МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерних наук

ТВОРКО Михайло Вадимович

Підсистема "Викладач" програмної системи тестування студентів/ "Lecturer" subsystem of software system for students testing

напрям підготовки: 6.050103 - Програмна інженерія
фахове спрямування - Програмне забезпечення систем

Бакалаврська дипломна робота

Виконав студент групи ПЗС-42
М. В. Творко

_____

Науковий керівник:
викладач КРЕПИЧ С.Я.

_____

Бакалаврську дипломну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **А. В. Пукас**

ТЕРНОПІЛЬ - 2016

РЕЗЮМЕ

Дипломна робота містить 84 сторінки, 20 таблиць, 34 рисунки, список використаних джерел із 23 найменувань та 2 додатки.

Метою дипломної роботи є розробка програмного забезпечення та бази даних для автоматизації процесу тестування для викладача.

Об’єктом дослідження є процес підготовки тестів викладачем для проведення тестування студентів.

Предметом дослідження є застосування сучасних інформаційних технологій для створення підсистеми “Викладач” програмного продукту для тестування студентів по SQL запитах.

Методи розробки базуються на технології .NET, бази даних MS SQL Server.

Одержані результати полягають в розробці підсистеми “Викладач” програмної системи для тестування студентів по SQL запитах.

Ключові слова: викладач, студент, запит, програмний комплекс, база даних, діаграма класів, реляційна модель бази даних, тестування.

SUMMARY

Thesis contains 84 pages, 20 tables, 34 figures, list of sources with 23 titles and 2 applications.

The aim of the thesis is the development of software and databases to automate the testing process for the teacher.

The object of the research is a process of creating tests by teacher for testing students.

The subject of research is the use of modern information technology to create a subsystem "Teacher" of the system for testing students on SQL queries.

Methods of development based on technology .NET, database MS SQL Server.

The results are in development of subsystem "Teacher" of the system for testing students on SQL queries.

Keywords: teacher, student, query, software system, database, diagram classes relational model database testing.

ЗМІСТ

ВСТУП

Актуальність дипломної роботи полягає в тому, що на сьогоднішній день не існує такої програмної системи, яка б допомогла викладачам підготувати хороших спеціалістів по базам даних. На даний момент, таких спеціалістів готують власними силами, вручну перевіряючи знання студентів. Дуже важливим є заощадження часу при проведенні тестування, адже часто нехватка часу може бути критичною. Зберегти час можна різними способами, і одним з таких способів є автоматизація.

Автоматизація навчального процесу, в даному випадку процесу підготовки до тестування викладачеві, а саме створення тестових завдань по написанню SQL запитів, є дуже ефективною для викладача, адже це дозволяє швидко та легко створити завдання, враховуючи усю інформацію та статистику про виконані студентами завдання, та внести його в систему.

Мета роботи полягає в дослідженні та детальному аналізі уже існуючих систем для тестування, які дозволяють створювати тестові завдання та перевіряти результати автоматично, і розробка, на основі проведеного аналізу, підсистеми “Викладач” системи тестування студентів по SQL запитах.

Отож, спираючись на мету, можна виділи такі основні задачі, які ставляться до розроблюваного продукту:

1. Дослідити існуючі аналоги програмних систем для тестування. Виділити їх основні позитивні та негативні риси.
2. На основі виділених рис розробити функціональні та не функціональні вимоги до розроблюваного продукту.
3. Розробити модель бази даних продукту.
4. Обрати необхідні технології для реалізації обраного функціоналу.
5. Розробка програмної системи “Викладач” для тестування студентів по SQL запитах, описаних в пунктах 1-4.

Об’єкт дослідження – процес автоматизованого тестування студентів по SQL запитах зі сторони викладача.

Предмет досліджень – застосування сучасних технологій створення додатків для розробки підсистеми “Викладач ”програмної системи тестування студентів по SQL запитах.

Під час створення системи використовуватимемо наступні технології необхідні для розробки програмної системи:

- MS SQL Server – реляційна система управління базами даних, створена компанією Microsoft.
- EntityFramework – спеціальна об’єктна-орієнтована технологія на базі фреймворка .NET для роботи з даними.
- C# – об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET.
- SQL – стандартна мова програмування, яка використовується для отримання доступу до бази даних, а також для маніпуляції з нею.

Практична цінність розроблюваної програмної системи полягає в автоматизації тестування студентів по SQL запитах, швидкого створення тестових завдань, перегляду статистики результатів.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Коротка характеристика об'єкту управління

Дуже важливим є заощадження часу при проведенні тестування, адже часто нехватка часу може бути критичною для студентів. Зберегти час можна різними способами, і одним з таких способів є автоматизація.

Автоматизація навчального процесу, в даному випадку процесу підготовки до тестування викладачеві, а саме створення тестових завдань, є дуже ефективною для викладача, адже це дозволяє швидко та легко створити завдання, враховуючи усю інформацію та статистику про виконані студентами завдання, та внести його в систему.

Створення завдання викладачем проходить у декілька кроків. Спершу, викладач повинен авторизуватись у систему, увівши свій логін та пароль. Потім є можливість приступити до створення завдання одразу або після перегляду статистики. Статистика показує, які завдання були виконані та не виконані студентами, та хто саме із студентів виконав або не виконав те чи інше завдання.

Переглянувши статистику, викладач складає завдання відповідної складності, прикріпляє діаграму ERD у вигляді зображення (ERD – діаграма “сутність-зв’язок”) та зберігає завдання у БД.

Під час проведення тестування, студенти, які вже завершили виконання завдання, відразу отримують повідомлення з оцінкою їх роботи. Якщо студент виконав завдання не правильно, то програма повідомить йому про це, та не дозволить завершити завдання. Також, студенту повинна відводитись певна кількість часу, тому система веде лік часу у вигляді таймера. У випадку завершення часу у таймері, програма автоматично ставить студенту незадовільну оцінку.

Після того як усі студенти завершили виконання завдань і викладач отримав усі оцінки, йому залишається лише переписати їх у свій журнал. Тому, використання комп’ютерних технологій в навчальному процесі є дуже ефективним.

1.2. Опис предметної області модуля «Викладач» програмної системи для тестування студентів

Перед нами постає задача створення модуля викладача для системи тестування студентів. Метою розробки є можливість для викладача створювати завдання та слідкувати за їх виконанням. Необхідно визначити основні задачі системи. Для коректного проектування програмної системи потрібно визначити функції системи.

