчання. При поточному розвитку ця модель ДН включає використання таких коштів, як відео, телефон і телефакс (для проведення відеоконференцій) при одночасному широкому використанні різних засобів.

Класичні методи розробки онлайн навчальних матеріалів, найчастіше, дороги, вимагають багато часу і вимагають спеціалізованих навичок. Щоб повністю реалізувати онлайнове навчання, компанії використовують системи управління вмістом навчання - LCMS - для швидкої організації, розгортання та управління контентом онлайнових курсів. LCMS - це програмний та апаратний комплекс, який використовується для представлення, зберігання, збирання і доставки користувачеві персоналізованого навчального контенту в формі «навчальних об'єктів»).

У 2000 р ініціативна група ADL розробила стандарт SCORM, створений для систем дистанційного навчання. Даний стандарт містить ряд вимог до організації навчального контенту і всієї системи ДО. SCORM

заснований на стандарті XML і дозволяє підтримати сумісність компонентів і можливість їх багаторазового використання: навчальний матеріал представлений невеликими розділами, які можуть включатися в навчальні курси і використовуватися системою ДН незалежно від того, ким, де і з допомогою чого були створені.

SCORM - це набір специфікацій і стандартів, які представлені в кілька розділів:

- Модель зберігання змісту;
- Серед поточних виконання;
- Пошук і навігація.

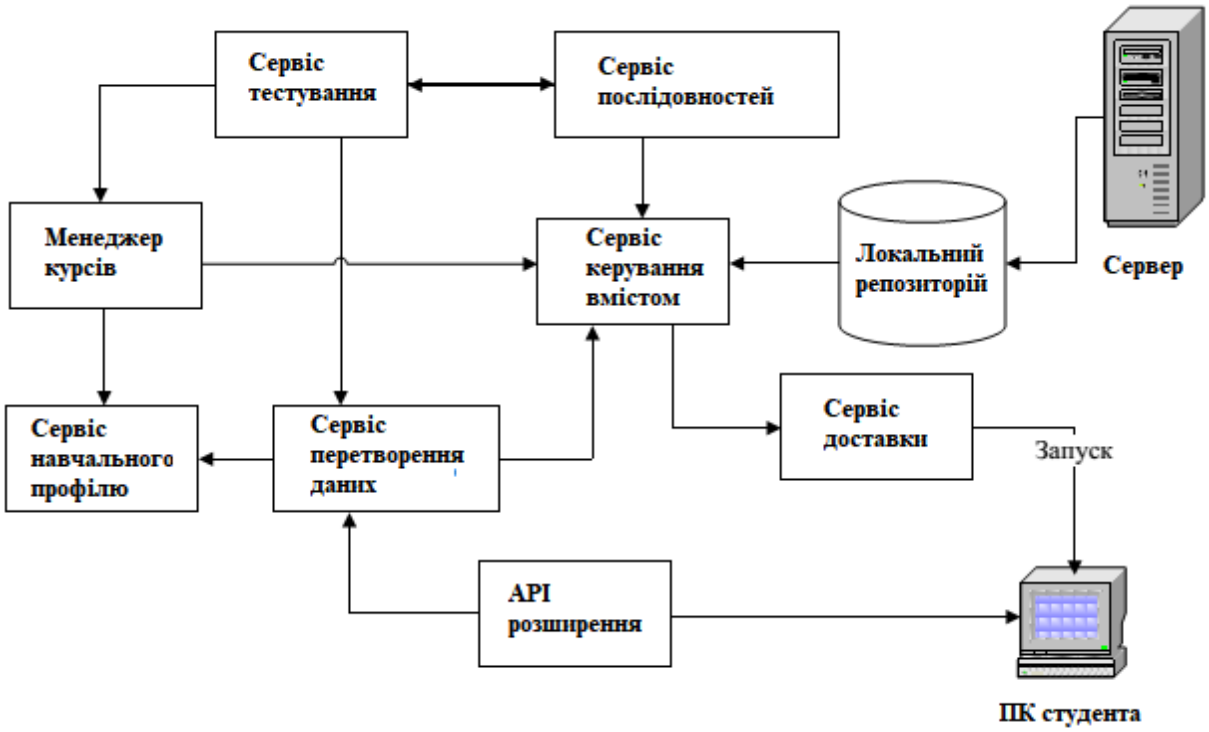


Рисунок 1.3 - Схема процесу реалізації відображення вмісту SCORM-пакету

Google Форми – це один з найшвидших і простих способів створити опитування або тест: потрібно записати завдання та вибрати тип відповіді (рис. 1.4). Для використання ресурсу потрібно лише мати аккаунт Google (для прикладу: https://docs.google.com/forms/d/e/1FeazjZ3f8V9pmw/viewform

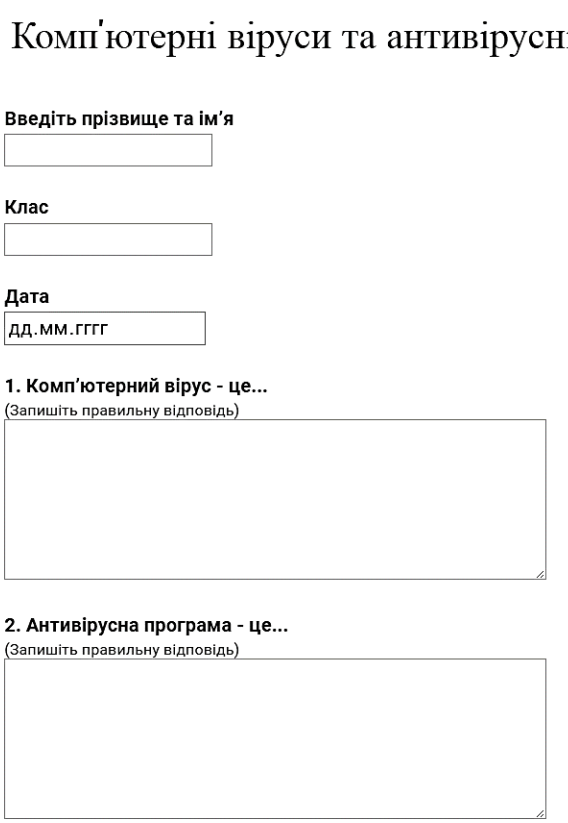


Рисунок 1.4 - Google-форма на тему «Комп'ютерні віруси та антивірусні програми» (Початок)

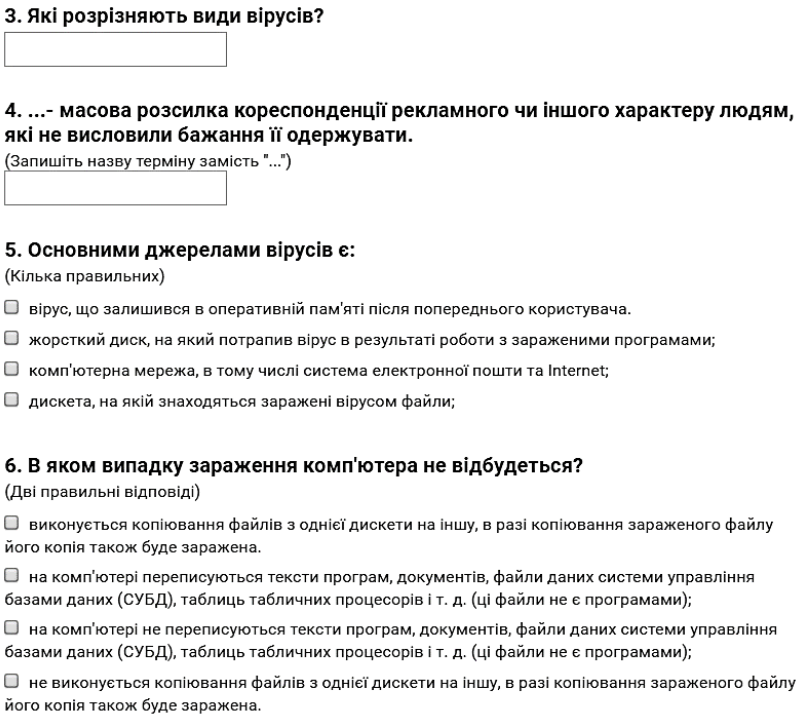


Рисунок 1.4 - Google-форма на тему «Комп'ютерні віруси та антивірусні програми» (Завершення)

LEARNINGAPPS.ORG – безкоштовний он-лайн сервіс, який дозволяє зберегти створені додатки. Пропонується за допомогою готових шаблонів створити власні розробки (рис. 1.5) та опублікувати для інших користувачів. Потрібно зареєструватися на LearningApps для того, щоб мати можливість створювати інтерактивні завдання.

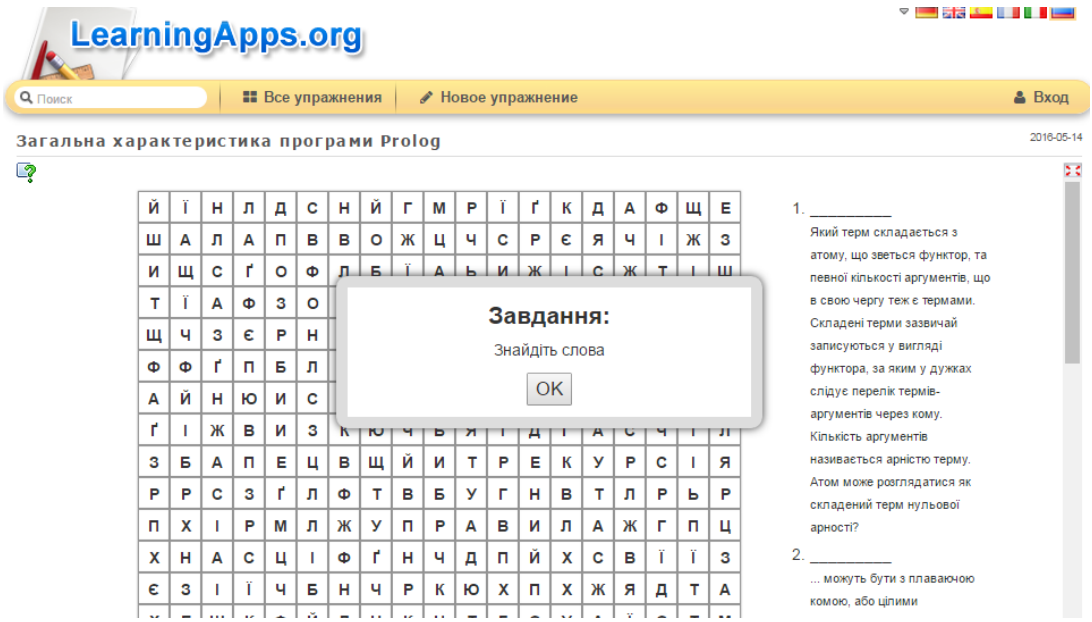


Рисунок 1.5 - Тест на тему «Загальна характеристика програми Prolog»

LearningApps.org – це безкоштовний онлайн-сервіс для створення, обміну та використання інтерактивних навчальних матеріалів. Основні характеристики та функції сервісу включають:

1. Створення інтерактивних завдань:

- Платформа пропонує різні шаблони для створення інтерактивних завдань, таких як вікторини, пазли, хмари слів, хронологічні лінії, кросворди тощо.

- Учителі та учні можуть створювати власні завдання або використовувати вже готові шаблони.

2. Різноманітні формати завдань:

- Сервіс підтримує текстові, графічні та мультимедійні формати, що дозволяє створювати багатофункціональні навчальні матеріали.

- Завдання можуть містити текст, зображення, аудіо та відео.

3. Зручний інтерфейс:

- Інтерфейс LearningApps.org інтуїтивно зрозумілий і легкий у використанні.

- Налаштування завдань проводиться через візуальні редактори, що не вимагають знання програмування.

4. Спільне використання та інтеграція:

- Користувачі можуть ділитися створеними завданнями з іншими через посилання або вбудовувати їх на вебсайти та в електронні платформи навчання.

- Платформа підтримує інтеграцію з іншими навчальними ресурсами та системами управління навчанням.

5. Оцінювання та зворотний зв'язок:

- Учні можуть проходити тести та отримувати миттєвий зворотний зв'язок про свої результати.

- Вчителі можуть відстежувати успішність виконання завдань своїми учнями.

6. Спільнота користувачів:

- LearningApps.org має активну спільноту користувачів, яка ділиться своїми завданнями та досвідом використання сервісу.

- На платформі доступний великий каталог вже створених завдань з різних предметів та тем.

LearningApps.org – це потужний інструмент для створення інтерактивного навчального контенту, який допомагає зробити процес навчання більш цікавим і ефективним як для вчителів, так і для учнів.

PROPROFS – дозволяє створити тести будь-якого вигляду - можна запропонувати на вибір один або кілька варіантів, заповнити пропущене слово або написати розгорнуту відповідь. Сервіс дозволяє вставляти в завдання текстові документи та презентації, файли PDF, а також зображення,

аудіо- та відеофайли. Завершивши роботу над тестом, можна залишити його в загальному доступі на сайті Proprofs (рис. 1.6) або вбудувати на свою сторінку.

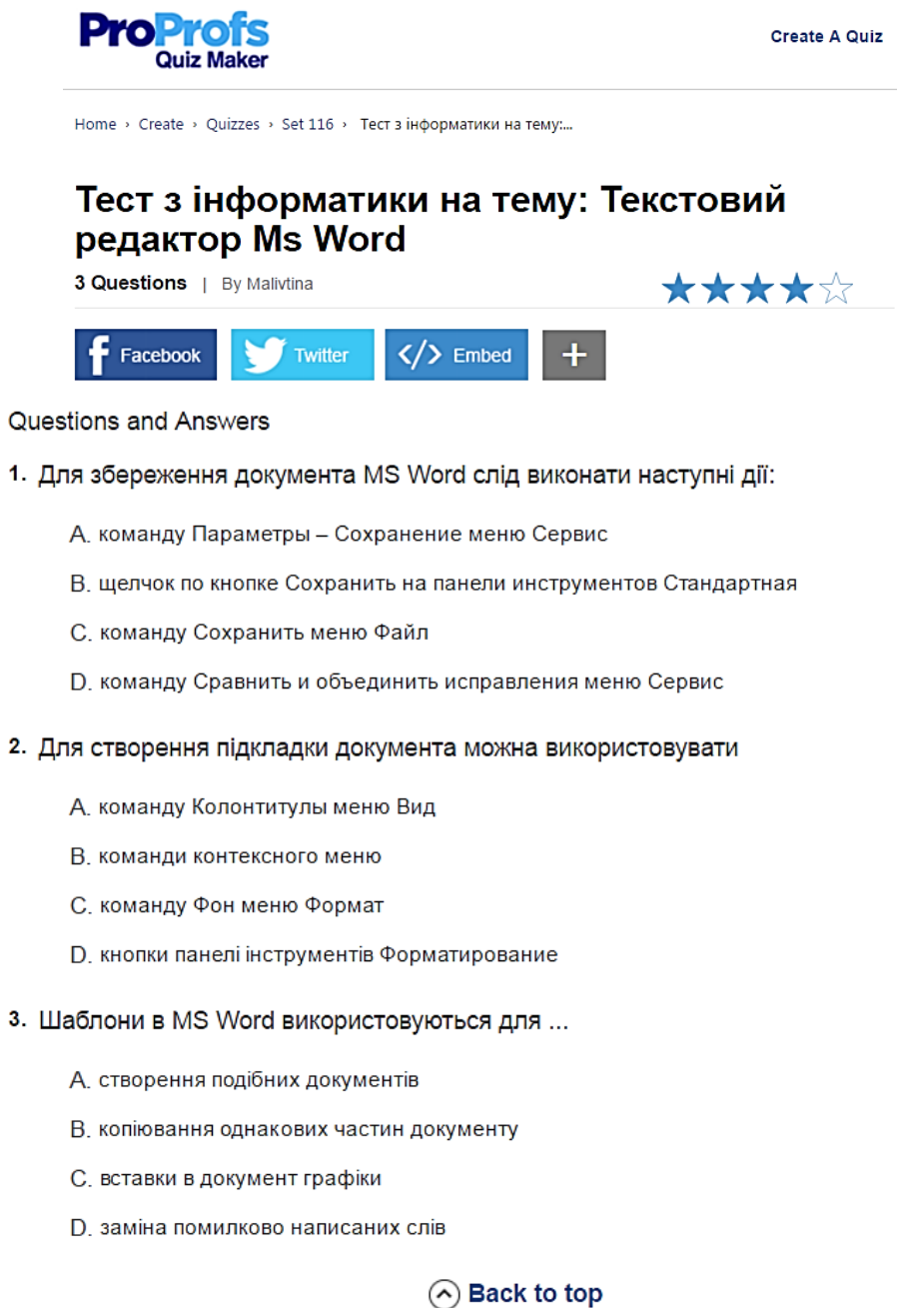


Рисунок 1.6 - Створення тесту у ProProfs

У ClassMarker можна робити опитування з різними форматами відповідей - окрім звичних варіантів наявні навіть есе. Для початку роботи викладачу потрібно створити віртуальний клас і розіслати запрошення-коди учням. ClassMarker зберігає результати всіх проведених тестів, ведучи

статистику успішності. Якщо у викладача є власна веб-сторінка, він може вбудовувати тестові завдання на неї.

ClassMarker — це онлайн-система тестування, яка дозволяє створювати та керувати тестами для студентів, співробітників або інших груп користувачів. Ця платформа використовується для оцінювання знань, проведення опитувань та екзаменів в різних галузях, таких як освіта, бізнес, тренінги та інші.