Для того, щоб повністю представити функціонал програмної системи, виділено такі основні бізнес-процеси:

- процес формування завдання викладачем;
- процес перегляду статистики;
- процес внесення змін;

На рисунку 1.1 зображено діаграму бізнес-процесів розроблюваного програмного продукту.

Рисунок 1.1 – Діаграма бізнес-процесів розроблюваного програмного продукту

Тепер детальніше розглянемо кожен з вище представлених бізнес-процесів. На рисунку 3.2 зображено процес створення завдання викладачем.

Рисунок 1.2 – Процес створення завдання викладачем

Першим, що потрібно зробити викладачеві, це створити завдання для подальшого додавання його в БД. Завдання поділяється на такі особливості:

- умова – містить інформацію про виконання тестового завдання;
- додатки – в основному це ERD-діаграма, яка містить опис сутностей БД для виконання завдання;
- правильна відповідь – це відповідь, яка буде означати правильне виконання завдання.

Характеристику бізнес-процесу створення завдання викладачем наведено в таблиці 1.1.

Таблиця 1.1

Характеристика бізнес-процесу створення завдання викладачем

Назва характеристики	Значення характеристики
Ім'я бізнес-процесу	Створення завдання викладачем
Основні учасники	Викладач
Вхідна подія	Запит на створення завдання
Вхідні документи	ERD-діаграма
Вихідна подія	Перегляд створеного завдання
Вихідні документи	-
Клієнт бізнес-процесу	-

На рисунку 3.3 зображено діаграму функцій процесу перегляду статистики

Рисунок 1.3 – Діаграма функцій перегляду статистики

Процес перегляду статистики є допоміжним процесом, адже він дозволяє переглянути статистику виконаних завдань та успішності студентів для подальшого створення нових завдань з більшою або меншою складністю, відповідно до успішності. Він поділяється на такі частини:

- пошук за студентом;
- пошук за завданням.

Характеристику бізнес-процесу перегляду статистики наведено в таблиці 1.2.

Таблиця 1.2

Характеристика бізнес-процесу здійснення пошуку

Назва характеристики	Значення характеристики
Ім'я бізнес-процесу	Перегляд статистики
Основні учасники	Викладач
Вхідна подія	Запит на перегляд статистики
Вхідні документи	-
Вихідна подія	Відображена статистика у текстовому варіанті
Вихідні документи	-
Клієнт бізнес-процесу	-

1.3. Огляд і аналіз існуючих аналогів

На сьогодні є достатньо програмних систем, за допомогою яких можна проводити тестування студентів. В даному пункті будуть розглядатися кілька з таких.

Розглянемо для початку програму тестування знань “Айрен”

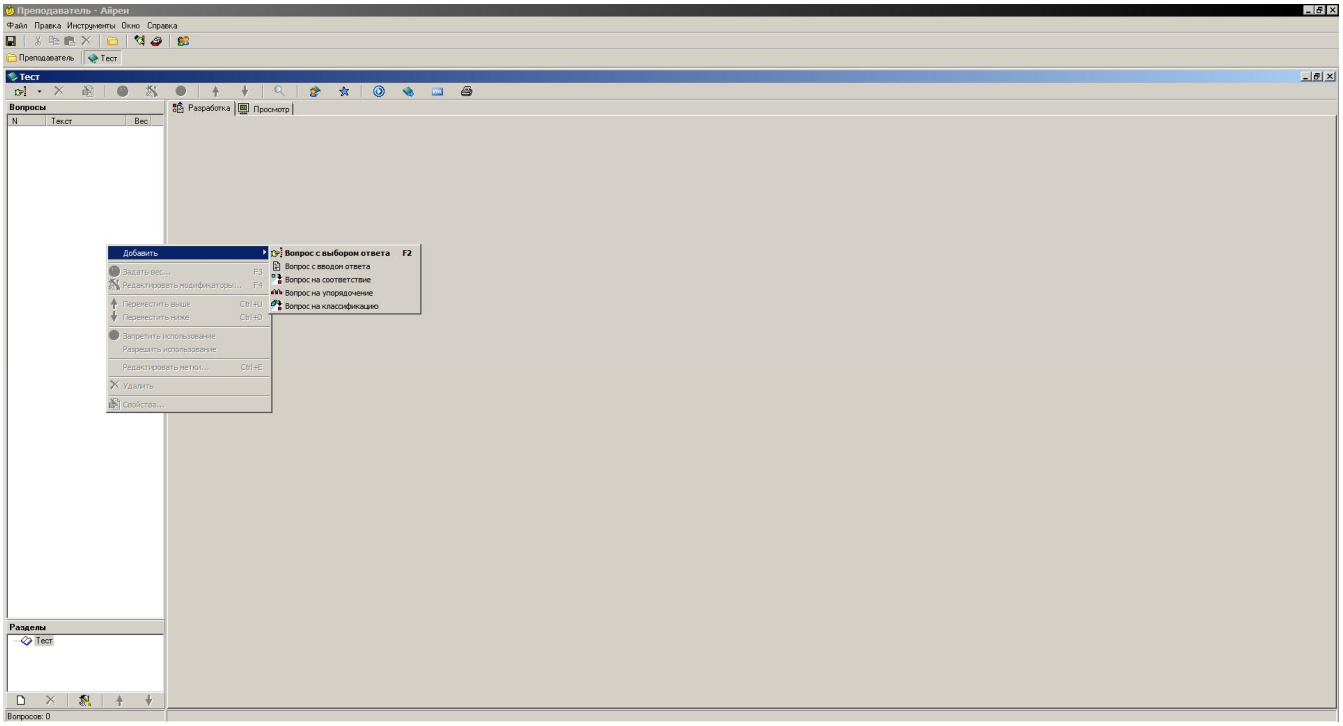

Рисунок 1.4 – Вікно створення тесту у програмі «Айрен»

Дана програма має дуже простий та доволі зручний інтерфейс, за допомогою якого користувач може досить легко зорієнтуватися в подальших діях. Недоліком є те, що у такій програмі немає можливості створити завдання, яке не буде являтись тестовим. Програма не виконує SQL запити та не вказує помилки, тому вона не підходить для виконання таких завдань, як перевірки знань студентів з SQL запитів.

Дана програмна система має наступні основні функції:

- створення завдання
- перегляд завдання
- збереження тестів у файл.

Для порівняння розглянемо програмний продукт MyTestStudentPRO

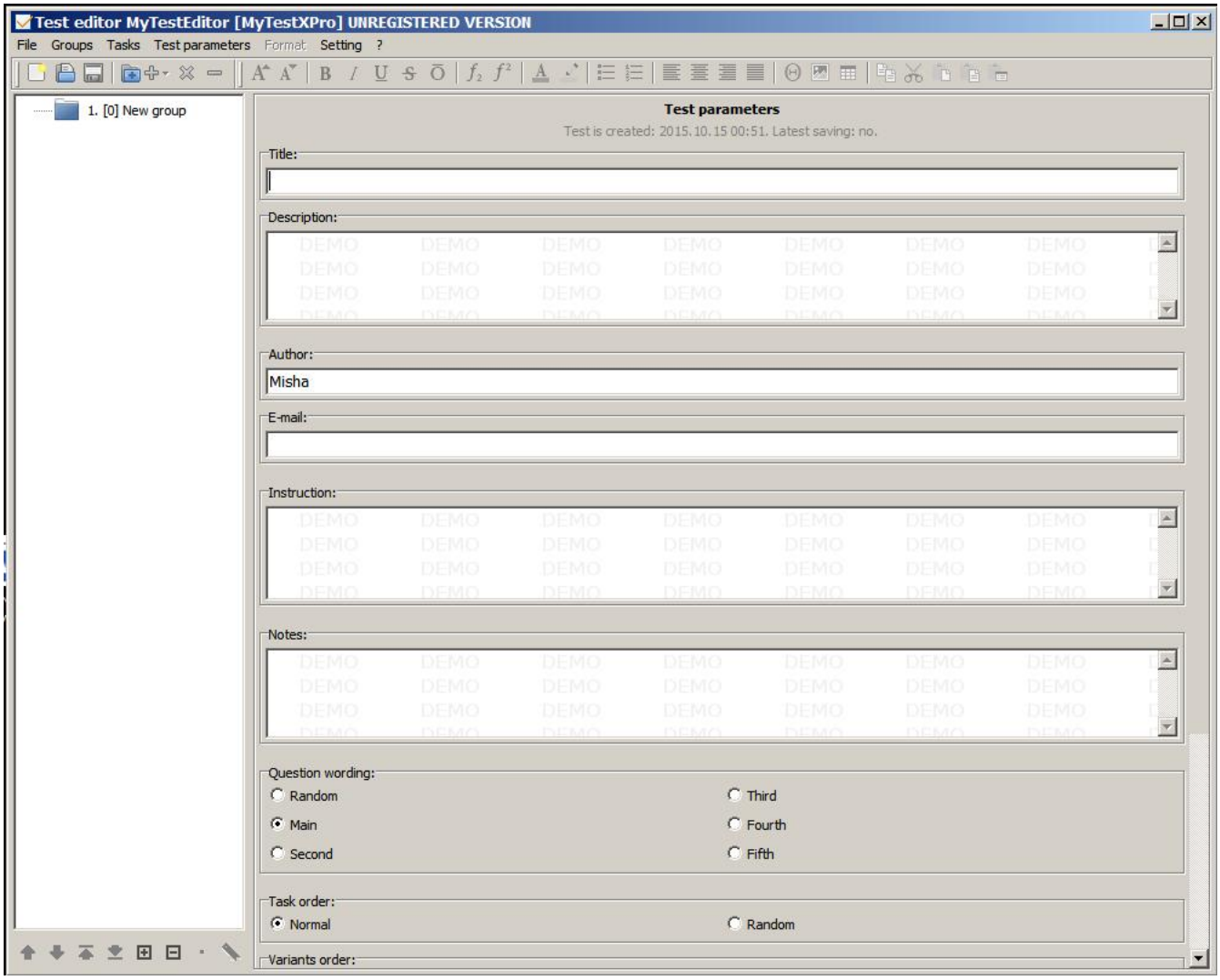

Рисунок 1.5 – Сторінка створення завдання програми MyTestStudentPRO

Даний продукт дозволяє створювати та редагувати тести. Також цей продукт має два модуля: модуль для викладача, де можна створювати завдання, та модуль для студента, де останній ці тести проходить. Даний продукт має багато різноманітних функцій, таких як:

- створення та редагування тестів.
- групування тестів.
- пошук завдань.

Наступним аналогом є веб-система “ExamBuilder”