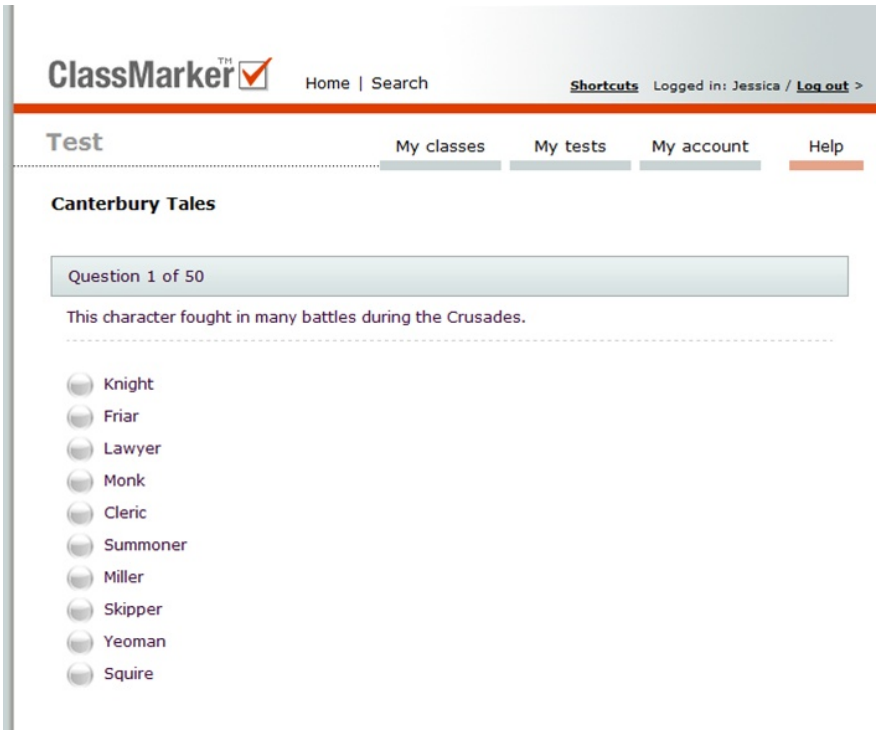


Рисунок 1.7 – Вигляд тесту в ClassMarker

Основні функції ClassMarker:

1. Створення тестів: Користувачі можуть створювати тести з різними типами запитань, включаючи множинний вибір, правдивість/хибність, короткі відповіді, есе та інші.

2. Керування тестами: Можливість налаштовувати тести, включаючи час проходження, кількість спроб, доступність та інші параметри.

3. Оцінювання: Автоматичне оцінювання відповідей та надання детальних звітів про результати тестування.

4. Безпека: Опції для забезпечення безпеки тестування, такі як випадковий порядок запитань, обмеження часу та унікальні посилання для доступу.

5. Аналітика: Надання докладних звітів та аналізу результатів, які допомагають оцінити прогрес та виявити області для поліпшення.

6. Інтеграція: Можливість інтеграції з іншими системами навчання та управління, такими як Learning Management Systems (LMS).

ClassMarker популярний серед освітніх установ, тренінгових компаній та підприємств, які шукають зручний та ефективний спосіб оцінки знань та навичок своїх користувачів.

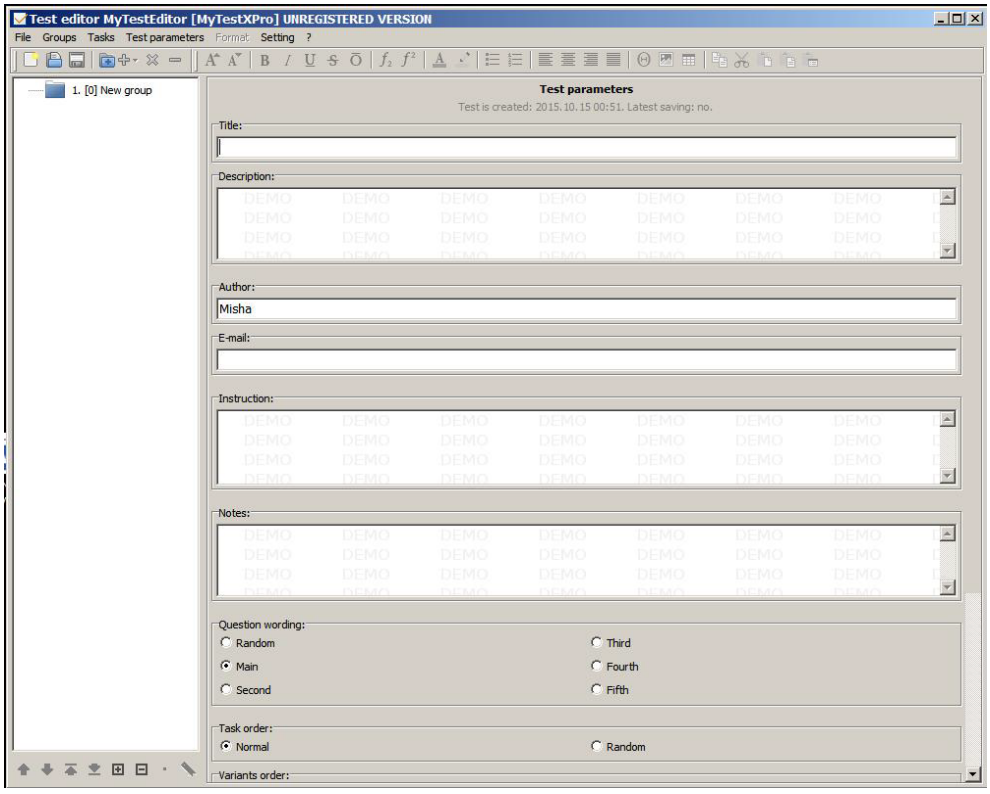


Рисунок 1.8 – Сторінка створення завдання засобами програми MyTestStudentPRO

Програма MyTest має вбудований текстовий редактор, за допомогою якого можна створювати, а також редагувати запитання та відповіді

практично будь-якої складності. Також перевагою даної програми є те, що вона працює з дев’ятьма типами завдань.

MyTest — це програма для створення та проведення комп'ютерних тестів, яка широко використовується в навчальних закладах та для професійного тестування. Вона має ряд особливостей, які роблять її популярною серед користувачів:

1. Різноманітність типів запитань: MyTest підтримує різні типи запитань, такі як:

- Множинний вибір
- Відкриті запитання (коротка відповідь)
- Заповнення пропусків
- Встановлення відповідності
- Послідовність (порядок)
- Запитання на відповідність
- Інтерактивні запитання (наприклад, вибір областей на зображенні)

2. Гнучкість налаштувань: Програма дозволяє налаштовувати багато параметрів тестування, включаючи:

- Кількість спроб на тест
- Обмеження часу
- Випадковий порядок запитань і відповідей
- Налаштування складності тесту

3. Оцінювання та звіти: MyTest надає можливості для автоматичного оцінювання відповідей та формування докладних звітів по результатам тестування. Можна отримати інформацію про загальні результати, середній бал, час виконання тесту тощо.

4. Зручний інтерфейс: Програма має інтуїтивно зрозумілий інтерфейс, який дозволяє легко створювати та редагувати тести, а також керувати ними.

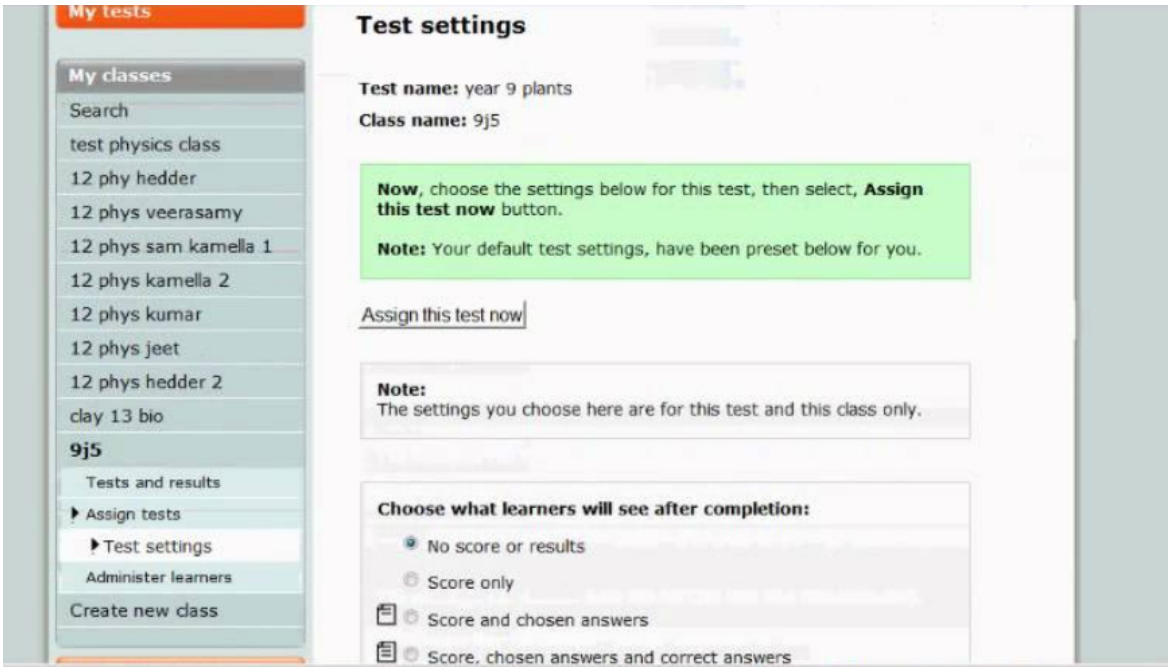


Рисунок 1.9 – Вигляд тестів в системі MyTest

5. Безпека: MyTest пропонує функції для забезпечення безпеки тестування, такі як захист тесту паролем, обмеження доступу до тесту за IP-адресою або по часу.

6. Можливість імпорту та експорту: Програма підтримує імпорт та експорт тестів у різних форматах, що спрощує обмін тестами між різними користувачами та системами.

7. Статистика та аналіз: MyTest пропонує інструменти для детального аналізу результатів тестування, що дозволяє виявляти сильні та слабкі сторони тестованих.

8. Локалізація та підтримка різних мов: Програма підтримує кілька мов, що робить її доступною для користувачів з різних країн.

MyTest підходить як для індивідуального використання, так і для використання в навчальних закладах та компаніях, які потребують ефективного інструменту для оцінювання знань та навичок.

Система тестування "ExamBuilder" — це інтегроване програмне рішення, призначене для створення, проведення та аналізу тестів з метою оцінки знань і навичок студентів або співробітників. Вона використовується

в освітніх установах, тренінгових центрах і на підприємствах для організації ефективного процесу тестування.

Основні особливості системи "ExamBuilder":

1. Створення тестів:

- Різноманітність типів запитань: Підтримка різних типів запитань, включаючи множинний вибір, відкриті запитання, заповнення пропусків, встановлення відповідності, ранжування та інтерактивні запитання.

- Банк запитань: Можливість зберігання запитань в централізованому банку для повторного використання в різних тестах.

- Гнучке налаштування тестів: Встановлення часу на виконання тесту, кількості спроб, випадковий порядок запитань і відповідей, налаштування критеріїв оцінювання.

2. Проведення тестування:

- Доступність на різних платформах: Тести можуть проходити на різних пристроях, включаючи комп'ютери, планшети і смартфони.

- Безпека тестування: Захист тестів паролем, обмеження доступу за IP-адресами, шифрування даних, а також функції для запобігання шахрайству, такі як моніторинг активності користувача і контроль доступу.

- Інтуїтивно зрозумілий інтерфейс: Зручний і простий інтерфейс для студентів і тестованих, що дозволяє легко орієнтуватися в системі.

3. Оцінювання та аналіз результатів:

- Автоматичне оцінювання: Миттєва перевірка тестів і автоматичне надання результатів студентам.

- Детальні звіти: Генерація звітів для викладачів і адміністраторів з детальною інформацією про результати тестування, включаючи статистику по кожному запитанню, загальні показники успішності і тенденції.

- Аналітика: Інструменти для глибокого аналізу результатів тестування, що дозволяють виявляти слабкі місця і адаптувати навчальні матеріали відповідно.

4. Керування користувачами і групами:

- Ролі та права доступу: Налаштування різних рівнів доступу для адміністраторів, викладачів і студентів.

- Керування групами: Можливість створення і управління групами студентів, що дозволяє організовувати тестування для певних груп або класів.

5. Інтеграція з іншими системами:

- Інтеграція з LMS: Підтримка інтеграції з системами управління навчанням (LMS) через API або інші методи, що забезпечує безперебійний обмін даними і зручність у використанні.

- Імпорт/Експорт даних: Можливість імпорту та експорту тестів і результатів у різних форматах для спрощення обміну даними між різними системами.

Переваги використання "ExamBuilder":

1. Ефективність: Автоматизація процесу тестування знижує витрати часу і зусиль викладачів на створення та перевірку тестів.

2. Точність оцінювання: Автоматичне оцінювання і аналіз результатів забезпечують точність і об'єктивність оцінювання знань студентів.

3. Адаптивність: Гнучкі налаштування і можливість адаптації тестів під індивідуальні потреби студентів роблять систему зручною і ефективною для різних навчальних потреб.

4. Масштабованість: Система підходить для використання в освітніх установах різного розміру, від шкіл до університетів і корпоративних тренінгових центрів.

"ExamBuilder" — це сучасне і потужне рішення для організації тестування знань, яке сприяє підвищенню якості навчання та ефективності оцінювання знань. Система тестування знань "ExamBuilder" має ряд особливостей, які роблять її ефективним інструментом для оцінювання знань. Ось основні особливості "ExamBuilder":

1. Створення тестів:

- Різноманітні типи запитань: Підтримка множинного вибору, відкритих запитань, заповнення пропусків, встановлення відповідності, ранжування та інтерактивних запитань (наприклад, вибір областей на зображенні).

- Банк запитань: Централізоване зберігання запитань, що дозволяє легко створювати нові тести з існуючих запитань.

- Гнучке налаштування: Можливість встановлення часу на виконання тесту, кількості спроб, випадковий порядок запитань та відповідей, а також налаштування критеріїв оцінювання.

2. Проведення тестування:

- Кросплатформенність: Підтримка роботи на різних пристроях, включаючи ПК, планшети та смартфони, що дозволяє студентам проходити тести зручним для них способом.

- Безпека тестування: Захист паролем, обмеження доступу за IP-адресами, шифрування даних, функції для запобігання шахрайству, такі як моніторинг активності користувача і контроль доступу.

- Інтуїтивно зрозумілий інтерфейс: Зручний інтерфейс для тестованих, що полегшує орієнтацію в системі та проходження тестів.

3. Оцінювання та аналіз результатів:

- Автоматичне оцінювання: Швидке та точне оцінювання відповідей з миттєвим наданням результатів студентам.

- Детальні звіти: Генерація звітів для викладачів і адміністраторів з детальною інформацією про результати тестування, включаючи статистику по кожному запитанню, загальні показники успішності та тенденції.

- Аналітика: Інструменти для глибокого аналізу результатів тестування, що дозволяють виявляти слабкі місця та адаптувати навчальні матеріали відповідно.

4. Керування користувачами та групами:

- Ролі та права доступу: Налаштування різних рівнів доступу для адміністраторів, викладачів та студентів, що забезпечує безпеку і конфіденційність даних.

- Керування групами: Можливість створення та управління групами студентів, що дозволяє організовувати тестування для певних груп або класів.

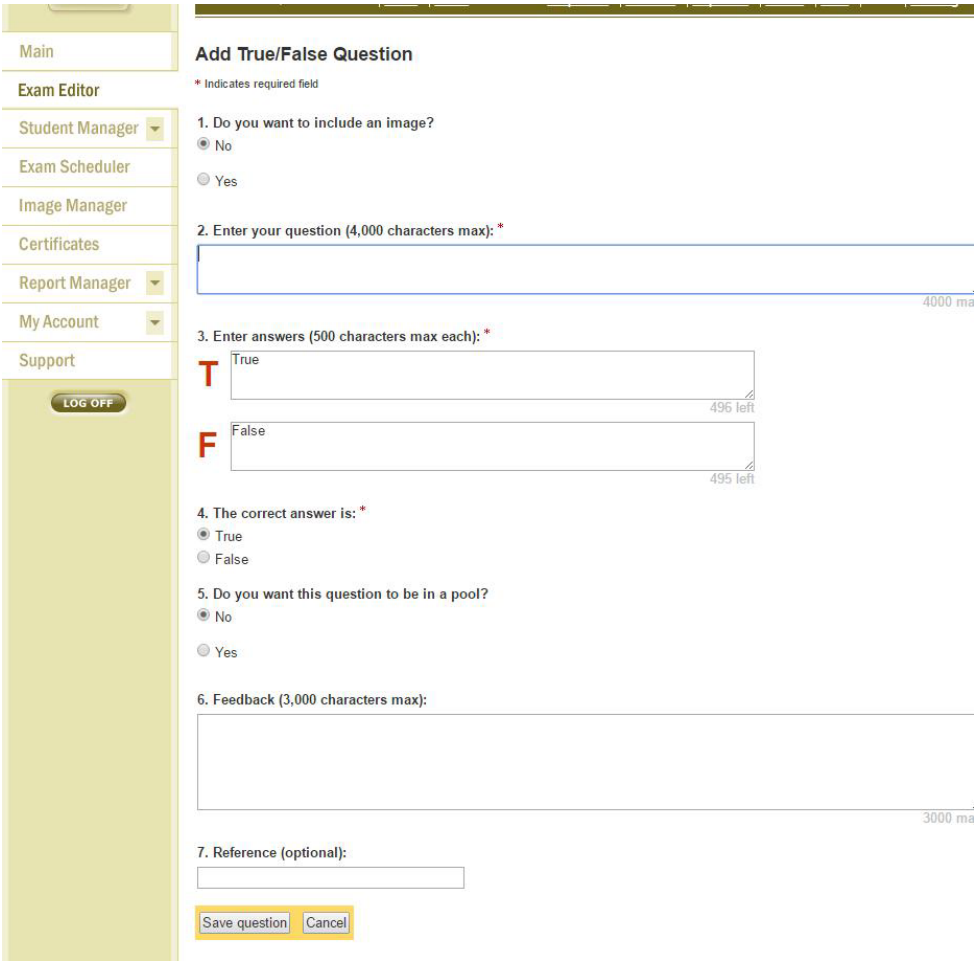


Рисунок 1.10 – Сторінка створення завдання системи Exambuilder

5. Інтеграція з іншими системами:

- Інтеграція з LMS: Підтримка інтеграції з системами управління навчанням (LMS) через API або інші методи, що забезпечує безперебійний обмін даними та зручність у використанні.

- Імпорт/Експорт даних: Можливість імпорту та експорту тестів і результатів у різних форматах для спрощення обміну даними між різними системами.

6. Підтримка асинхронного навчання:

- Зворотний зв'язок у реальному часі: Студенти отримують миттєвий зворотний зв'язок про свої результати, що сприяє швидшому засвоєнню матеріалу.

- Адаптивне тестування: Система налаштовує рівень складності тестів відповідно до знань кожного студента, що дозволяє забезпечити індивідуальний підхід до навчання.

7. Масштабованість та надійність:

- Висока продуктивність: Підтримка великої кількості одночасних користувачів без втрати продуктивності.

- Надійність: Використання сучасних технологій для забезпечення стабільної роботи системи та захисту даних.

Переваги використання "ExamBuilder":

1. Ефективність: Автоматизація процесу тестування знижує витрати часу і зусиль викладачів на створення та перевірку тестів.

2. Точність оцінювання: Автоматичне оцінювання і аналіз результатів забезпечують точність і об'єктивність оцінювання знань студентів.

3. Адаптивність: Гнучкі налаштування і можливість адаптації тестів під індивідуальні потреби студентів роблять систему зручною і ефективною для різних навчальних потреб.

4. Масштабованість: Система підходить для використання в освітніх установах різного розміру, від шкіл до університетів і корпоративних тренінгових центрів.

"ExamBuilder" забезпечує зручний і ефективний інструмент для організації комп'ютерного тестування знань, сприяє підвищенню якості

освіти та допомагає виявляти і розвивати знання та навички студентів або співробітників.

Основною особливістю продукту KTCNet 3 є його спрямованість на розподілену мережеву взаємодію з користувачами та адміністраторами комплексу. Такий підхід дозволяє максимально ефективно управляти системою й оперативно отримувати інформацію про результати.

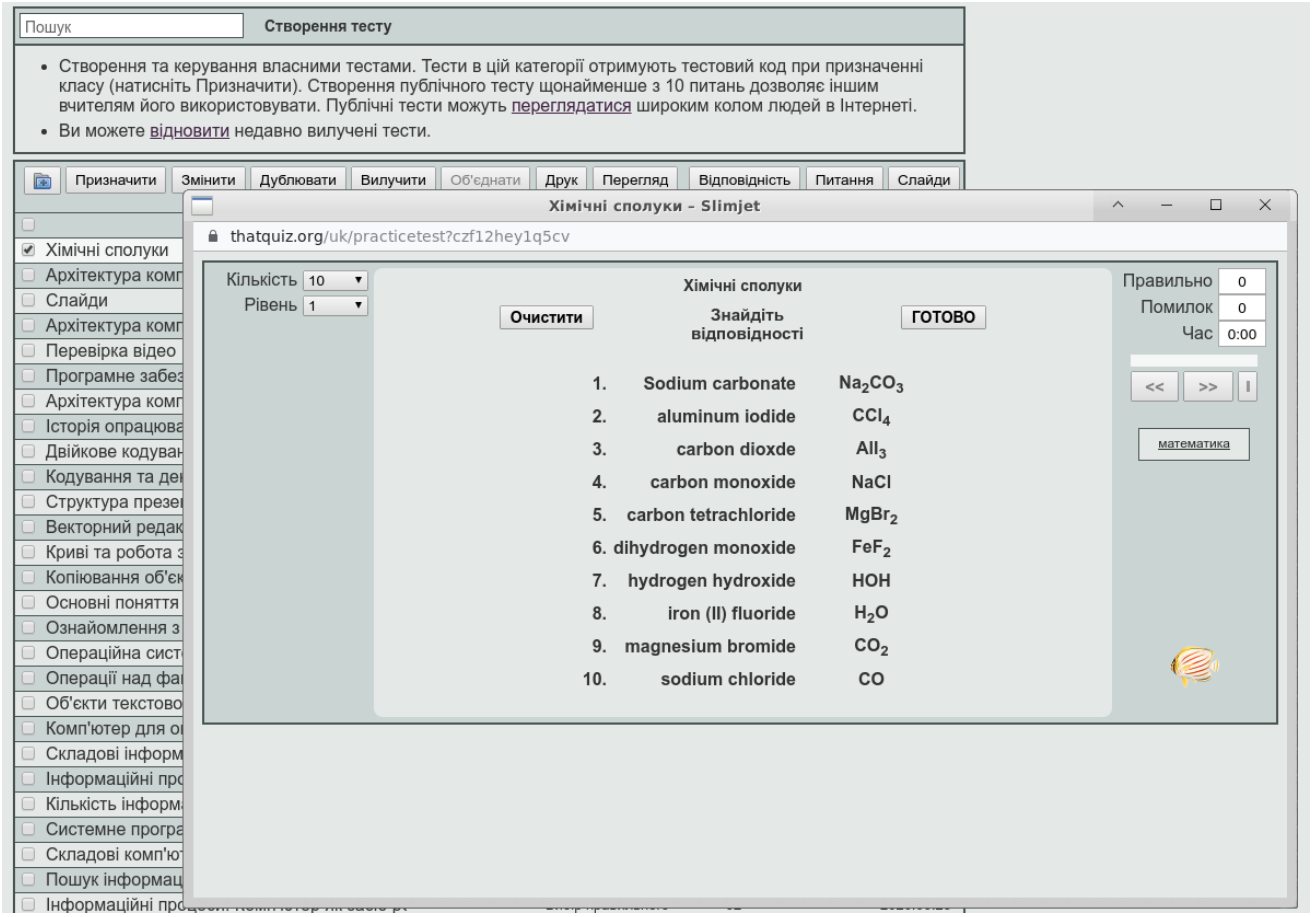


Рисунок 1.11 - Програмний комплекс KTCNet 3

KTCNet 3 — це програмний комплекс для створення, проведення та аналізу комп'ютерних тестів. Він часто використовується в освітніх установах, тренінгових центрах і на підприємствах для оцінки знань і навичок. Основні особливості цієї системи включають:

1. Широкий набір типів запитань: KTCNet 3 підтримує різноманітні типи запитань, такі як:

- Множинний вибір

- Відкрита відповідь

- Встановлення відповідності

- Ранжування (послідовність)

- Зображення з областями для вибору

2. Гнучкість налаштувань тестів: Можливість налаштування тестів включає:

- Обмеження часу на тест або на окремі запитання

- Випадковий порядок запитань і відповідей

- Налаштування кількості спроб

- Визначення критеріїв оцінювання

3. Мультиплатформенність: KTCNet 3 може працювати на різних операційних системах та пристроях, включаючи Windows, macOS, Linux, а також мобільні платформи.

4. Інтерфейс користувача: Зручний і інтуїтивно зрозумілий інтерфейс для як викладачів/адміністраторів, так і для студентів/тестованих. Легке створення, редагування та управління тестами.

5. Безпека тестування: Забезпечення високого рівня безпеки, включаючи шифрування даних, автентифікацію користувачів, контроль доступу, обмеження по IP-адресах і часі доступу.

6. Можливості аналітики та звітності: Платформа пропонує потужні інструменти для аналізу результатів тестування, включаючи детальні звіти про результати, статистичні дані, графіки та діаграми. Це дозволяє викладачам або адміністраторам отримувати всебічний огляд успішності студентів і ефективності тестів.

7. Підтримка інтеграції: Можливість інтеграції з іншими навчальними та управлінськими системами (LMS) через API або інші методи.

8. Керування групами та користувачами: Система дозволяє легко керувати великою кількістю користувачів і груп, надавати різні права доступу, створювати та керувати ролями.

9. Масштабованість: Підтримка великої кількості одночасних користувачів, що робить KTCNet 3 придатною для великих освітніх установ і компаній.

10. Локалізація та підтримка багатьох мов: Підтримка кількох мов, що робить систему доступною для користувачів з різних країн.

KTCNet 3 є потужним інструментом для організації та проведення комп'ютерного тестування, забезпечуючи зручність, гнучкість і високий рівень безпеки.

OpenTEST — це система комп'ютерного тестування знань, яка призначена для створення, проведення та аналізу тестів. Вона використовується в освітніх установах, тренінгових центрах та на підприємствах. Нижче наведено основні особливості та функції системи OpenTEST

Основні особливості OpenTEST:

1. Різноманітність типів запитань:
   - Множинний вибір
   - Відкриті запитання
   - Заповнення пропусків
   - Встановлення відповідності
   - Ранжування (послідовність)
   - Інтерактивні запитання (наприклад, вибір областей на зображенні)
2. Гнучкість налаштувань тестів:
   - Обмеження часу на тест або на окремі запитання
   - Випадковий порядок запитань і відповідей
   - Налаштування кількості спроб
   - Налаштування критеріїв оцінювання

3. Інтуїтивно зрозумілий інтерфейс:

- Зручний інтерфейс для створення та редагування тестів

- Простий інтерфейс для тестованих, що дозволяє легко проходити тестування

4. Автоматичне оцінювання:

- Миттєве оцінювання відповідей

- Надання детальних звітів про результати тестування

5. Безпека тестування:

- Захист тестів паролем

- Обмеження доступу до тестів за IP-адресою або по часу

- Шифрування даних

6. Аналіз результатів:

- Докладні звіти про результати тестування

- Статистика та аналітика для оцінки успішності тестованих

7. Підтримка інтеграції:

- Інтеграція з іншими системами навчання та управління через API

8. Масштабованість:

- Підтримка великої кількості одночасних користувачів

- Придатність для використання в великих освітніх установах та компаніях

OpenTEST забезпечує зручний і ефективний спосіб організації комп'ютерного тестування, пропонуючи потужні інструменти для створення, проведення та аналізу тестів.

Розробки програмних засобів навчання та контролю знань на основі застосування нових інформаційних технологій також ведуться в Національному університеті “Львівська політехніка”, Херсонському державному технічному університеті, Харківському національному університеті радіоелектроніки, Одеському національному політехнічному університеті та в інших ВНЗ України.

1.3. Формалізація складових системи тестування знань

Модель процесу тестування можна представити у вигляді сукупності трьох компонент (рис. 1.12)

- компонента тестування;
- компонента тестів;
- компонента того, хто тестується.

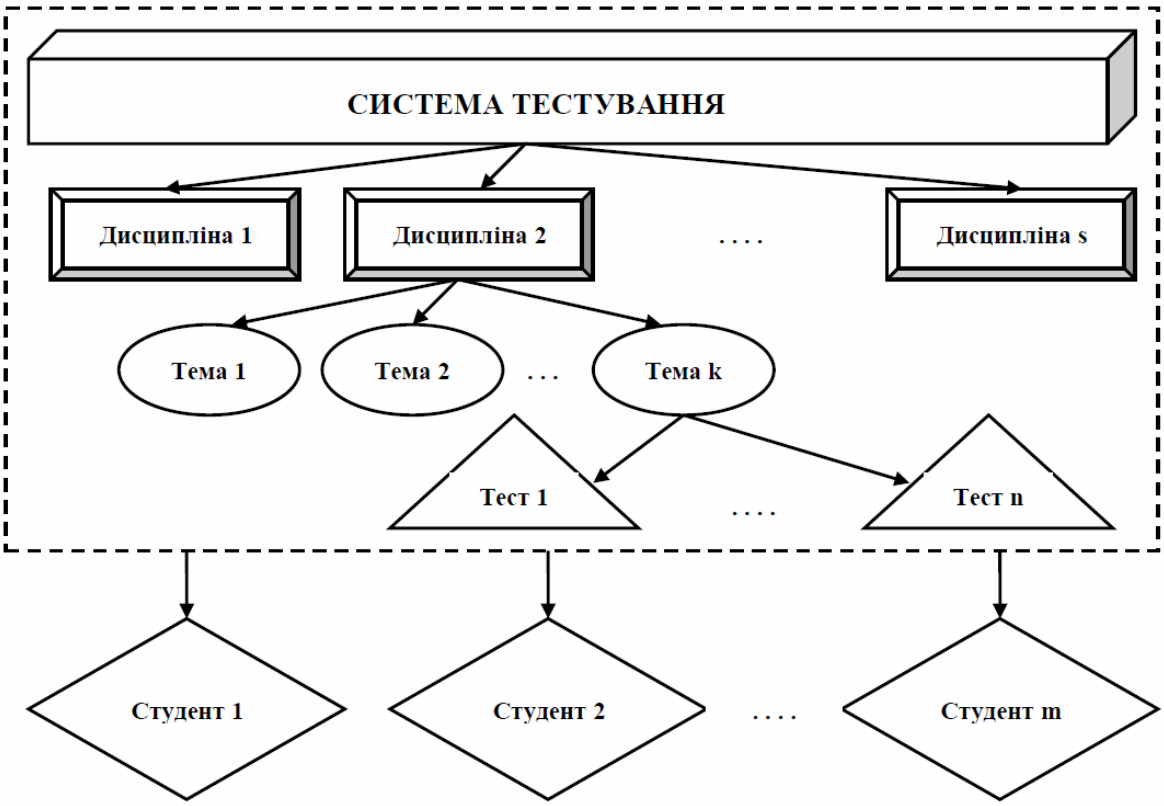


Рисунок 1.11 - Узагальнена модель процесу тестування

Згідно узагальненої моделі процесу тестування (рис. 1.3) вивчення конкретної дисципліни з множини S = {1,2,...,s} припускає освоєння деякої кількості навчальних тем з множини K = {1,2,...,k}.

В результаті покомпонентної розробки навчального матеріалу будь-яка дисципліна може бути представлена N = {1 ,2,...,n} тестовими завданнями по будь-якій темі для студентів, які утворюють множину M = {1,2, ... , m} .

В таблиці 1.1 наведено моделі тестових запитань закритого типу.

Таблиця 1.1 - Моделі тестових запитань

Моделі тестових запитань	Пояснення щодо виконання	Множина варіантів відповідей	Правильна відповідь	Оцінка (бали) +/-	Примітка
Прості вибіркові із множинним вибором	Необхідно вибрати одну відповідь в заданій множині варіантів	$A=\{A_1, A_2, A_3, ..., A_n\}$.	$A_i \subset A$	1/0	
Вибіркові із множинним вибором	Необхідно вибрати відповіді (декілька), які є вірними	$A=\{A_1, A_2, A_3, ..., A_n\}$.	$A_{i1} \cap A_{i3} \cap A_{i2} ... ... \cap A_{in} \cap A_{i(n-1)}$	1/0	Можливий підхід, при якому остаточний результат визначається як частка правильних відповідей
Вибірково-упорядковані	Необхідно вибрати відповіді, які відносяться до вірних відповідей, і розмістити їх у правильній послідовності	$A=\{A_1, A_2, A_3, ..., A_n\}$.	$A_{i1} \cap A_{i2} ... \cap A_{in}$ (тут є важливим порядок відповідей)	1/0	
Перехресні	Необхідно визначити відповідність елементів однієї множини елементам іншої множини	Множина варіантів 1-ї множини $A=\{A_1, A_2, A_3, ..., A_{nt}\}$. Множина варіантів 2-ї множини $B=\{B_1, B_2, B_3, ..., B_{nt}\}$.	Множина пар A_x й B_y	1/0	
Вибірково-об'єднуючі	Необхідно визначити подібні елементи декількох підмножин	Множина варіантів 1-ї множини $A=\{A_1, A_2, A_3, ..., A_{at}\}$. Множина варіантів 2-ї множини $B=\{B_1, B_2, B_3, ..., B_{bt}\}$. ..... Множина варіантів F -ї множини $F=\{F_1, F_2, F_3, ..., F_{ft}\}$.	Множина подібних елементів A_x, B_y і F_z	1/0	При формуванні відповіді може застосовуватися як порівняння елементів підмножин, так й їхнє протиставлення

Типова система тестування знань студентів переважно має модульну структуру та включає в себе наступні компоненти:

1. модуль для створення тестів (конструктор тестів);
2. модуль для проведення сеансів тестування;
3. модуль адміністрування;
4. модуль обробки результатів тестування (статистика);

5. модуль допомоги;

6. модуль перевірки якості створених тестів.

Модульна структура системи забезпечує легкість розширення її функціональності без необхідності внесення змін в існуючі модулі. Система тестування знань студентів дозволяє створювати типи запитань наступних форм:

- запитання закритого типу (вибір однієї відповіді з декількох, вибір декількох відповідей з декількох, вибір зайвої відповіді, альтернативна відповідь);

- запитання на встановлення відповідності (аналогій);

- запитання на встановлення порядку;

- запитання відкритого типу (продовження відповіді, конструювання відповіді).

За допомогою програмного комплексу визначення рівня знань можна вирішувати наступні завдання:

- створювати тести, використовуючи запитання різних типів;

- здійснювати їх налагодження й експорт/імпорт у систему;

- організувати проходження одного тесту необмеженим числом випробуваних;

- організовувати проведення тестування в комп'ютерному класі, які знаходиться в мережі;

- генерувати велику кількість варіантів тестів по шаблону;

- проводити автоматизовану статистичну обробку результатів тестування (індивідуальний та узагальнений аналіз);

- проводити аналіз складених тестів з метою підвищення якості тесту;

- проводити пошук неоднозначних, невдало сформульованих питань.

На рисунку 1.12 приведена структура реляційної бази даних системи тестування, яка дозволяє:

- використовувати й обробляти різні типи тестових завдань;

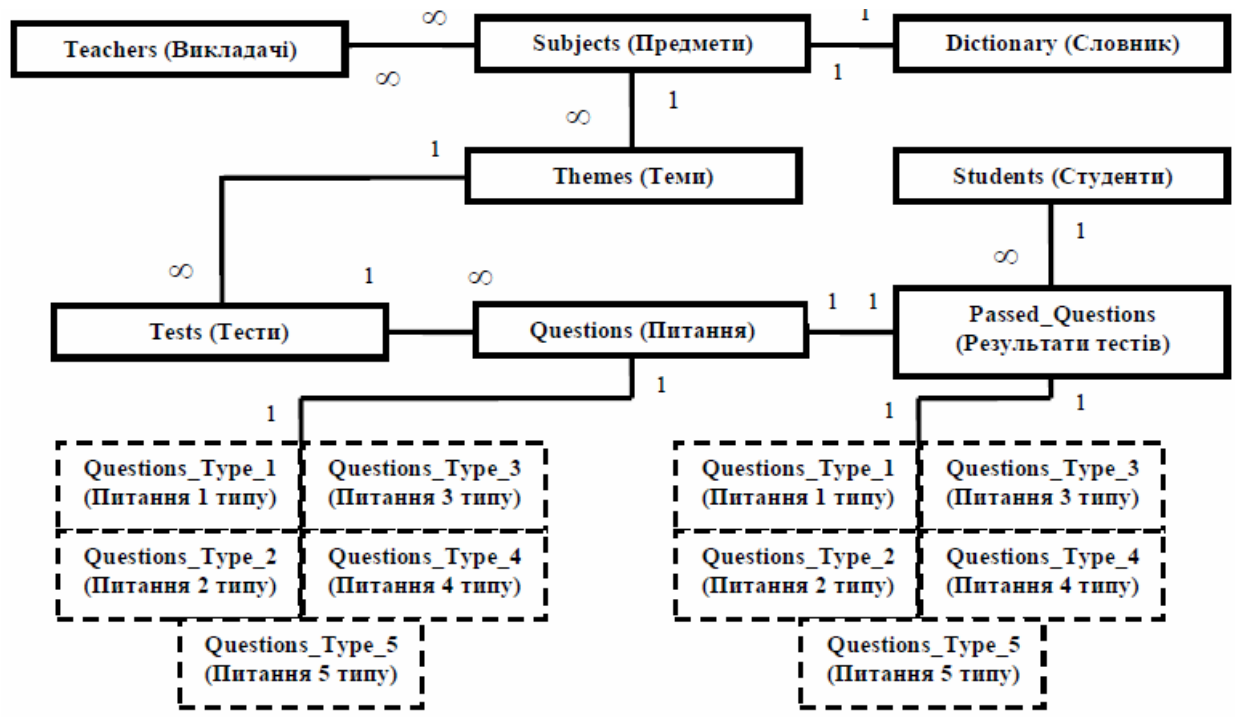


Рисунок 1.12 - Структура бази даних системи тестування

- робити генерацію тестів по заданих шаблонах з датчиком випадкових чисел;

- зберігати результати тестування для наступної обробки

1.4. Організація формування тестових завдань і процесу тестування знань

Організація підготовки тестових завдань складається з двох автономних забезпечуючих підсистем. Це підсистеми підготовки процесу тестування і організації розробки тестів. В рамках підготовки процесу тестування приймається багато організаційних рішень, в тому числі рішення по організаційним, правовим, нормативним питанням з автоматичної ідентифікації студентів. Схеми дій по організації формування тестових завдань наведені на рис. 1.13 і 1.14.