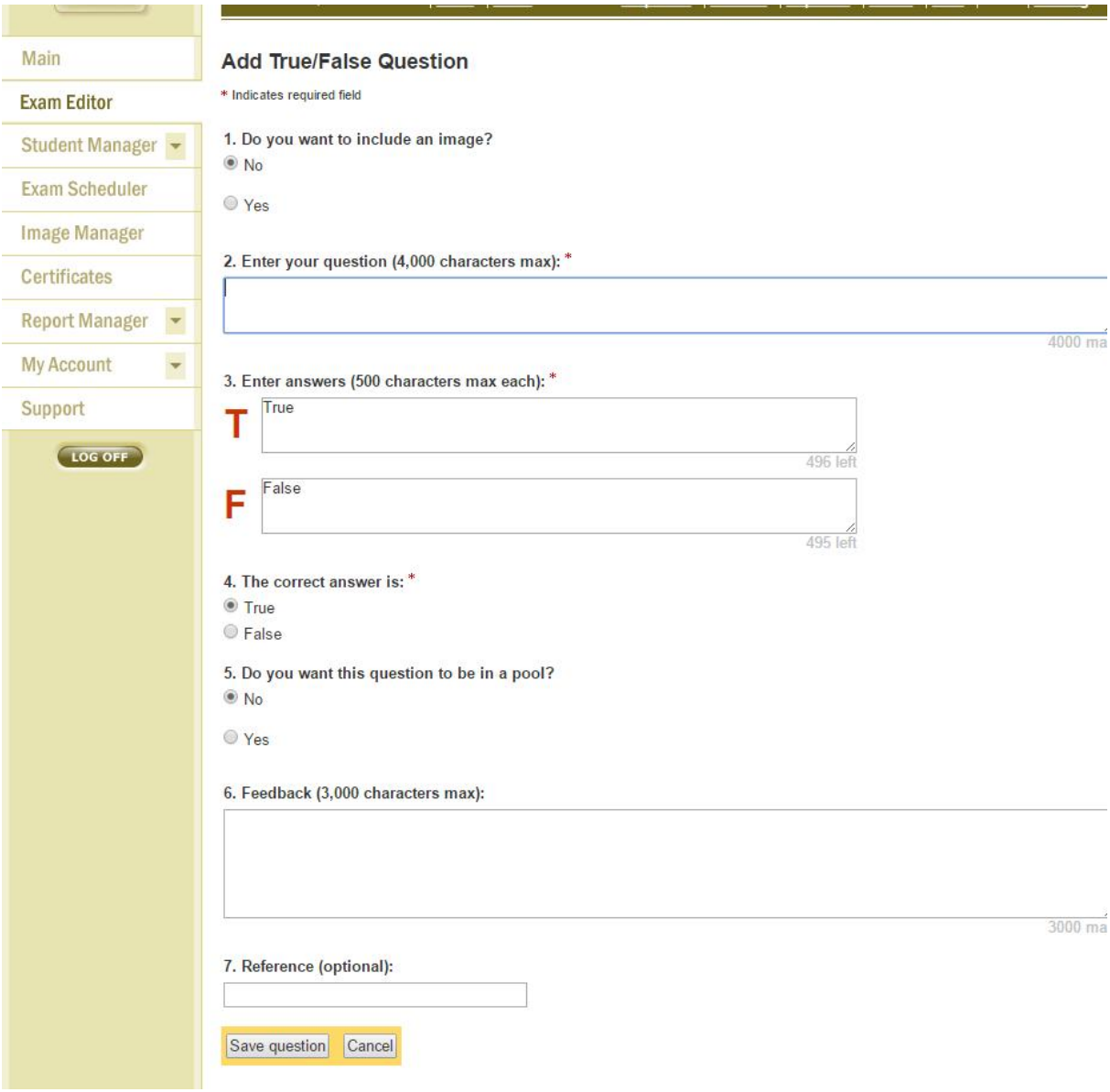

Рисунок 1.6 – Сторінка створення завдання системи Exambuilder

Дана система є доволі потужною, адже вона дозволяє створювати тести за багатьма категоріями та рівнями складності. Присутній перегляд студентського маніфесту та результатів. Викладач має можливість створювати графіки проведення тестування, сертифікати, завантажувати зображення. Перевагою є те, що дана система є веб-системою, що дозволяє отримати доступ до неї з будь-якого місця.

Оглянувши та зробивши аналіз існуючих аналогів, можна зробити висновок, що вище перелічені програмні системи є хорошими для проведення різноманітних тестів. Кожна програма функціонує досить добре. Але,

переглянувши функції кожного програмного продукту очевидним є те, що з їх використанням не можливо створити тести для перевірки студентів на знання мови SQL. Дані програмні продукти орієнтовані лише на завдання у формі тестів, без використання SQL запитів та розширених відповідей. Тому оптимальним рішенням даної проблеми є створення продукту для тестування студентів з можливістю дати розширену відповідь на завдання, використовуючи SQL запит та з можливістю викладачеві контролювати процес виконання завдань.

Здійснивши аналіз альтернатив, створено таблицю, де відображається порівняльна характеристика аналогів (див. табл. 1.4).

Таблиця 1.4

Порівняльна характеристика аналогів

Фірма-розробник	Сергій Останін	Олександр Башлаков	ExamBuilder
Назва програмного продукту	Програма тестування знань “Айрен”	MyTestEditorPRO	ExamBuilder
Версії продукту	2.3	11.0.0.37	2
Функціональність	• Можливість створення набору тестів • Можливість збереження тестів у різних форматах • Можливість створення тестів з різними способами вводу	• Можливість створення набору тестів • Можливість збереження тестів у різних форматах • Можливість створення тестів з різними способами вводу відповідей • Можливість назначення тестів	• Можливість створення набору тестів • Можливість задання різних категорій тестування та рівня складності • Можливість створення графіку тестування

	відповідей	для певних користувачів	

Продовження таблиці 1.4

		• Можливість задання системи оцінювання	• Можливість перегляду студентського маніфесту
Допомога користувачу	Присутня система допомоги користувачу та вікі проекту	Присутня система допомоги користувачу та вікі проекту	Присутня система допомоги користувачу та вікі проекту

1.4. Специфікація вимог до модуля (системи)

Специфікація вимог до програмного забезпечення являє собою повний опис передбачуваної мети і середовища для програмного забезпечення в стадії розробки. SRS повністю описує те, що програмне забезпечення буде робити і як воно буде це робити. Специфікація вимог зводить до мінімуму час і зусилля, необхідні розробниками для досягнення бажаних цілей, а також зводить до мінімуму витрати на розробку. Хороший SRS визначає, як додаток буде взаємодіяти з системою апаратних засобів, іншими програмами та користувачами різноманітних ситуаціях. Оцінюються такі параметри, як швидкість роботи, час відгуку, доступність, переносимість, ремонтопридатність, компактність, безпека і швидкість відновлення від несприятливих подій оцінюються.

У таблиці 1.5 подано глосарій основних використовуваних термінів при створенні модуля «Викладач» системи тестування знань студента.

Таблиця 1.5

Глосарій основних термінів

Термін	Опис терміну
1. Основні поняття та категорії предметної області та проекту	
База даних	Це сукупність даних, організованих як набір формально описаних

	таблиць, до яких можна доступитись без потреби реорганізовувати самі таблиці.

Продовження таблиці 1.5

Таблиця бази даних	Організовує інформацію в рядки та стовпці. Кожна частинка даних є полем у таблиці. Стовпець складається з усіх записів у одному полі.
SQL (*Structured query language)*	Стандартна мова програмування, яка використовується для отримання доступу до бази даних, а також для маніпуляції з нею.
Модель «сутність-зв'язок» (ER-модель)	Модель даних, яка показує відношення сутностей в базі даних. Ілюструє логічну структуру бази даних.
Система керування базами даних	Системне програмне забезпечення для створення та управління базами даних. СУБД надає користувачам і програмістам можливість створювати, видаляти, оновлювати та управляти даними.
Тестування	Це метод перевірки знань та оцінювання, який найчастіше проявляється у виконанні завдань у вигляді тестів.
Оцінка	Зазвичай визначається у вигляді числа, визначає рівень знань учня чи студента або рівень виконання певної роботи.
2. Користувачі системи	
Викладач	Особа, що використовує модуль «Викладач» системи тестування знань студента для того, щоб створювати завдання для студентів, перегляд та редагування студентського маніфесту, перегляд статистики результатів.
3. Вхідні та вихідні документи	
Запит	Запит інформації з бази даних.