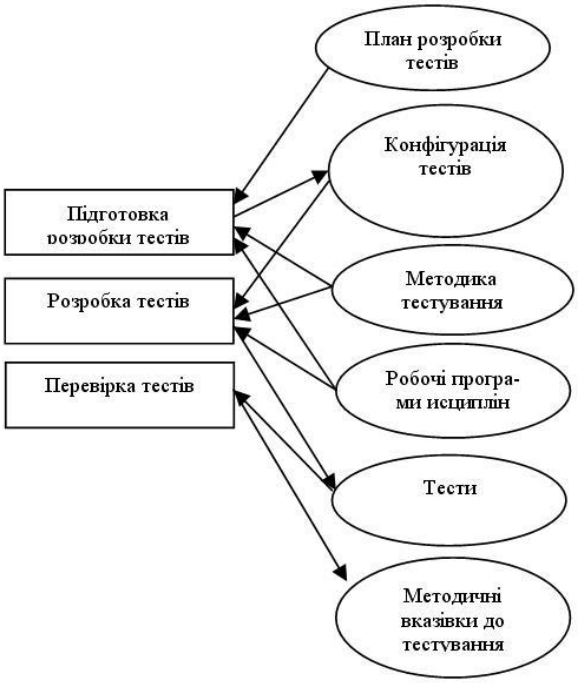


Рисунок 1.13 – Граф-модель організації розробки тестів

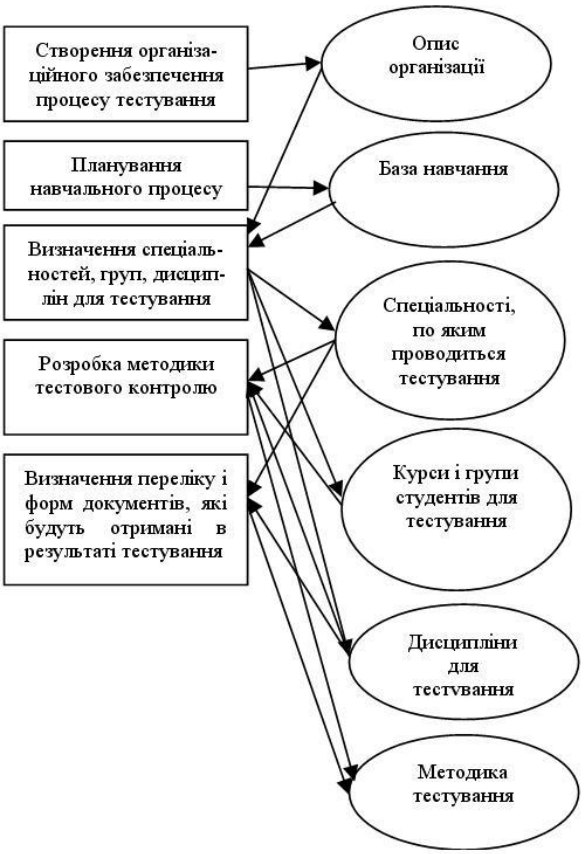


Рисунок 1.14 - Граф - модель підготовки процесу тестування

Схема організації тестування відображає порядок проведення проміжного та підсумкового тестування студентів. Під час тестування необхідно чітко організувати процеси формування розкладу тестування,

підготувати студентів і спеціалізований клас тестування. Тому в рамках організації тестування ведеться планування процесу тестування, затвердження плану та його реалізація. Схема дій по наведена на рис. 1.15.



Рисунок 1.15 - Граф - модель організації тестування

Проведений аналіз існуючих вітчизняних і зарубіжних автоматизованих систем контролю знань показав, що основними недоліками в їх реалізації є значна кількість різних розробок та недостатня увага

питанням організації процесів підготовки тестування, тестування та використання результатів тестування.

Отже, фактори суспільного розвитку, які змінюються, висувають високі вимоги до рівня підготовки фахівців і диктують необхідність змін в освітніх технологіях. Тільки при широкому впровадженні в навчальний процес комп'ютеризованих систем навчання та тестування університети, інститути та академії будуть у змозі вирішувати якісно нові завдання щодо підвищення якості підготовки фахівців на рівні вимог XXI століття.

2. АЛГОРИТМІЧНА РЕАЛІЗАЦІЯ ТА МОДЕЛЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СТУДЕНТО-ЦЕНТРИЧНОГО РІШЕННЯ ТЕСТУВАННЯ ЗНАНЬ

2.1. Розробка діаграм використання для акторів розроблюваної системи

Діаграми використання (use case diagrams) є важливим інструментом в моделюванні програмних систем, особливо для опису взаємодії між користувачами (акторами) і системою. Ось кілька ключових особливостей діаграм використання:

1. Актори:

- Визначення акторів: Актори представляють ролі, які відіграють користувачі або інші системи, що взаємодіють із системою. Це можуть бути люди, зовнішні системи або обладнання.

- Зв'язки акторів: Актори пов'язані з діаграмами використання лініями асоціації, що показують, які актори взаємодіють з якими сценаріями використання.

2. Сценарії використання (Use Cases):

- Опис функціональності: Сценарії використання описують функціональність, яку система надає акторові. Це конкретні завдання або дії, які система виконує.

- Зв'язки між сценаріями: Сценарії використання можуть бути пов'язані один з одним за допомогою відносин включення (include), розширення (extend) та узагальнення (generalization).

3. Взаємодія акторів та сценаріїв:

- Взаємодія: Діаграма показує, як актори взаємодіють з різними сценаріями використання. Це допомагає зрозуміти, які функції системи використовуються різними користувачами.

- Сценарії на високому рівні: Діаграми використання зазвичай зображують систему на високому рівні, показуючи загальні функції без детального опису реалізації.

4. Розширюваність і повторне використання:

- Розширення: Сценарії використання можуть розширювати інші сценарії, додаючи нову поведінку до базового сценарію. Це корисно для створення гнучких і повторно використовуваних моделей.

- Включення: Один сценарій використання може включати інший, що дозволяє уникнути дублювання і забезпечує модульність системи.

5. Візуальне уявлення:

- Простота і зрозумілість: Діаграми використання є простими для розуміння і можуть бути використані як технічними фахівцями, так і нефахівцями.

- Узгодженість: Діаграми забезпечують узгоджене уявлення про функціональність системи, що допомагає в комунікації між різними зацікавленими сторонами проекту.

6. Документування вимог:

- Збір вимог: Діаграми використання часто використовуються для збору і документування вимог до системи.

- Аналіз вимог: Вони допомагають аналізувати вимоги, виявляючи можливі прогалини або суперечності в системних вимогах.

Таким чином, діаграми використання є важливим інструментом для моделювання взаємодії користувачів із системою, забезпечуючи чітке і зрозуміле уявлення про функціональність системи і допомагаючи у зборі та аналізі вимог.

Далі, наведемо діаграму варіантів використання для акторів системи на рисунку 2.1.

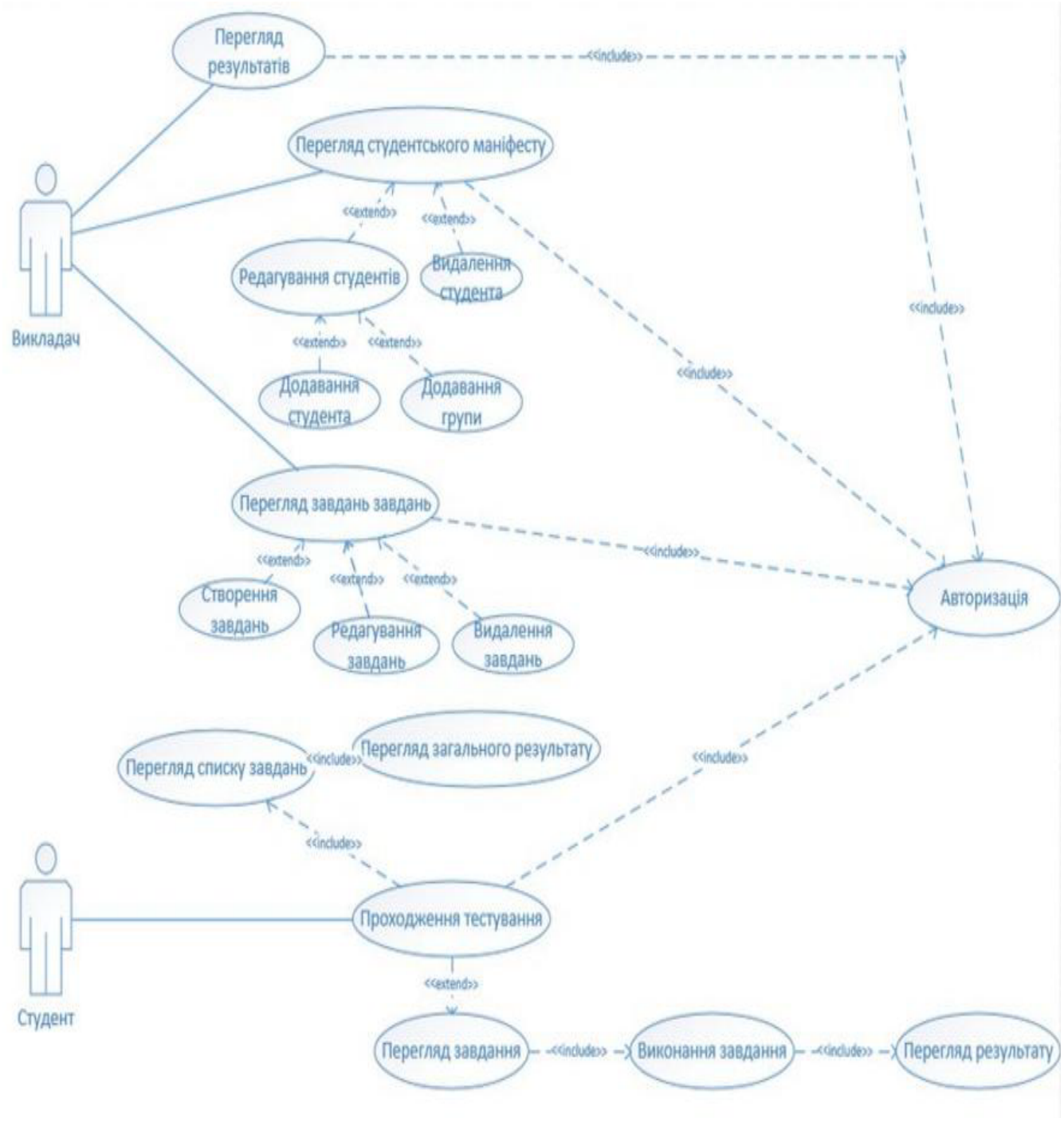


Рисунок 2.1 – Діаграма варіантів використання системи тестування знань студентів

На рисунку 2.2 представлена діаграма варіантів використання студенто-центричного модуля системи тестування знань.



Рисунок 2.2 – Діаграма варіантів використання студенто-центричного модуля системи тестування знань

Варіанти використання представлено нижче на рисунках 2.3 – 2.8.

Передумова	Сценарій	Результат
Запуск додатку (системи)	Користувач вводить дані для авторизації	1. Вхід в систему 2. Можливість проходження тестування

Роль: Студент

Рисунок 2.3 - Варіант використання «Авторизація»

Передумова	Сценарій	Результат
Відкрите вікно перегляду завдання	Користувач переглянувши завдання вводить відповідний запит	Перегляд результату виконання завдання

Роль: Студент

Рисунок 2.4 - Варіант використання «Виконання завдання»



Рисунок 2.5 - Варіант використання «Перегляд результату»



Рисунок 2.6 - Варіант використання «Перегляд списку завдань»



Рисунок 2.7 - Варіант використання «Перегляд завдання»



Рисунок 2.8 - Варіант використання «Перегляд загального результату»

2.2. Розробка прототипів та макетів інтерфейсу користувача

У розробці сучасного програмного забезпечення (ПЗ) ключових фактором виробничого процесу є проектування. Даний процес створення проекту системи часто підрозділяється на 2 окремі частини: проектування функціоналу і інтерфейсу.

Для проектування функціоналу використовуються існуючі нотації UML і IDEF0, є фактично промисловими стандартами при розробці ПЗ. У проектуванні графічного інтерфейсу користувача (GUI) в даний час відсутні усталені стандарти, проте існують окремі рекомендації, закони дизайну, прийоми, традиції, різні умови експлуатації ПЗ. При цьому важливою частиною цього процесу є прототипування або макетування, тобто операція, яка полягає в створенні макета або прототипу майбутньої системи.

Макети в практиці бувають: паперовими, презентаційними, імітаційними та ін. Велика частина сучасних інтегрованих середовищ програмування (IDE) дозволяє розробляти деякі типи макетів, однак це сильно залежить від знань і навичок у використанні конкретної середовища і мови програмування. При цьому в разі створення великого проекту GUI реалізується окремим фахівцем, який не завжди повинен бути фахівцем в області розробки програмного коду. У зв'язку з цим на практиці розробнику

зручно мати гнучкий інструмент для прототипування GUI, що підтримує можливості швидкого створення складних макетів, насичених елементами та об’єктами.

Більшість існуючих програмних продуктів, які застосовуються для вирішення завдань створення GUI, не є спеціалізованими і функціональними інструментами прототипування, однак дозволяють створювати прийнятні макети різного ступеня складності.

Останнім часом у зв’язку з підвищенням вимог до розробки інтерфейсів проявляються тенденції використання спеціалізованого ПЗ, націленого на створення прототипів GUI. Подібні системи підтримують створення прототипів для різних видів ПО: десктопних, веб і мобільних додатків.

Перевагами створення прототипу GUI є:

– можливість побачити і протестувати інтерфейс програми, оцінивши зручність взаємодії його частин без необхідності програмування всього функціоналу, що знижує витрати по зміні GUI (чим швидше інтерфейс буде приведений до підсумкового виду, тим дешевше вийде підсумковий програмний продукт);

– наочна демонстрація можливостей замовнику;

– перевірка зручності використання GUI;

– тестування нестандартних підходів до зовнішнього вигляду інтерфейсу.

Саме прототипування GUI користувача дозволило сформуватися тенденції по відділенню розробки зовнішнього вигляду програми від безпосереднього програмування всього функціоналу. Фактично, це означає, що здійснюється не розробка дизайну інтерфейсу, на базі чого далі йде його реалізація, а паралельна розробка функціональної та графічної складової проекту.

Прототип для функції «Авторизація» зображено на рисунку 2.9.

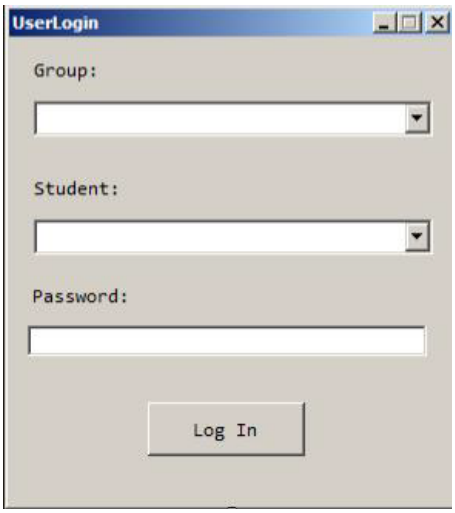


Рисунок 2.9 – Прототип для вікна «Авторизація»

Прототип для функції «Перегляд завдання» представлено на рисунку 2.10.