Тестове завдання	Складова частина тесту, що відповідає вимогам до завдань у тестовій формі та пройшла обов’язкову перевірку статистичних властивостей.

Продовження таблиці 1.5

Дані	Різні фрагменти інформації, як правило, відформатовані особливим чином. Дані можуть існувати в різних формах - у вигляді чисел або тексту на аркушах паперу, в якості бітів і байтів, що зберігаються в електронній пам'яті, або у вигляді фактів, що зберігаються в свідомості людини.

На рисунку 1.7 наведено повну діаграму варіантів використання системи тестування студентів по SQL запитах.

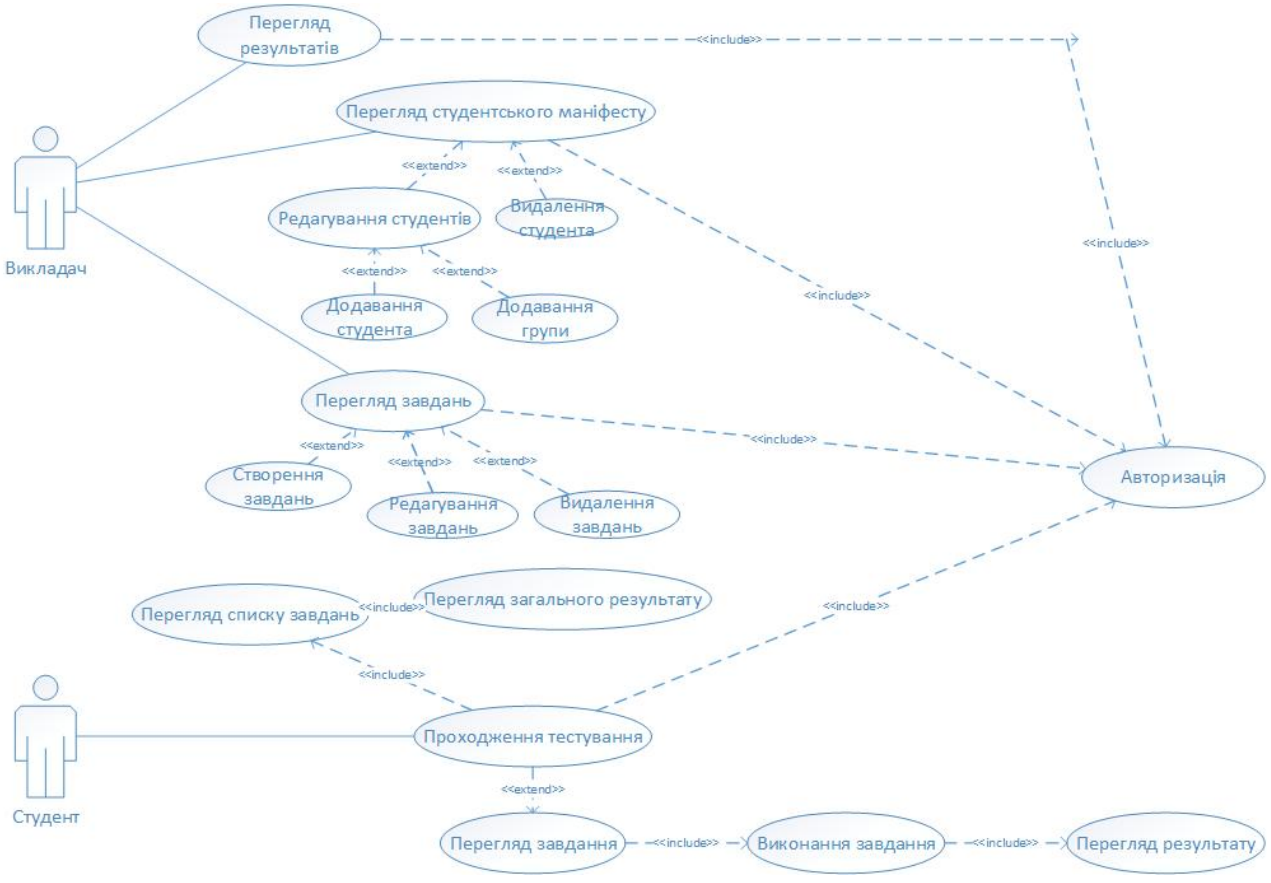

Рисунок 1.7 – Діаграма варіантів використання системи

Далі, наведемо діаграму варіантів використання для акторів підсистеми “Викладач” на рисунку 1.8.

Рисунок 1.8 – Діаграма варіантів використання підсистеми “Викладач”

Наступним кроком у визначенні специфікацій вимог до системи буде опис варіантів використання.

Варіанти використання представленні у таблицях 1.6-1.12.

Варіант використання «Перегляд результатів» подано у таблиці 1.6. «Перегляд результатів» передбачає перегляд статистики результатів виконання тестів студентами, огляд оцінок та успішності виконання завдань. Є можливість сортування завдань за кількістю зданих або не зданих тестів.

Таблиця 1.6

Варіант використання «Перегляд результатів»

Контекст використання	Перегляд результатів з метою оцінки якості виконаних завдань
Дійові особи	Викладач
Передумова	Користувач авторизувався в системі
Триггер	Користувач авторизувався в системі
Сценарій	Користувач переглядає результати виконаних завдань
Післяумова	-

Варіант використання «Перегляд студентського маніфесту» подано у таблиці 1.7. «Перегляд студентського маніфесту» дає можливість переглянути

списки студентів та інформацію про них. Є можливість сортування за групами та за іменем.

Таблиця 1.7

Варіант використання «Перегляд студентського маніфесту»

Контекст використання	Перегляд студентського маніфесту з метою перевірки даних
Дійові особи	Викладач
Передумова	Користувач авторизувався в системі
Триггер	Користувач авторизувався в системі
Сценарій	Користувач переглядає студентський маніфест
Післяумова	-

Варіант використання «Авторизація» подано у таблиці 1.8. «Авторизація» - вхід користувача у систему з метою подальшого користування.

Таблиця 1.8

Варіант використання «Авторизація»

Контекст використання	Авторизація в систему з метою ідентифікації
Дійові особи	Викладач
Передумова	Запуск додатку
Триггер	Запуск додатку
Сценарій	Користувач вводить авторизаційні дані
Післяумова	Користування додатком

Варіант використання «Редагування студентів» подано у таблиці 1.9. «Редагування студентів» надає змогу редагування чи видалення інформації про студента у меню перегляду студентського маніфесту.