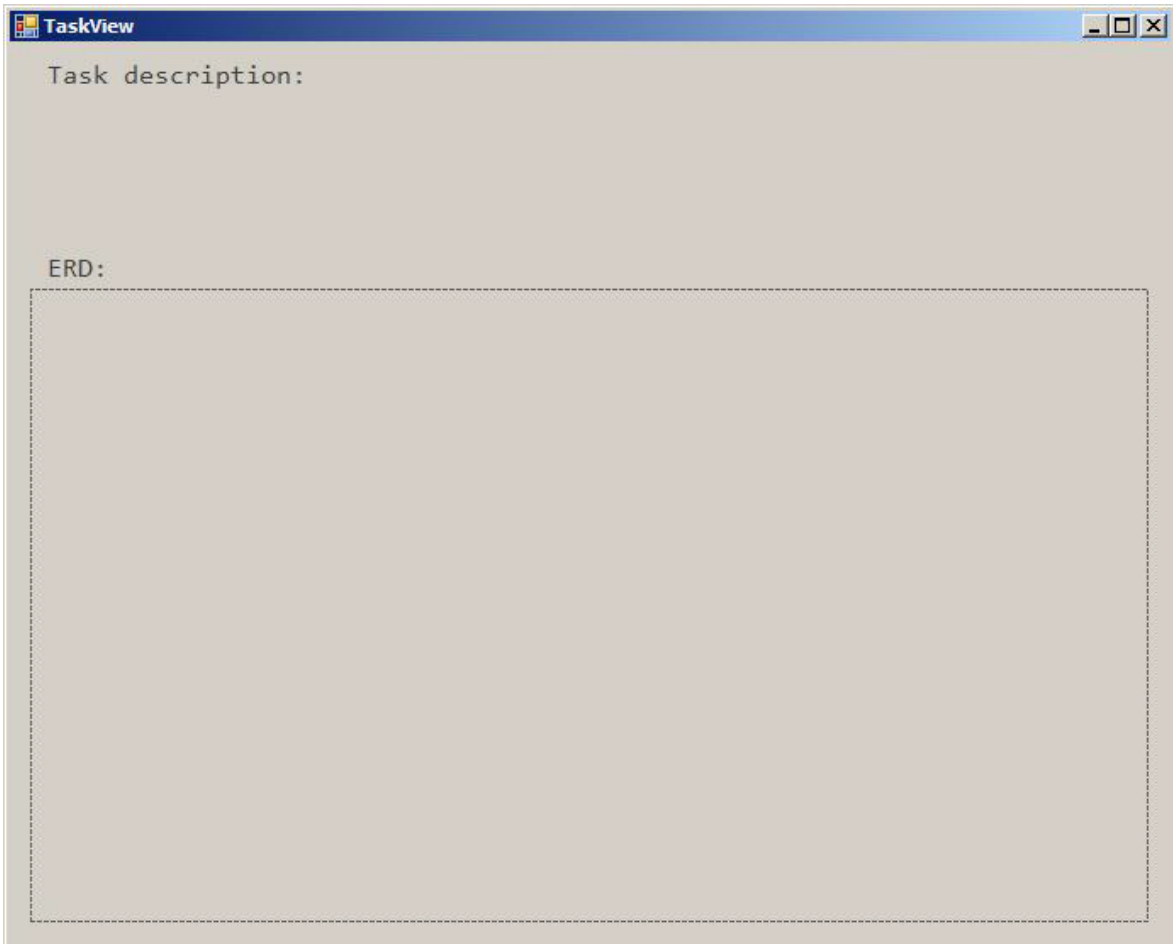


Рисунок 2.10 - Прототип вікна «Перегляд завдання»

Прототип для функції «Виконання завдання» можна побачити на рисунку 2.11.

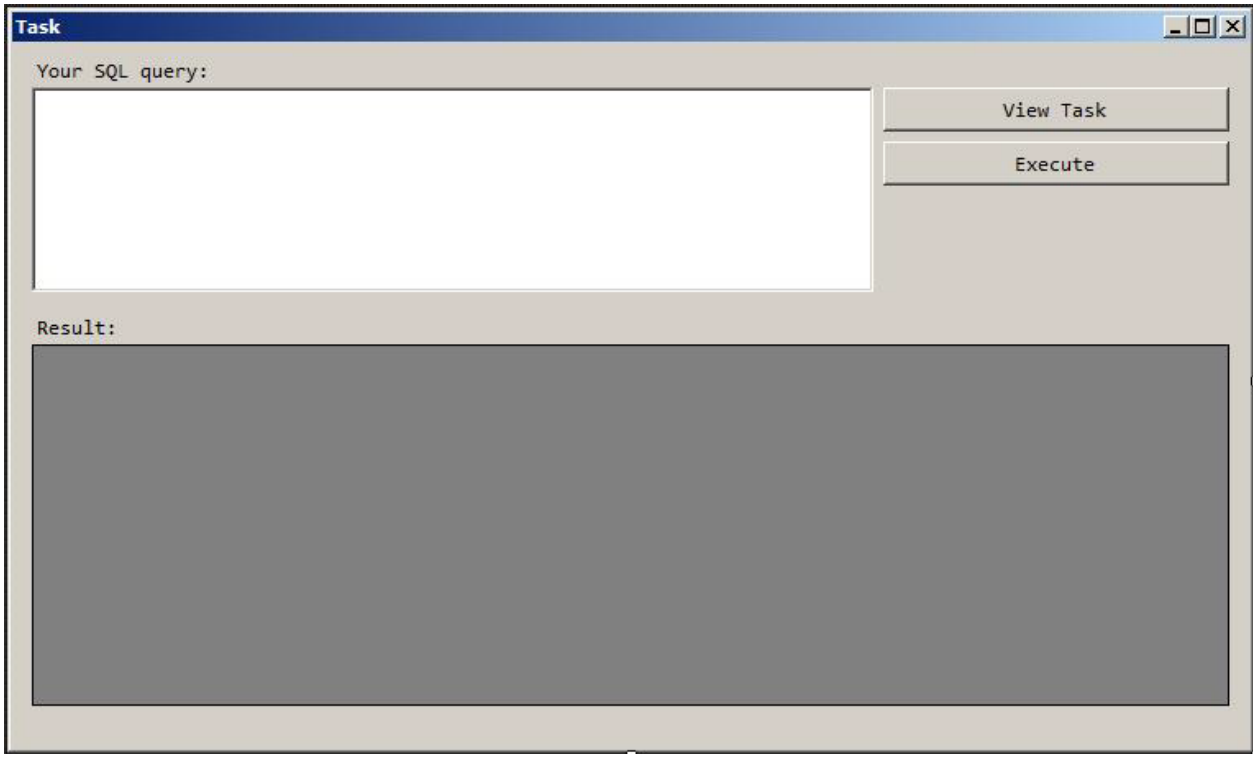


Рисунок 2.11 – Прототип інтерфейсного вікна «Виконання завдання»

Прототип для функцій «Перегляд списку завдань» подано на рисунку 2.12.

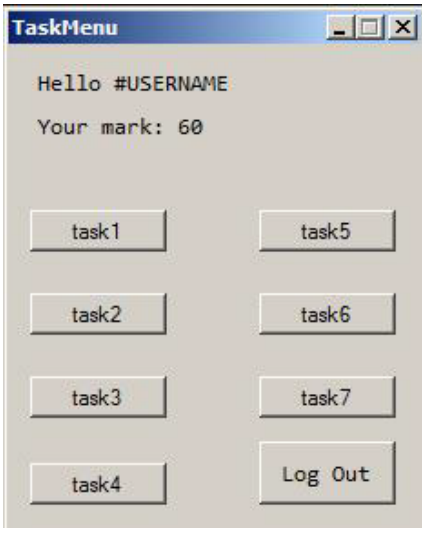


Рисунок 2.12 – Прототип для вікна «Перегляд списку завдань»

Прототип для функції «Перегляд загального результату» представлено на рисунку 2.13.



Рисунок 2.13 – Прототип вікна «Перегляд загального результату»

2.3. Розробка архітектури системи тестування

Для розробки системи було вибрано трьохрівневу архітектуру. Трирівнева архітектура – включає в себе такі компоненти: клієнтський додаток, підключений до сервера і сервер бази даних.

Структурну схему розроблюваної програмної системи тестування подано на рисунку 2.14.

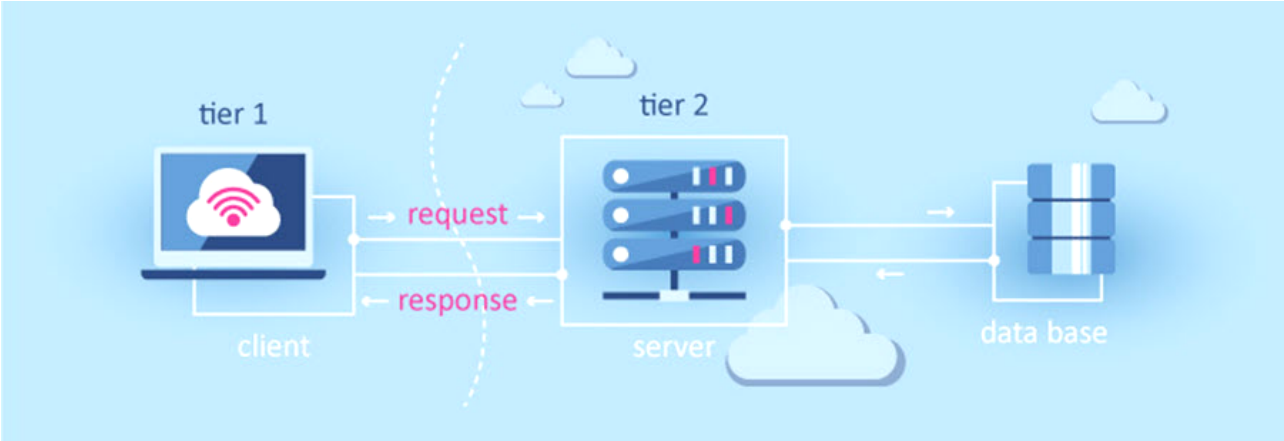


Рисунок 2.14 – Структурна схема системи

Перший рівень - це власне сам додаток для кінцевого користувача. Цей рівень не має прямого зв’язку з базою даних. Тут зосереджений інтерфейс взаємодії користувача з системою.

Другий рівень – рівень сервера. Цей рівень являє собою систему управління базами даних, яка виконує запити додатку в масштабі спеціальної тестової бази даних. Результати таких запитів та можливі помилки надсилаються на перший рівень.

Сервер бази даних забезпечує зберігання даних, необхідних для проходження тестування та містить в собі саму базу даних.

Функціональну структуру розроблюваної системи можна поділити на такі взаємопов’язані модулі:

- модуль обробки даних користувачів:

1) процес авторизації;

- модуль обробки інформації про завдання:

1) перегляд списку завдань;

2) перегляд умов конкретного завдання;

- модуль виконання завдання:

1) виконання відповіді-запиту;

2) порівняння з правильною відповіддю;

3) оцінка результатів;

Для кращого розуміння процесу функціонування системи створено діаграми станів для певних модулів.

Діаграма станів, що представляє процес авторизації користувача подано на рисунку 2.15.

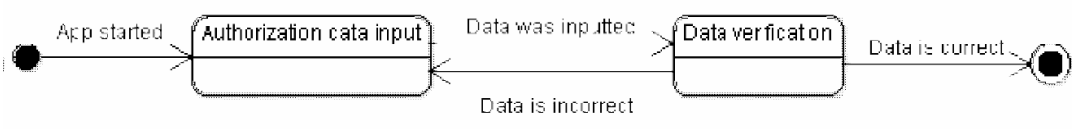


Рисунок 2.15 – Діаграма станів процесу авторизації користувача

Діаграму станів модуля виконання завдання зображено на рисунку 2.16.

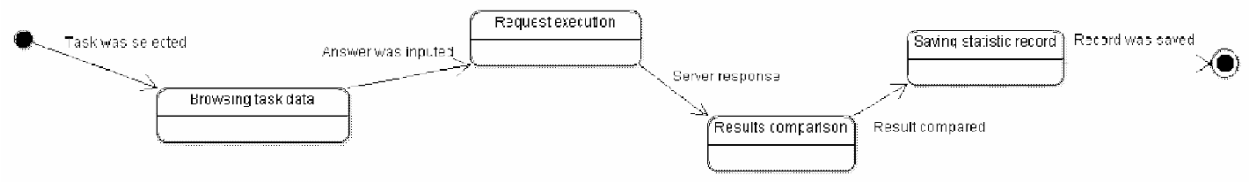


Рисунок 2.16 – Діаграма станів модуля виконання завдання

2.4. Побудова діаграми елементів та UML моделювання

Процес виявлення основних сутностей та зв’язків між ними під час проектування програмного забезпечення є ключовою частиною моделювання та розробки системи. Цей процес включає кілька етапів:

1. Збір вимог:

- Аналіз документації: Вивчення існуючої документації, щоб зрозуміти бізнес-процеси та функціональні вимоги.

2. Аналіз вимог:

- Визначення ключових функцій: Визначення основних функцій та процесів, які система повинна виконувати.

- Ідентифікація сутностей: Виявлення основних об'єктів, які беруть участь у цих процесах. Сутності зазвичай є іменниками, такими як "користувач", "замовлення", "продукт".

3. Моделювання сутностей:

- Опис атрибутів: Визначення атрибутів кожної сутності. Атрибути є властивостями або характеристиками сутності, наприклад, ім'я користувача, ціна продукту.

- Створення діаграм: Використання діаграм класів UML для візуалізації сутностей та їх атрибутів.

4. Виявлення зв’язків між сутностями:

- Аналіз взаємодій: Визначення, як сутності взаємодіють одна з одною в межах бізнес-процесів.

- Встановлення відносин: Визначення типів відносин між сутностями, таких як асоціації, агрегації, композиції та успадкування.

5. Моделювання відносин:

- Створення діаграм: Використання діаграм класів UML для відображення зв’язків між сутностями. Кожен зв’язок може включати кардинальність і ролі.

6 Документування:

- Створення документації: Документування моделі сутностей та зв’язків для подальшого використання в розробці та тестуванні.

- Оновлення документації: Регулярне оновлення документації у разі змін вимог або моделі.

Виконаємо виявлення головних сутностей та зв’язків між ними. З цією метою побудуємо діаграму елементів та зв’язків, що подана на рисунку 2.17.

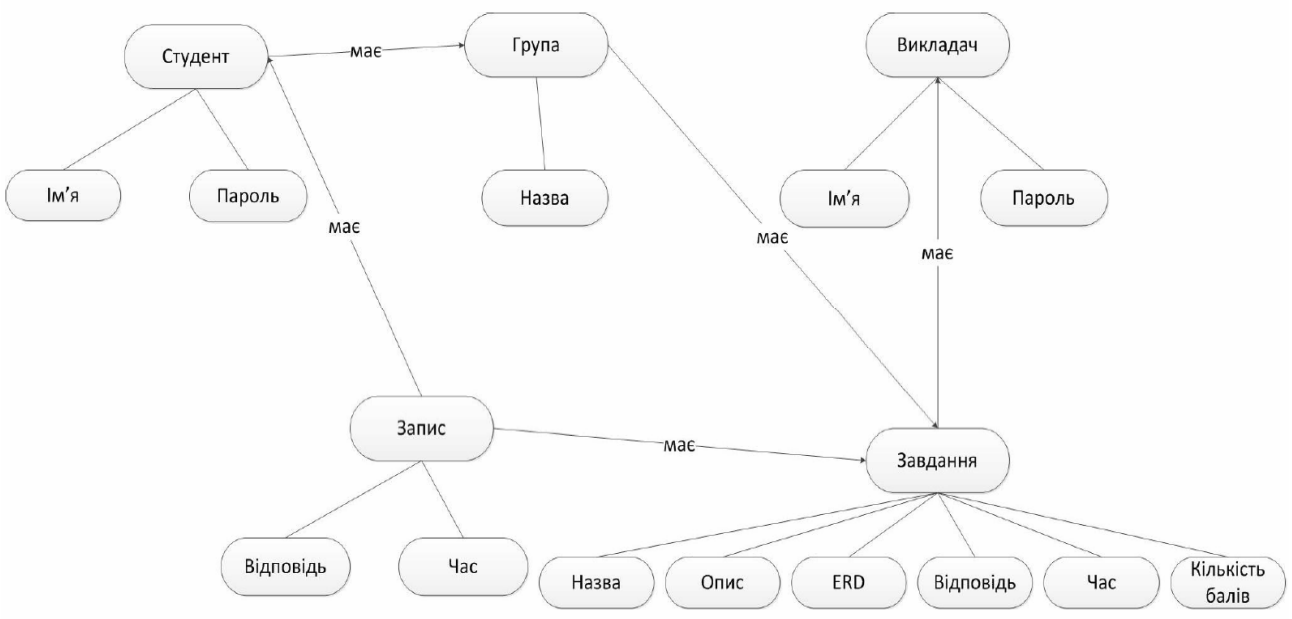


Рисунок 2.17 – Розроблена діаграма елементів та зв’язків

Наступним кроком є побудова таблиці ідентифікаторів яка представлена нижче.

Таблиця 2.1 - Таблиця ідентифікаторів

Об’єкт	Властивість	Тип	Ідентифікатор
Студент	 Ім’я Пароль	 String String	**Student** Name Password
Група	 Назва	 String	**Group** Name
Викладач	 Ім’я Пароль	 String String	**Teacher** Name Password
Запис	 Відповідь Час	 String String	**Record** Answer Time
Завдання	 Назва Опис ERD Відповідь Значення	 String String String String String Float	**Task** Name Description ERD Answer Time Value

Діаграми класів UML (Unified Modeling Language) є однією з найважливіших і найпоширеніших типів діаграм у моделюванні програмних систем. Вони використовуються для опису статичної структури системи, зокрема, класів, їхніх атрибутів, методів і зв'язків між ними. Розглянемо більш детальніше про діаграми класів UML.

Основні компоненти діаграми класів

1. Класи:

- Назва класу: Це ім'я класу, яке зазвичай пишеться у верхній частині блоку.

- Атрибути: Властивості або характеристики класу. Вони пишуться в середній частині блоку.

- Методи (операції): Функції або поведінки класу. Вони пишуться в нижній частині блоку.

2. Зв’язки (відносини):

- Асоціація: Визначає відношення між об'єктами двох класів. Вона може бути односпрямованою або двоспрямованою і зазвичай позначається лінією між класами.

- Агрегація: Особливий вид асоціації, який вказує на "частина-ціле" відношення, де частини можуть існувати незалежно від цілого. Позначається порожнім ромбом на стороні "цілого".

- Композиція: Сильніший вид агрегації, де частини не можуть існувати незалежно від цілого. Позначається заповненим ромбом на стороні "цілого".

- Успадкування (генералізація): Вказує на відношення між базовим класом і похідним класом, де похідний клас успадковує атрибути і методи базового класу. Позначається стрілкою з порожньою головкою, що вказує на базовий клас.

- Реалізація: Відношення між класом і інтерфейсом, який цей клас реалізує. Позначається пунктирною стрілкою з порожньою головкою.

Основні елементи діаграми класів:

1. Клас:

- Прямокутник, який містить три частини: назву, атрибути і методи.

2. Інтерфейс:

- Подібний до класу, але зазвичай позначається стереотипом <<interface>> і містить тільки методи.

3. Відносини:

- Лінії з різними позначеннями для відображення типів зв'язків між класами.

Приклади використання діаграми класів

1. Проектування системи:

- Використовується для визначення основної структури системи, включаючи основні класи і їх взаємозв'язки.

2. Документування системи:

- Служить для документування існуючої системи, надаючи візуальне представлення її структури.