Таблиця 1.9

Варіант використання «Редагування студентів»

Контекст використання	Редагування студентських даних з метою їх оновлення
Дійові особи	Викладач

Продовження таблиці 1.9

Передумова	Перегляд студентського маніфесту
Триггер	Перегляд студентського маніфесту
Сценарій	Користувач редагує студентські дані
Післяумова	Студентські дані оновлені

Варіант використання «Перегляд завдань» подано у таблиці 1.10. «Перегляд завдань» надає змогу викладачеві переглядати список завдань, створювати, редагувати, та видаляти завдання.

Таблиця 1.10

Варіант використання «Перегляд завдань»

Контекст використання	Перегляд завдань
Дійові особи	Викладач
Передумова	Користувач авторизувався в системі
Триггер	Користувач авторизувався в системі
Сценарій	Користувач переглядає дані
Післяумова	-

Варіант використання «Створення завдань» подано у таблиці 1.11. «Створення завдань» дає можливість викладачеві створити нове тестове завдання для студентів.

Таблиця 1.11

Варіант використання «Створення завдань»

Контекст використання	Створення завдань
Дійові особи	Викладач
Передумова	Перегляд завдань
Триггер	Вибрана опція створення завдань
Сценарій	Користувач створює завдання
Післяумова	Створене завдання

Варіант використання «Редагування завдань» подано у таблиці 1.12. «Редагування завдань» надає змогу викладачеві редагувати або видаляти вже існуюче завдання.

Таблиця 1.12

Варіант використання «Редагування завдань»

Контекст використання	Редагування завдань
Дійові особи	Викладач
Передумова	Перегляд завдань
Триггер	Вибрана опція редагування завдань
Сценарій	Користувач редагує завдання
Післяумова	Оновлене завдання

Розкадровка варіантів використання

Прототип для функції «Авторизація» зображено на рисунку 1.9

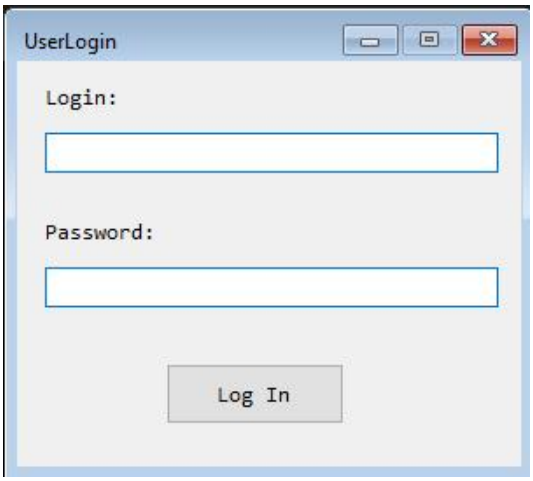

Рисунок 1.9 – Прототип для функції «Авторизація»

Прототип для функції «Створення завдання» на рисунку 1.10.

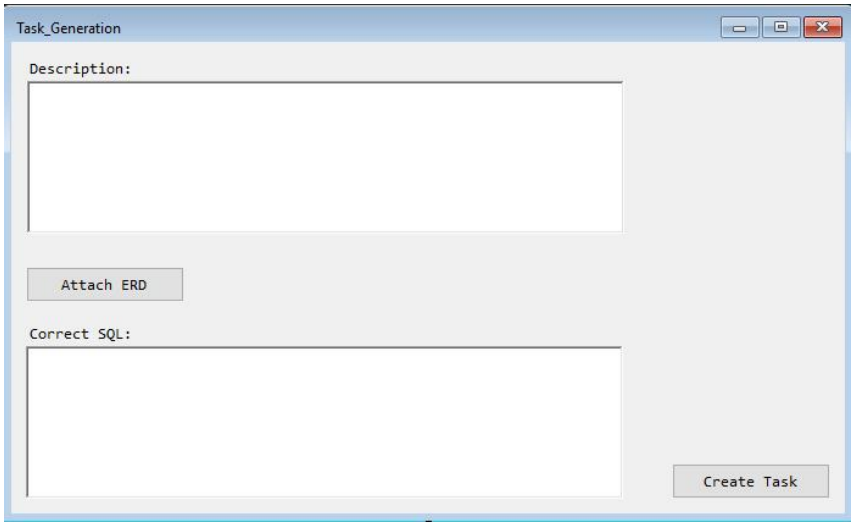

Рисунок 1.10 – Прототип функції «Створення завдання»

Прототип для функції «Перегляд завдань» на рисунку 1.11.

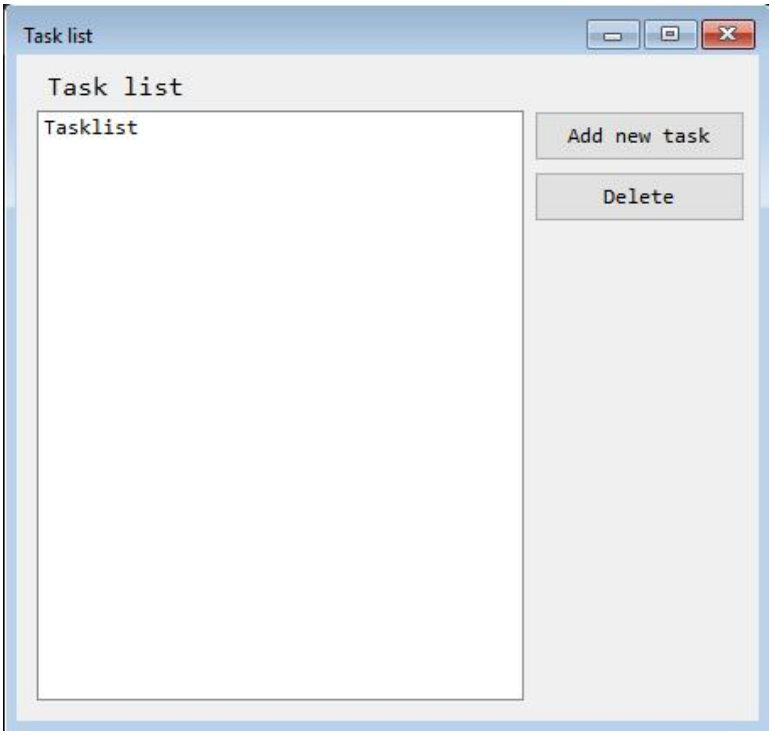

Рисунок 1.11 - Прототип для функції «Перегляд завдань»

Прототип для функції «Перегляд студентського маніфесту» зображено на рисунку 1.12.