3. Аналіз вимог:

- Допомагає виявити і деталізувати вимоги до системи, показуючи, які класи і відносини необхідні для реалізації функціональності.

4. Підтримка розробки:

- Сприяє кращому розумінню системи розробниками, полегшуючи процес кодування і тестування.

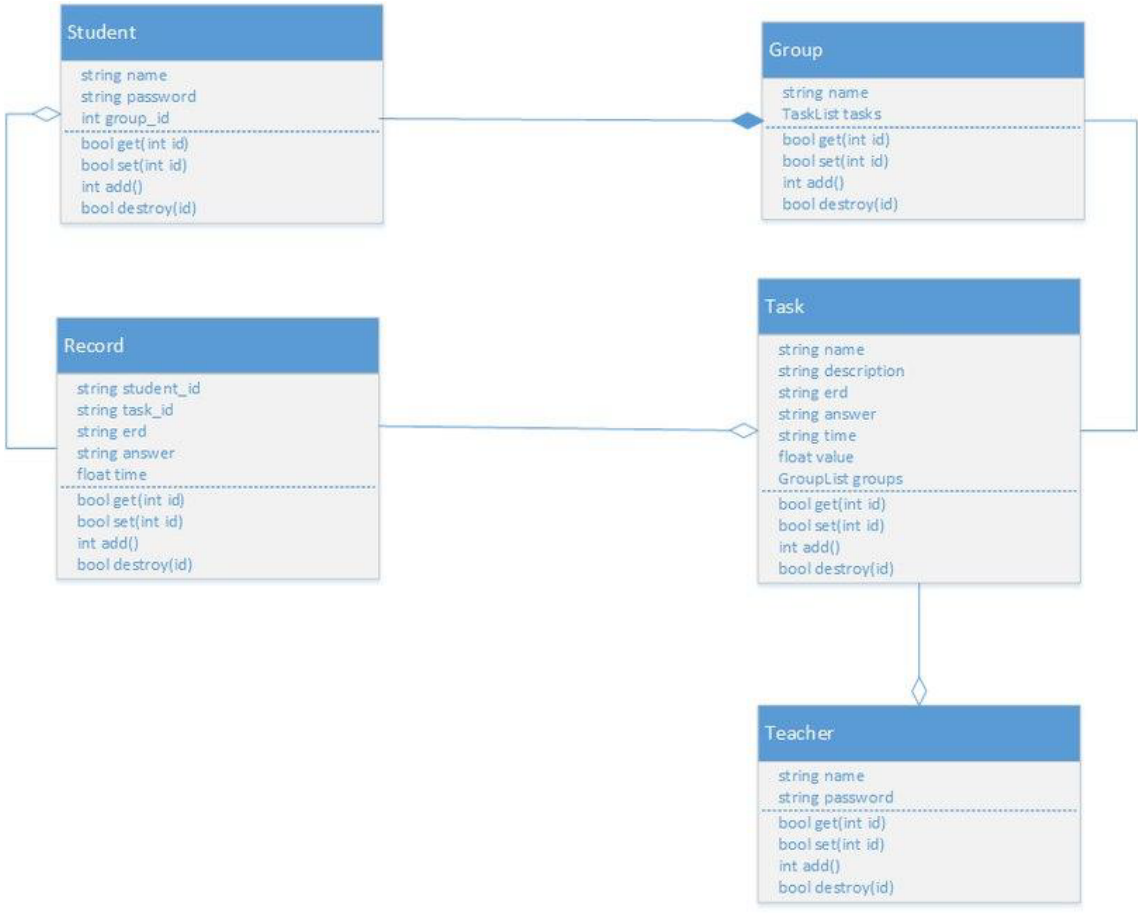


Рисунок 2.18 – Діаграма класів UML

Наступним кроком є створення діаграми класів UML, що показано на рис. 2.18.

Діаграми класів UML є потужним інструментом для моделювання об'єктно-орієнтованих систем. Вони допомагають розробникам і архітекторам візуалізувати, аналізувати і документувати структуру системи, що значно полегшує процес розробки та обслуговування програмного забезпечення.

2.5. Вибір та опис інструментів для розробки

Було обрано C# як мови програмування для цього проєкту через її численні переваги. Як сучасна, універсальна та потужна мова, C# пропонує потужну типізацію та широкий вибір бібліотек. Екосистема .NET, яка забезпечує підтримку та масштабованість, гарантує, що C# є ідеальним вибором для створення надійних, ефективних та підтримуваних додатків.

Більше того, репутація C# як зручної для читання мови ще більше зміцнює її придатність для проєктів, які можуть потребувати постійних оновлень та вдосконалень.

Ця мова програмування характеризується строгою типізацією та пріоретизує створення безпечних та надійних додатків. За допомогою C# можна розробити додатків для Windows, веб та розподілених, клієнт-серверних додатків чи та додатків роботи з базами.

Мова програмування C# є однією з популярних мов для розробки програмного забезпечення завдяки своїм численним перевагам. Ось основні з них:

1. Об'єктно-орієнтована парадигма:

- C# підтримує об'єктно-орієнтоване програмування (ООП), що дозволяє розробникам створювати модульний, масштабований і багаторазово використовуваний код.

2. Потужний і гнучкий синтаксис:

- Синтаксис C# є чистим і чітким, що спрощує читання і написання коду, зменшуючи кількість помилок.

3. Підтримка платформи .NET:

- C# є основною мовою для платформи .NET, яка надає розробникам доступ до великого набору бібліотек і інструментів для створення різноманітних типів додатків, включаючи веб-додатки, десктопні додатки, мобільні додатки та сервіси.

4. Кросплатформеність з .NET Core і .NET 5/6:

- Завдяки .NET Core і його наступникам (.NET 5/6), розробка на C# стала кросплатформенною, дозволяючи створювати додатки для Windows, macOS і Linux.

5. Безпека типів:

- C# забезпечує строгий контроль типів, що допомагає уникнути багатьох типів помилок, пов'язаних з неправильним використанням даних.

6. Сильна підтримка асинхронного програмування:

- C# має потужні механізми для асинхронного програмування, такі як async/await, що спрощують розробку високопродуктивних і відповідних додатків.

7. Автоматичне управління пам'яттю:

- C# використовує збирач сміття (Garbage Collector), який автоматично керує виділенням і звільненням пам'яті, що знижує ризик виникнення помилок управління пам'яттю.

8. Розширюваність і масштабованість:

- Завдяки підтримці компонентної архітектури і можливостям інтеграції з іншими технологіями, C# дозволяє створювати масштабовані та розширювані системи.

9. Сильна підтримка інструментів:

- Існує широкий спектр інструментів для розробки на C#, включаючи Visual Studio, яка надає розширені можливості для налагодження, тестування і профілювання коду.

10. Активна спільнота і велика кількість ресурсів:

- C# має велику і активну спільноту розробників, що надає багато ресурсів, таких як документація, форуми, навчальні матеріали і приклади коду.

11. Підтримка сучасних стандартів і технологій:

- C# постійно оновлюється, включаючи нові функції і можливості, що дозволяє розробникам використовувати сучасні стандарти і передові технології в своїх проектах.

12. Інтеграція з іншими мовами і платформами:

- C# дозволяє легко інтегруватися з іншими мовами програмування та платформами, що розширює можливості використання існуючого коду.

Ці переваги роблять C# потужною мовою для розробки програмного забезпечення, яка підходить для широкого спектру застосувань і дозволяє розробникам створювати високоякісні, продуктивні та надійні додатки.

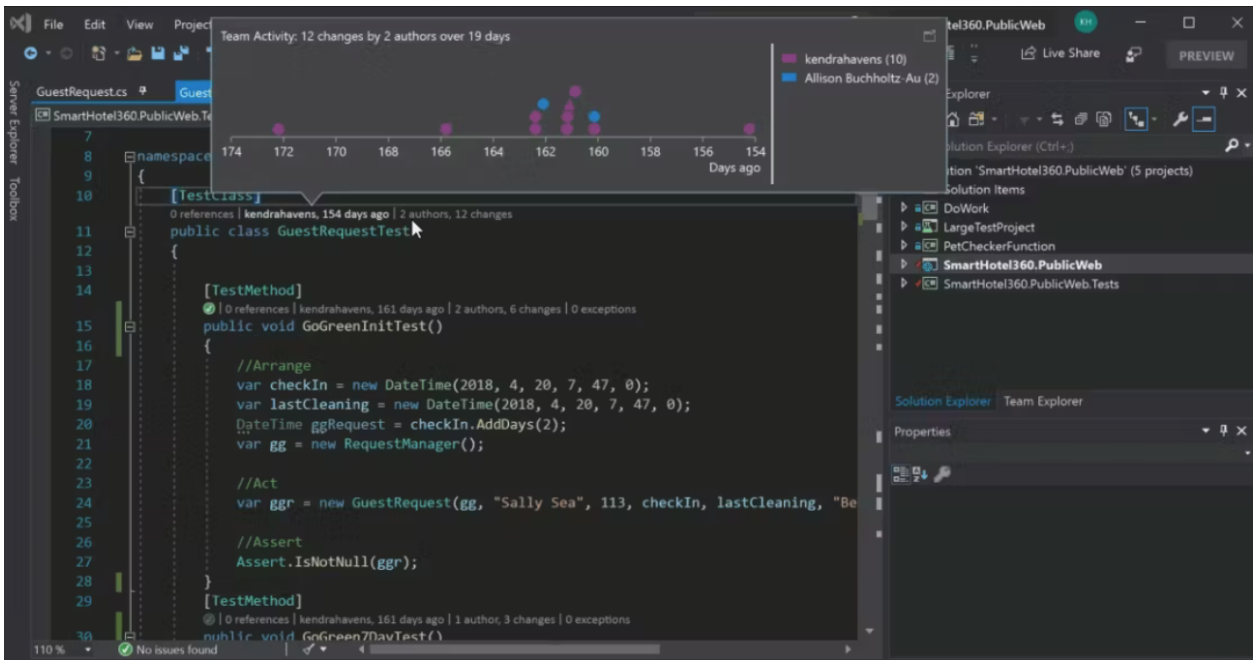


Рисунок 2.19 - Microsoft Visual Studio

Microsoft Visual Studio — це інтегроване середовище розробки (Integrated Development Environment, IDE), розроблене компанією Microsoft. Воно надає розробникам потужні інструменти для створення різноманітних типів додатків, включаючи веб-додатки, десктопні додатки, мобільні додатки, сервіси і багато інших

Середовищем розробки виступає Microsoft Visual Studio, яке містить просунутий редактор коду, інструменти розробки користуватського інтерфейсу, вбудований відлагоджувач та безліч інших інструментів та функцій для полегшення процесу розробки.

Отже, в пропонованому розділі було спроектовано систему тестування студентів, проведено аналіз алгоритмічної сторони виконання завдання дипломної роботи, виконано UML моделювання. Також виконана розробка прототипів та макетів інтерфейсу користувача системи тестування знань студентів.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ СТУДЕНТО-ЦЕНТРИЧНОГО РІШЕННЯ ТЕСТУВАННЯ ЗНАНЬ

3.1. Розробка діаграм потоків даних

DFD-модель (Data Flow Diagram) або діаграма потоків даних є корисним інструментом під час проектування бази даних. Ось деякі основні переваги її використання:

1. Візуалізація потоків даних:

- Простота розуміння: DFD-модель дозволяє візуалізувати, як дані переміщуються через систему, що робить процеси і взаємодії зрозумілішими як для технічних, так і для нетехнічних користувачів.

- Чітке представлення потоків: Вона чітко показує, звідки надходять дані, як вони обробляються і куди вони йдуть, що допомагає краще зрозуміти бізнес-процеси і їхні вимоги.

2. Документування вимог:

- Аналіз вимог: DFD-моделі допомагають у зборі та аналізі вимог до системи, забезпечуючи зрозуміле уявлення про те, які дані необхідні і як вони використовуються.

- Виявлення прогалин: Вони допомагають виявити можливі прогалини або суперечності у вимогах, що дозволяє їх своєчасно виправити.

3. Підтримка структури бази даних:

- Визначення сутностей і атрибутів: DFD-моделі допомагають ідентифікувати сутності, атрибути та їхні взаємозв'язки для проектування БД.

- Планування нормалізації: Вони сприяють виявленню можливих проблем з повторенням даних і допомагають спланувати нормалізацію БД.

4. Полегшення комунікації:

- Спільна мова: DFD-моделі надають спільну мову для розробників, аналітиків, користувачів та інших зацікавлених сторін.

- Візуальне представлення: Візуальне представлення процесів і даних сприяє кращому розумінню складних систем.

5. Підтримка проектування процесів:

- Оптимізація процесів: DFD-моделі допомагають виявити і оптимізувати процеси, що призводить до підвищення ефективності системи.

- Виявлення вузьких місць: Вони дозволяють виявити вузькі місця і потенційні проблеми в потоках даних, що дозволяє їх вирішити на ранніх етапах проектування.

6. Гнучкість і масштабованість:

- Модульність: DFD-моделі можна розділити на рівні, що дозволяє моделювати як високорівневі, так і деталізовані процеси.

- Масштабованість: Вони легко масштабуються і можуть використовуватися для моделювання як малих, так і великих систем.

Таким чином, використання DFD-моделей під час проектування бази даних допомагає забезпечити більш структурований, зрозумілий і ефективний підхід до розробки, що знижує ризики помилок і підвищує якість кінцевої системи.

Створена DFD-модель предметної області подана на рис. 3.1.

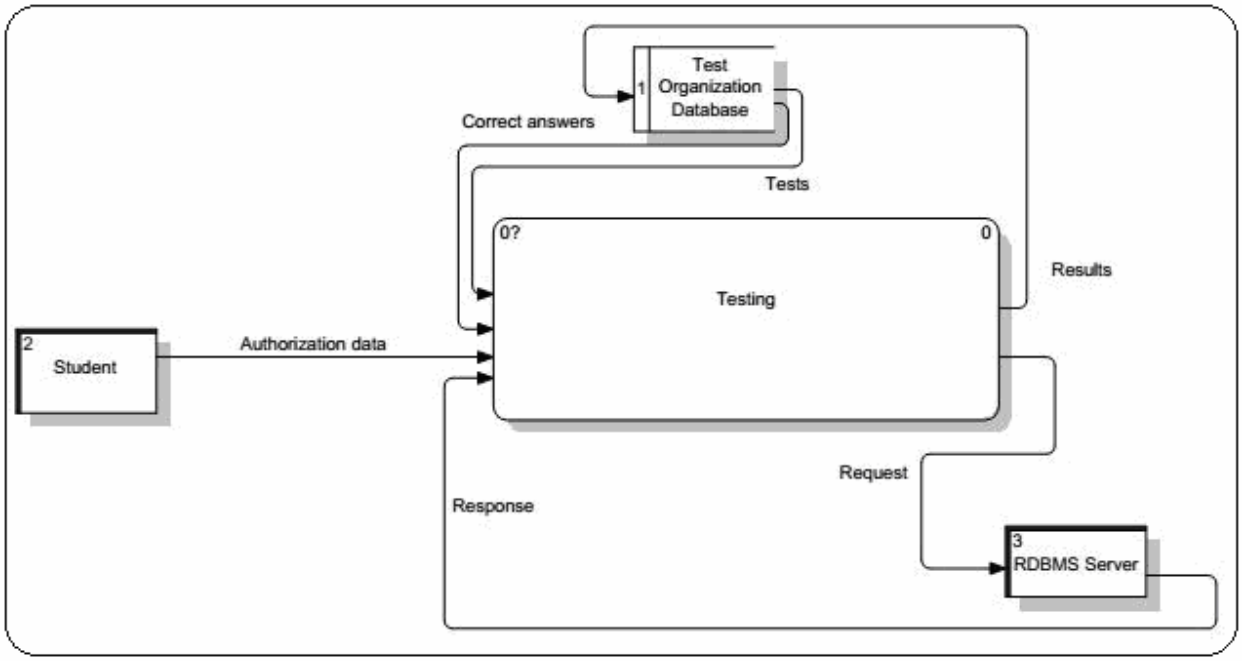


Рисунок 3.1 – Діаграма потоків даних

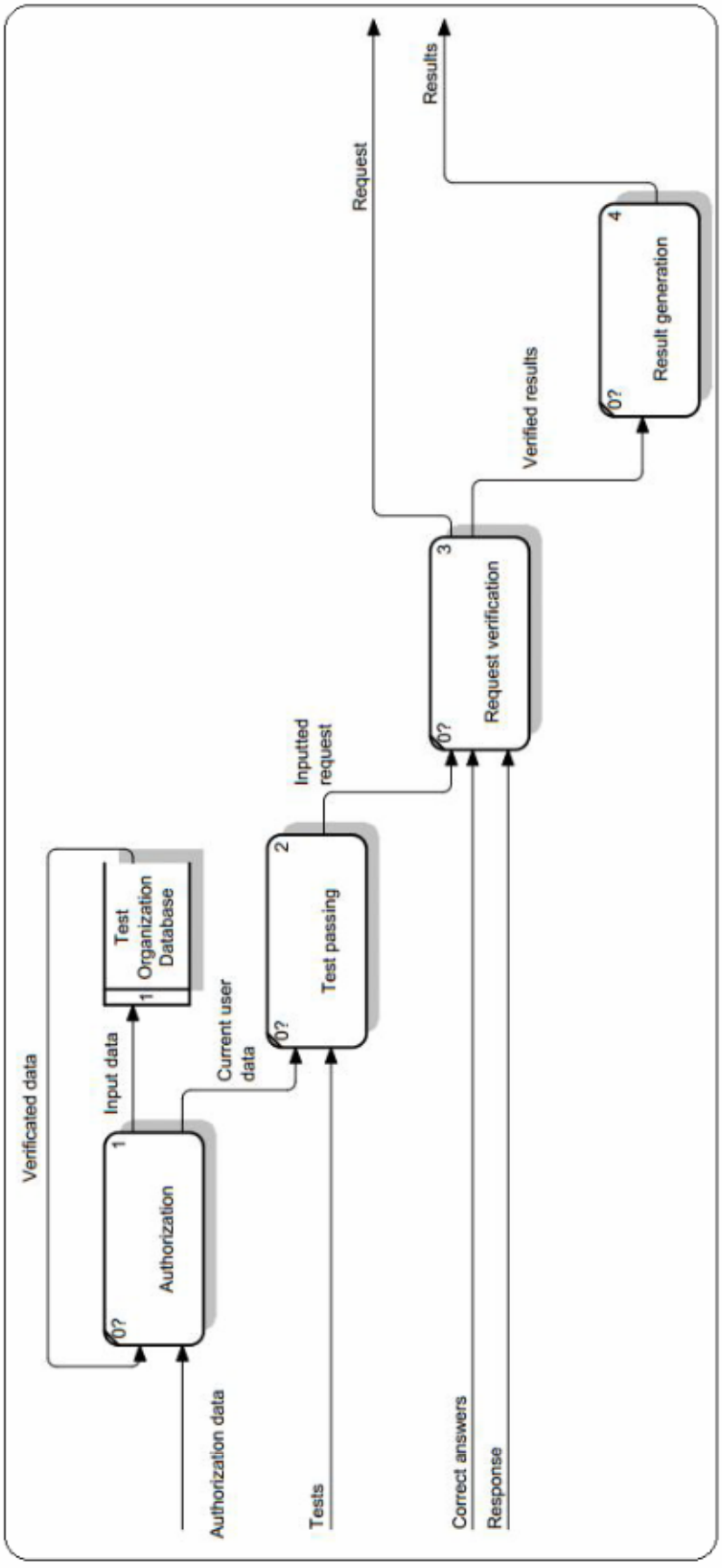


Рисунок 3.2 – Діаграма декомпозиції потоків даних

Як видно з рисунку 3.2 робота, системи тестування виконується наступним чином :

1. Вхідні потоки переробляються системою у вихідні величини:

- Authorization data (авторизаційні дані);
- Tests (інформація про завдання);
- Correct answers (правильні відповіді);
- Response (відповідь СУБД на запит).

2. Вихідні потоки:

- Result (результат);
- Request (запит до СУБД);

Процес проходження тестування відбувається за допомогою студента, який взаємодіє з програмною системою. Студент авторизується, отримує список завдань та вибирає одне з них Після цього він має змогу переглянути умову завдання та приступити до його виконання. Після введення відповіді та підтвердження, відповідь-запит надсилається на виконання до СУБД, відповідь на який аналізується та порівнюється з правильними відповідями. Після отримання результату перевірки студент приступає до виконання інших завдань, у разі негативного результату студент може спробувати виконати завдання знову.

3.2. Моделювання структури бази даних з використанням ER-діаграм

Найбільше поширеним засобом моделювання даних є діаґрами “сутність-взаємозв'язок” (ERD). Моделювання бази даних з використанням діаграми "сутність-взаємозв'язок" (ERD, Entity-Relationship Diagram) є одним із ключових етапів у проектуванні баз даних. ERD допомагає візуалізувати структуру бази даних, визначити її сутності, атрибути і взаємозв'язки між ними. Розглянемо деякі особливості цього підходу.

Основні компоненти ERD:

1. Сутності (Entities):

- Визначення сутностей: Сутності представляють об'єкти або поняття, що зберігаються в базі даних. Наприклад, "Користувач", "Продукт", "Замовлення".

- Атрибути сутностей: Властивості або характеристики сутностей, такі як ім'я, адреса, ціна. Ключові атрибути (Primary Keys) однозначно ідентифікують кожну сутність.

2. Зв'язки (Relationships):

- Визначення зв'язків: Зв'язки відображають відносини між сутностями, наприклад, "Користувач створює Замовлення".

- Кардинальність зв'язків: Визначає кількісні обмеження на зв'язки, наприклад, один-до-одного (1:1), один-до-багатьох (1:N), багато-до-багатьох (M:N).

Основні особливості ERD:

1. Візуальне представлення:

- Чіткість і простота: ERD надає чітке та зрозуміле представлення структури бази даних, що допомагає виявити основні сутності та їхні взаємозв'язки.

- Легкість для розуміння: ERD легко зрозуміти як технічним фахівцям, так і нефахівцям, що полегшує комунікацію між розробниками і замовниками.

2. Аналіз і проектування:

- Збір вимог: Допомагає зібрати та проаналізувати вимоги до даних на ранніх етапах проекту.

- Визначення структури даних: Визначає структуру даних, необхідну для підтримки бізнес-процесів, що знижує ризики помилок у проектуванні.

3. Кардинальність і обмеження:

- Визначення кардинальності: Чітко визначає кількісні обмеження на зв'язки, що допомагає в моделюванні реальних сценаріїв використання даних.

- Обмеження цілісності: ERD дозволяє визначити обмеження цілісності даних, такі як первинні і зовнішні ключі.

4. Гнучкість і масштабованість:

- Модульність: ERD можна легко розбити на менші піддіаграми, що полегшує роботу з великими і складними системами.

- Масштабованість: ERD підтримує розширення і модифікації, що дозволяє легко оновлювати модель даних при зміні вимог.

5. Підтримка нормалізації:

- Запобігання аномаліям: Допомагає в нормалізації бази даних, що мінімізує дублювання даних і запобігає аномаліям при вставці, оновленні та видаленні даних.

- Оптимізація структури: Підтримує оптимізацію структури даних для підвищення продуктивності і ефективності роботи бази даних.

Моделювання бази даних з використанням ERD забезпечує структурований та логічний підхід до проектування баз даних. Це допомагає розробникам створити ефективну і цілісну модель даних, яка відповідає вимогам бізнесу і підтримує необхідні функціональні можливості системи.

Тому на основі аналізу предметної області виконано проектування моделі бази даних засобами нотацій ERD діаграми (рисунок 3.3).

Для створення таблиць було обрано Microsoft SQL Server Management Studio 2014. Цей програмний засіб по замовчуванню постачається разом з СУБД Microsoft SQL Server 2014 та є рекомендованим Microsoft ПЗ для взаємодії з нею. Вибір середовищ Microsoft Visual Studio, Microsoft та SQL Server Management Studio дозволяє підтримувати хороший рівень сумісності БД оскільки всі програмні засоби були розроблені однією компанією з використанням спільних стандартів.

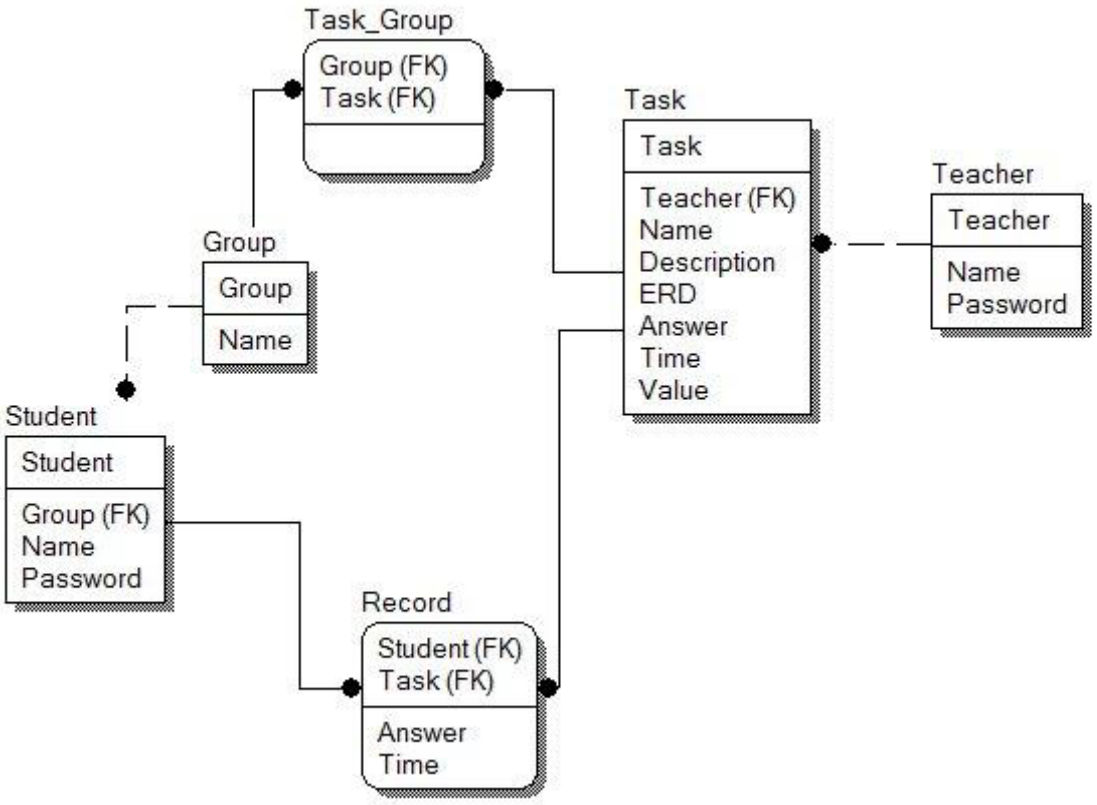


Рисунок 3.3 – ERD-діаграма

Вікно створення бази даних подано на рисунку 3.4.

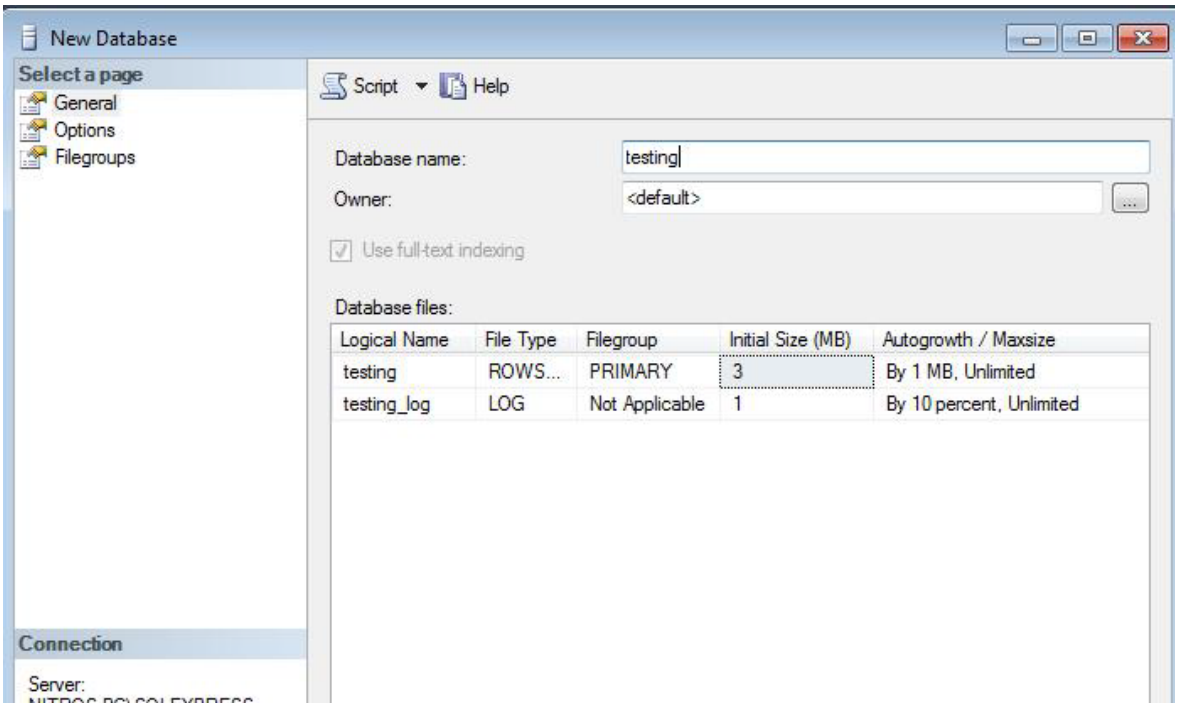


Рисунок 3.4 – Вікно створення бази даних засобами Microsoft SQL Server

Наступним кроком буде створення таблиці Group. Для цього запустимо редактор скриптів. На рисунку 3.5 зображено вікно редактора скриптів.

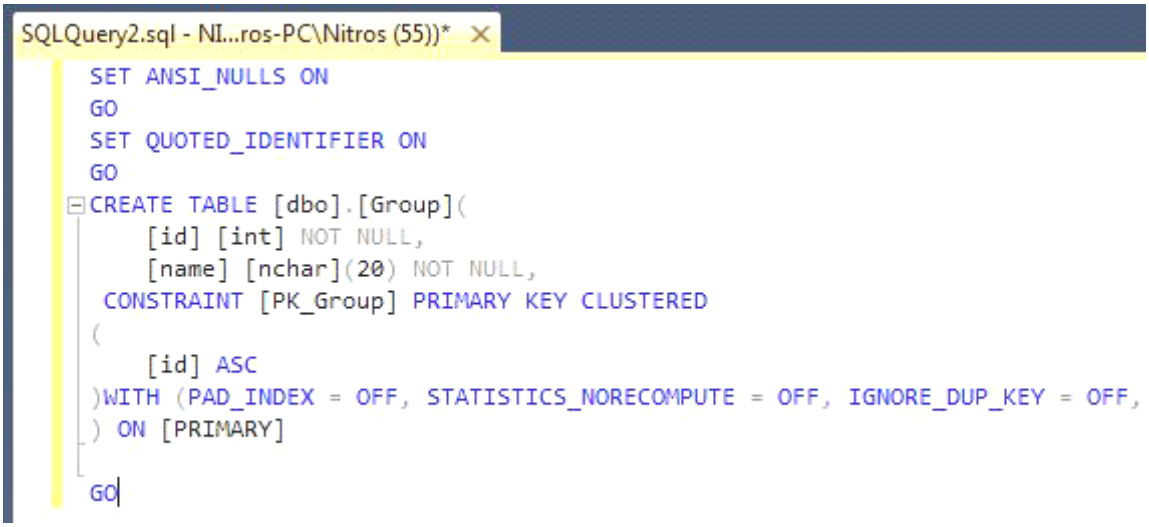


Рисунок 3.5 – Створення таблиці Group за допомогою вікно редактора скриптів

Тепер створимо таблицю виконаємо створення таблиці Student за допомогою SQL-скрипта, що подано нижче.

Лістинг 3.1. SQL-скрипт створення таблиці Student

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Student](
[id] [int] NOT NULL,
[name] [nchar](20) NOT NULL,
[password] [nchar](16) NOT NULL,
[group_id] [int] NOT NULL,
CONSTRAINT [PK_Student] PRIMARY KEY CLUSTERED
([id] ASC)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE =
OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
[PRIMARY]
) ON [PRIMARY]
GO
```

Далі створимо таблицю Record. Буде використано наступний SQL-скрипт (лістинг 3.2).

Лістинг 3.2. SQL-скрипт створення таблиці Record

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Record](
[id] [int] NOT NULL,
[asnwer] [nchar](50) NULL,
[time] [nchar](10) NULL,
[student_id] [int] NOT NULL,
[task_id] [int] NOT NULL,
CONSTRAINT [PK_Record] PRIMARY KEY CLUSTERED
([id] ASC)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE =
OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
[PRIMARY]
) ON [PRIMARY]
GO
```

3.3. Програмна реалізація системи тестування

Розроблювана система проведення тестування знань складається з двох взаємопов'язаних модулів: модуля "Студент" та модуля "Викладач", які пов'язані між собою засобами СУБД. Кожен модуль можна розглядати як додаток з клієнт-серверною архітектурою, оскільки він побудований на основі постійного діалогу з СУБД.

Завдання модуля "Студент" полягає в ідентифікації студента, виданню приписаних до його групи завдань для виконання, надання можливості приймати спроби з їх проходження, їх оцінювання та збереження результатів.

Програмний модуль було створено засобами мови C# платформи .Net. Вхідними даними для розроблюваного модуля є список груп, студентів, завдань. Джерелом всіх цих даних є база даних, з метою безпеки паролі студентів та правильні відповіді зберігаються в БД в зашифрованому алгоритмом SHA1 вигляді. Вихідними для модуля даними є статистичні записи з усіма спробами студентів виконати завдання. Стрічки запитів також подаються в зашифрованому вигляді.

SHA-1 (Secure Hash Algorithm 1) — це криптографічний хеш-функція, яка приймає вхідні дані і генерує 160-бітний (20-байтовий) хеш-значення або дайджест повідомлення. Вона була розроблена Національним інститутом стандартів і технологій США (NIST) у співпраці з Агентством національної безпеки (NSA) і опублікована як стандарт FIPS PUB 180-1. SHA-1 широко використовується в багатьох криптографічних додатках і протоколах, таких як TLS і SSL, PGP, SSH та IPsec.

SHA-1 працює шляхом обробки вхідних даних у блоках по 512 біт. Вхідні дані спочатку доповнюються, щоб їх довжина стала кратною 512, після чого обробляються блоками, використовуючи серію логічних операцій

Систему можна розділити на бібліотеки, форми, класи і конфігураційний файл. Конфігураційний файл містить в собі стрічки підключення до баз даних. Кожна форма імплементує певне вікно користувацького інтерфейсу та функціонал який стоїть за ним.

Логіку та функціонал додатку крім форм реалізовують також бібліотеки класів. Загалом модуль містить такі бібліотеки:

1. Бібліотека DataLib — бібліотека для роботи з БД, що відповідно містить класи DB.cs, ManageDB.cs.

Клас Student.cs відповідає таблиці Students і містить поля, що відповідають полям цієї таблиці. Властивості та методи цього класу наведені в таблицях 3.1 і. 3.2.

Таблиця 3.1 – Властивості класу Student.cs

Клас	Назва властивості	Опис
Student.cs	string name	Ім’я об’єкту навчання (студента)
	string password	поле зашифрованого пароля студента
	int group_id	id групи до якої належить студент

Таблиця 3.2 – Методи класу Student.cs

Клас	Назва методу	Опис
Student.cs	bool get(int id)	Метод, що виконує зчитування даних з таблиці у відповідний клас за id
	bool set(int id)	Метод, що виконує запис даних в таблицю за поточним id
	int add()	Метод, що здійснює додавання нового запису в таблиці з даними відповідного класу
	bool destroy(id)	Метод, що відповідає за видалення таблиці за поточним id запису

Властивості класу Group.cs подано в таблиці 3.3, а методи відповідно такі ж як в попереднього.

Таблиця 3.3 – Властивості класу Group.cs

Клас	Назва властивості	Опис
Group.cs	string name	Поле, де вказано назву групи
	TaskList tasks	Перелік завдань, що відносяться до поточної групи

Клас Task.cs є відповідником таблиці Task, властивості якого наведено в таблиці 3.4.