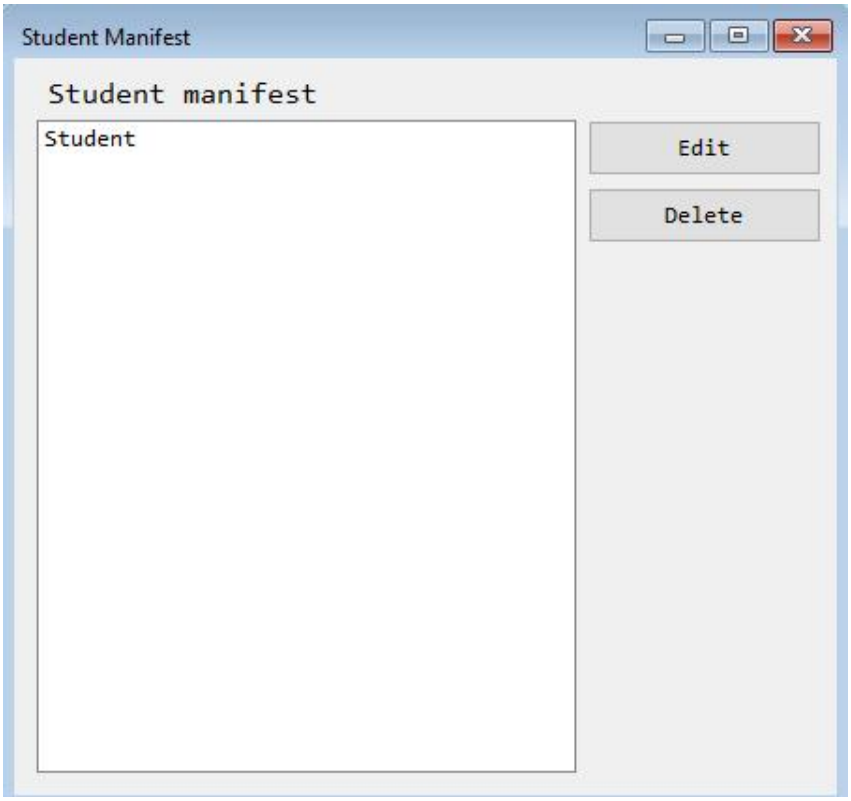

Рисунок 1.12 - Прототип для функції «Перегляд студентського маніфесту»

Прототип для функції «Перегляд результатів» на рисунку 1.13.

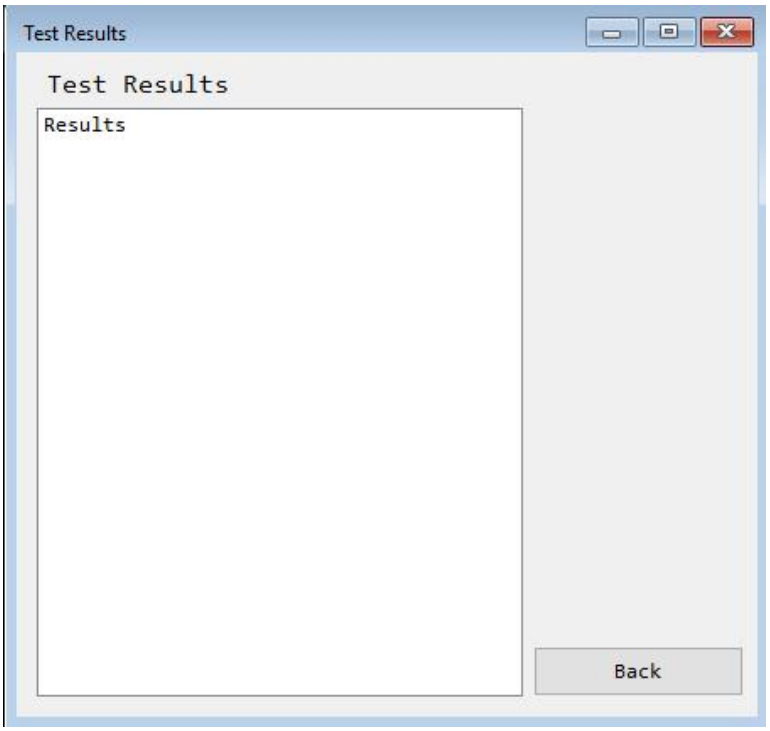

Рисунок 1.13 - Прототип для функції «Перегляд результатів»

У таблиці 1.13 наведено специфікацію функціональних вимог до програмної системи.

Таблиця 1.13

Специфікація функціональних вимог

Ідентифікатори вимоги	Назва вимоги	Атрибут вимог		
		Пріоритет	Складність	Контакт
1	Авторизація	Обов’язкове	Низька	Викладач
2	Перегляд завдань	Середній	Середня	Викладач
3	Перегляд результатів	Високий	Середня	Викладач
4	Створення завдань	Обов’язкове	Висока	Викладач
5	Редагування завдань	Високий	Висока	Викладач

Продовження таблиці 1.13

6	Перегляд студентського маніфесту	Середній	Середня	Викладач
7	Редагування студентів	Середній	Висока	Викладач
8	Виконання запиту в СУБД	Обов’язкове	Висока	Викладач
9	Запис даних у базу даних	Обов’язкове	Висока	Викладач

У таблиці 1.14 наведено специфікацію нефункціональних вимог до програмної системи.

Таблиця 1.14

Специфікація нефункціональних вимог

Ідентифікатор вимоги	Назва вимоги	Атрибути вимог		
		Пріоритет	Складність	Контакт
1. Застосовність				
1.1	Основні вимоги застосовності нової системи відносно інших систем, які знають користувачі	Рекомендована	Низька	Викладач
1.2	Вимоги по відповідальності стандартам графічного інтерфейсу користувача	Обов’язкова	Низька	Викладач

Продовження таблиці 1.14

1.3	Час, необхідний для навчання звичайних і досвідчених користувачів	Рекомендована	Середня	Викладач
2. Надійність				
2.1	Доступність	Обов’язкова	Середня	Викладач
2.2	Середній час безвідмовної роботи	Обов’язкова	Середня	Викладач
2.3	Точність	Обов’язкова	Середня	Викладач
3. Робочі характеристики				
3.1	Використання ресурсів	Рекомендована	Середня	Викладач
4. Проектні обмеження				
4.1	Вимоги до технології програмування	Рекомендована	Середня	Викладач