Таблиця 3.4 – Властивості класу Task.cs

Клас	Назва властивості	Опис
Task.cs	string name	поле імені завдання
	string description	Поле з вказанням опису поточного завдання
	string erd	Шляху, що вказується до відповідного зображення erd
	string answer	поле результату правильного запиту у зашифрованому вигляді
	string time	Вказівник часу, відведений на виконання завдання
	float value	кількість балів, для оцінювання поточного завдання
	GroupList groups	Перелік груп, що відповідають поточному завданню

Клас Teacher.cs відповідає таблиці Teacher в базі даних. Основні методи цього класу представлено в таблиці 3.5, а властивості цього класу такі ж як в попереднього.

Таблиця 3.5 – Властивості класу Teacher.cs

Клас	Назва властивості	Опис
Teacher.cs	string name	Поле, де вказується ім’я викладача
	string password	Пароль викладача в зашифрованому вигляді
	int group_id	id групи до якої належить студент

За допомогою класу DB.cs здійснюється підключення до бази даних, в якій буде здійснюватись тестування.

Клас ManageDB.cs використовується для підключення до бази даних організації тестування.

2. Бібліотека Hash — в даному випадку виконує хешування даних згідно алгоритму SHA1 і містить тільки клас HashSum.cs.

Для роботи системи необхідне наступне встановлене програмне забезпечення: Microsoft .Net Framework 4, MS SQL Server 2014, Microsoft Visual C++ Redistributable x86.

3.4. Розробка та опис інтерфейсу користувача студенто-центричного рішення тестування знань

Організація взаємодії з користувачем базується на графічному інтерфейсі, представленому формами додатку та передбачає собою прямий діалог користувача з системою.

Графічний інтерфейс є одним з типів інтерфейсів, в якому взаємодія з користувачем базується на використанні графічних зображень та елементів. Діалог типу регламентованого діалогу є першим в історії способом взаємодії користувача з комп'ютером.

Загалом форм в модулі "Студент" є чотири: UserLogin для вікна авторизації, TaskMenu для вікна вибору задвань, TaskExec для виконання завдання та TaskView для перегляду умов завдання.

На рисунку 3.6 представлено вікно авторизації користувача, в даному вікні користувач може здійснити вхід у систему, обраши свою групу та ім'я в випадаючих списках "Stundent" та "Group", ввівши персональний пароль в поле "Password". Виконання входу здійснюється натисканям кнопки "Login". Також є можливість завершення роботи програми при натисканні стандартної кнопки вікна "Закрити".

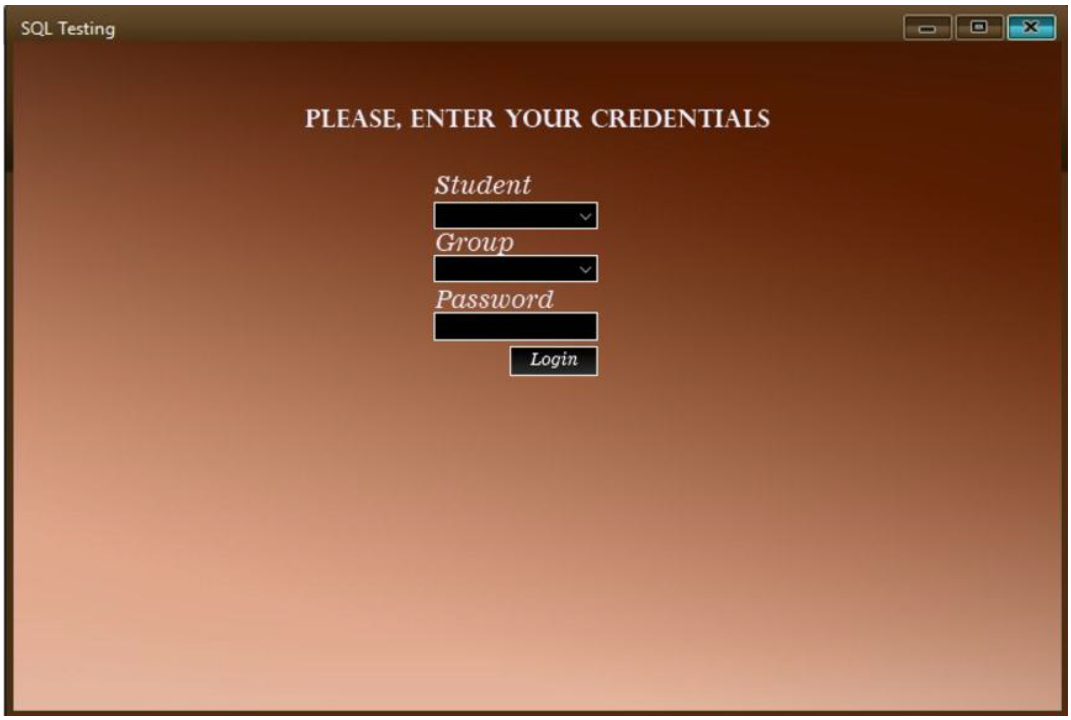


Рисунок 3.6 – Вікно входу в систему (авторизації)

На рисунку 3.7 зображено вікно вибору завдань. У цьому вікні користувач бачить список завдань, представлений у таблиці короткою інформацією про завдання. Також вікно містить інформацію про користувача та поточну оцінку.



Рисунок 3.7 – Вікно вибору завдань

На рисунку 3.8 зображено вікно виконання завдання. У цьому вікні користувач має змогу ввести запит в поле “Your SQL query” як відповідь на завдання. Підтвердження виконання запиту здійснює кнопка “Execute”.

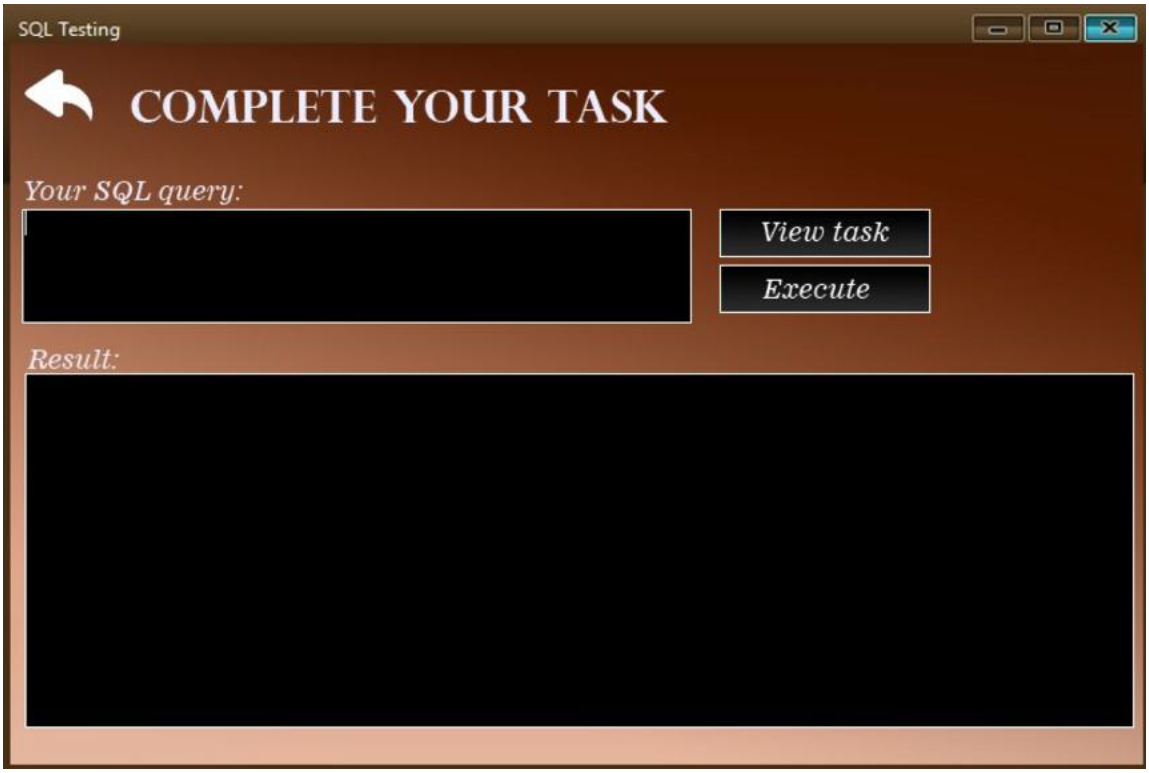


Рисунок 3.8 – Вікно виконання завдання

Результат виконання такого запиту відображається в полі “Result”. Кнопка “View task” передбачає відкриття вікна перегляду умови завдання.

Користувач може повернутися до попереднього меню натиснувши кнопку у вигляді стрілки.

На рисунку 3.9 зображено вікно перегляду умови завдання. У цьому вікні користувач має змогу побачити опис завдання в полі “Description” та зображення ERD в полі “ERD”. Є можливість повернення до попереднього меню з використанням кнопки у вигляді стрілки або завершити роботу з використанням кнопки “Закрити”.

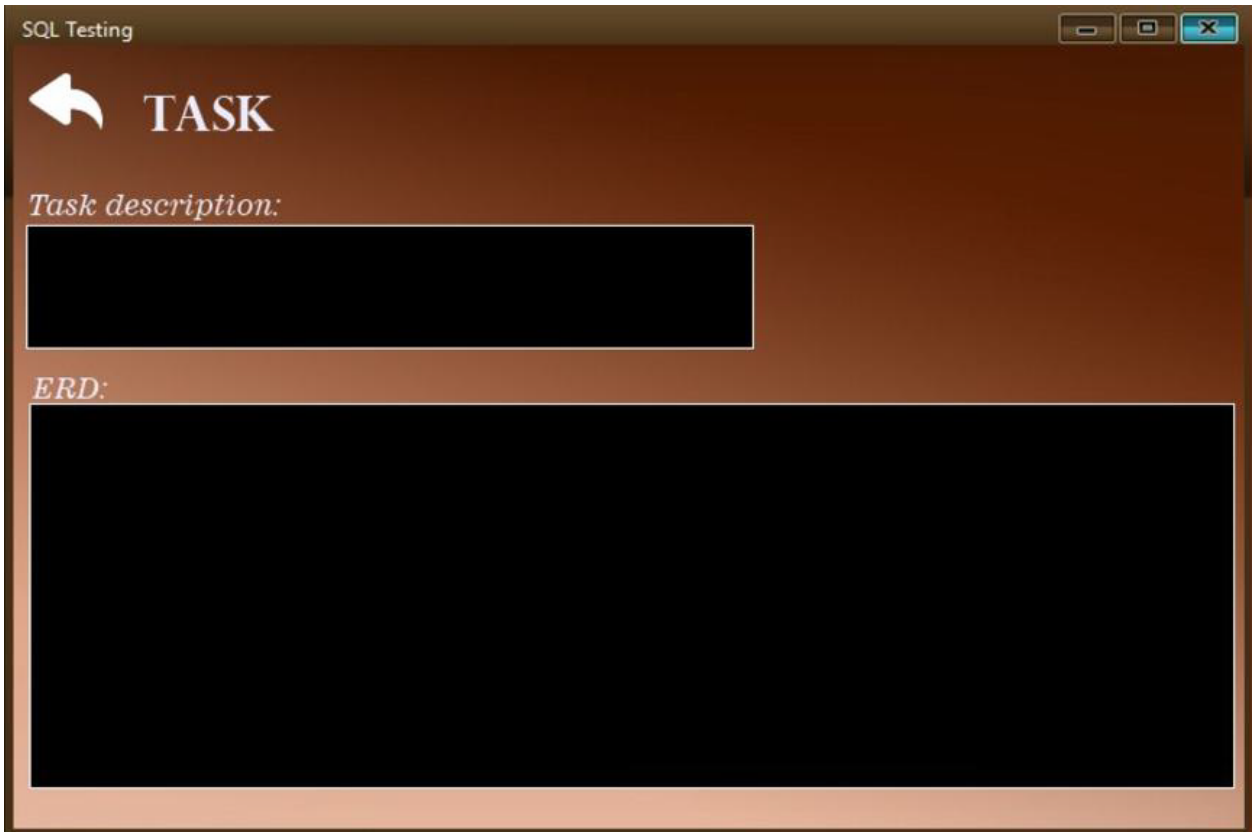


Рисунок 4.4 – Вікно перегляду умови завдання тесту

Отже, в даному розділі проведено реалізацію програмного продукту студенто-центричного рішення тестування знань, розроблено його архітектуру та модель бази даних. Також представлено і описано інтерфейс розробленого програмного рішення.

ВИСНОВКИ

В дипломній роботі виконано проектування та розробку студенто-центричного програмного рішення тестування знань, а саме програмного забезпечення та бази даних автоматизації процесу тестування для студента. Показано застосування сучасних технологій створення додатків для розробки програмної підсистеми тестування студентів по SQL запитах.

Виконано детальний аналіз вже існуючих систем для тестування, які дозволяють створювати тестові завдання та автоматично перевіряти результати.

Відповідно до мети роботи було виконано такі основні задачі, які ставились до розроблюваного програмного продукту:

1. Досліджено існуючі аналоги програмних систем для тестування з виділенням їх основних позитивних та негативних характеристик.

2. На основі виділених характеристик здійснено розробку функціональних та не функціональних вимог до розроблюваного продукту.

3. Розроблено та спроектовано модель бази даних розроблюваного програмного рішення.

4. Обрано та представлено обгрунтування необхідних технологій для реалізації обраного функціоналу.

5. Проведена розробка програмної системи для тестування знань студентів.

Розроблене студенто-центричне програмне рішення для тестування знань демонструє значний потенціал для покращення якості освіти. Запропонована система сприяє індивідуалізації навчання, забезпечує зручний і ефективний інструмент для оцінювання знань та надання зворотного зв'язку. Подальший розвиток та впровадження цього рішення в освітній процес може суттєво підвищити ефективність навчання та успішність студентів.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Слива М.В. Прототипування графічного інтерфейсу користувача як невід'ємна частина процесу розробки програмного забезпечення. Вісник Нижневартовского державного університету. 2013. No 1. С. 74–76.
2. Прототипування програми: від ідеї до робочого екрану. URL: https://habr.com/company/mobile_dimension/blog/327452/.
3. Мардан А. Швидке Прототипування з JS. Гнучка Розробка на JavaScript. М. : ІТУС, 2014. 236 с.
4. Немтінов В.А. Віртуальне моделювання, прототипування і промисловий дизайн. Тамбов : ТДТУ, 2019. 339 с.
5. Поляков О.М. Основи швидкого прототипування. Оренбург : ОДУ, 2014. 128 с.
6. Глухова Л.А. Технології розробки програмного забезпечення. Мінськ : БДУІР, 2014. 97 с.
7. Kolud R. Knowledge testing strategies based on internet systems / Proc. of the Intern. Conf. on Inductive Modeling "ICIM-2002".- V. 3.- Lviv, 2002.- P. 233-239.
8. 20 інструментів для прототипування: від швидкого і брудного wireframe до функціонального прототипу. URL: https://medium.com/ @denysergushki/.
9. Rudnichenko N.D., Shibaeva N.O., Boyko V.D. Model prototype development of interactive interface of information system for monitoring ship technical system. Інформаційні технології та комп'ютерна інженерія: матеріали статей конференції, м. Івано-Франківськ, 27-29 травня 2015 року. Івано-Франківськ, 2015. С. 217–218.
10. Vychuzhanin V.V., Rudnichenko N.D., Shibaeva N.O., Boyko V.D. The development of user interface prototype of decision support system for risk

management of complex technical systems. Sustainability and Competitiveness in Business. 2016. PP. 162–172.

11. Kolud R. Internetowa witryna edukacyjna Wyższej Szkoły Informatyki i Zarządzania.- Bielsko-Biała: Wyższa Szkoła Informatyki i Zarządzania, 2003.- 42 s.
12. Kolud R. Internetowy System Testowania - Egzamin Dyplomowy.- Bielsko-Biała: Wyższa Szkoła Informatyki i Zarządzania, 2001.- 22 s.
13. Ільїна О. І. Використання електронного тестування для контролю якості знань студентів : https://drive.google.com/file/d/0BzvVkHv_vLlSZlMxUkY3MHp3bjRoZjI5VE83MHZ2MUVRWDBJ/ view?usp=sharing
14. Таран І. Б. Застосування LEARNINGAPPS.ORG майбутніми вихователями для створення інтерактивних вправ ФІЗИКО-МАТЕМАТИЧНА ОСВІТА (ФМО) випуск 4(10), 2016: http://conf.iitlt.gov.ua/Images/Files/TARAN_88_1417785843_file.docx
15. Биков В.Ю. Інформаційне забезпечення навчально-виховного процесу: інноваційні засоби і технології: Колективна монографія / В.Ю. Биков, Гриценчук О.О., Жук Ю.І. та ін.// - К.: Атака, 2005. - 252 с.
16. Cohn, D. Learning for probabilistically relative identify authoritative documents. In Proc. 17th International Conf. on Computer Learning, pages 167174, 2000.
17. Maganti, A. An negotiation of linguistic actions and clustering algorithms for typical document proceeding. In Proc. of the SIGIR'2000, 2000.
18. Moran, S. The stochastic approach for link-structure analysis (salsa) and the tkc-effect. In Proc. WWW9, 2000.
19. Peter Brusilovsky. Adaptive Systems // User Modeling and UserAdapted Functions 11: 87 - 110, 2001.
20. Генсьорська М. М. Адаптивне тестування в освіті // Інформаційнокомунікаційні технології в освіті. – 2014. - №1 [Електронний ресурс] – Режим доступу: http://www.e-

journals.npu.edu.ua/ index.php/ikt/article/ viewFile/36/pdf_20 (дата звернення: 22.11.2019)

21. Weiss D. J.(Ed.) New Horizons in Testing: Latent Trait. Test Theory and Computerised Adaptive Testing. N.Y., Academic Press, 1983. – 345 pp.
22. Lord P. M. Application of Item Response Theory to Practical Testing Problems. Hillsdale N – J. Lawrence Erlbaum Ass., Publ. 1980, – 266 pp.
23. Гайтан О. М. Елементи технології реалізації автоматизованого адаптивного контролю знань студентів в комп'ютерних системах навчання // ISSN 18144225. Радіоелектронні і комп'ютерні системи. – 2014. – № 4 (68)

БІБЛІОГРАФІЧНА ДОВІДКА

Тема дипломної роботи: “Побудова студенто-центричного програмного рішення тестування знань ”

Обсяг пояснювальної записки: 79 аркушів.

Дата закінчення роботи: 21 червня 2024 р.

Підпис студента _____