Значення нефункціональних вимог:

1. Застосовність:

1.1 Вимоги по відповідності загальним стандартам застосовності та стандартам; всі функції системи є легкими у виконанні, а структура програми не відрізняється від існуючих аналогів;

1.2 Вимоги до графічного інтерфейсу користувача – програма повинна працювати на операційній системі Windows 7 і вище;

1.3 Час, необхідний для навчання звичайних користувачів – 15 хвилин; для навчання досвідчених користувачів – 5 хвилин;

2. Надійність:

2.1 Доступність – час, що витрачається на обслуговування системи не повинен перевищувати 2% від загального часу роботи;

2.2 Середній час безвідмовної роботи – 3години;

3. Робочі характеристики:

3.1 Використання ресурсів – мінімальні системні вимоги:

- Об’єм HDD: > 20 Мб
- Об’єм RAM: > 512 Мб
- Відео пам'ять: > 128 Мб
- Процесор: > 1 ГГц
- Клавіатура, маніпулятор миша

Висновок до першого розділу

Проведено аналіз проблеми, досліджено шляхи до її вирішення, оглянуто аналоги програмної системи, розроблено специфікацію вимог до програмного продукту.

РОЗДІЛ 2. РОЗРОБКА ПРОЕКТУ ПРОГРАМНОЇ СИСТЕМИ

2.1. Розроблення архітектури програмної системи

Для розробки системи було обрано трирівневу архітектуру. Вона включає в себе такі компоненти: клієнтський додаток, сервер додатку та сервер бази даних.

Структурну схему програмної системи зображено на рисунку 2.1.

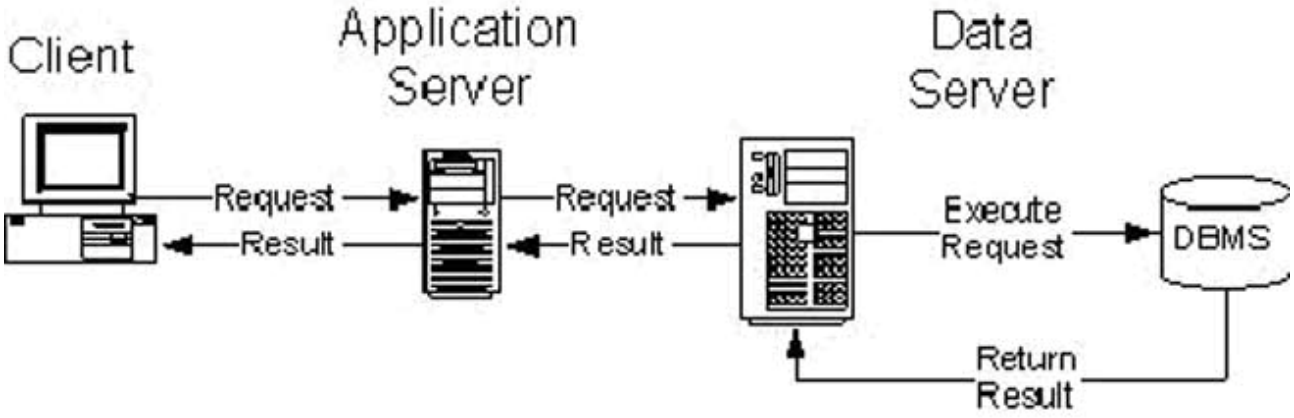

Рисунок 2.1 – Структурна схема програмної системи

Схема структурної організації такої програмної системи складається з трьох частин:

- клієнта, який відправляє запит, у відповідь отримуючи інформацію від сервера додатку (програмне забезпечення, яке реалізує передачу даних, кешування, користувацький інтерфейс та інші прості операції);
- сервера додатку, який отримує запити від клієнтів і інформацію від сервера бази даних, обробляє їх, здійснює необхідні операції, якщо необхідно – форматує в іншу форму;
- сервера бази даних, який здійснює запити від сервера додатку та надсилає йому результати.

Функціональну структуру системи умовно можна розділити на такі модулі:

- модуль обробки інформації про користувачів:

1) авторизація;

- модуль перегляду завдань:
  1) вибір завдання;
  2) редагування завдання;
  3) збереження змін у БД.
- модуль перегляду студентського маніфесту:
  1) вибір студента;
  2) редагування даних про студента;
  3) збереження змін у БД.

Для того, щоб краще зрозуміти процес функціонування системи побудовано діаграми станів для кожного модуля.

Діаграма станів авторизації користувача зображено на рисунку 2.2.

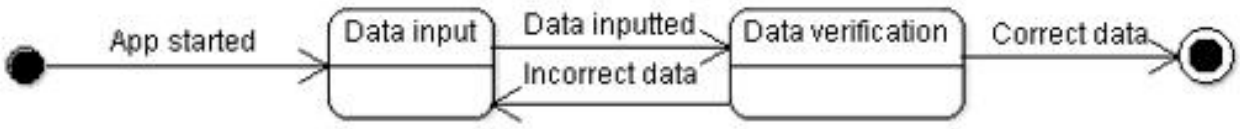

Рисунок 2.2 – Діаграма станів процесу авторизації користувача

Діаграму станів для перегляду списку завдань зображено на рисунку 2.3.

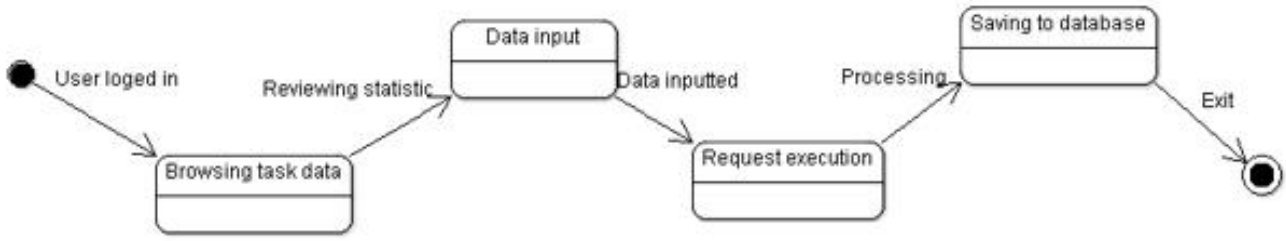

Рисунок 2.3 – Діаграма станів перегляду завдань

Діаграму станів для перегляду студентського маніфесту зображено на рисунку 2.4.

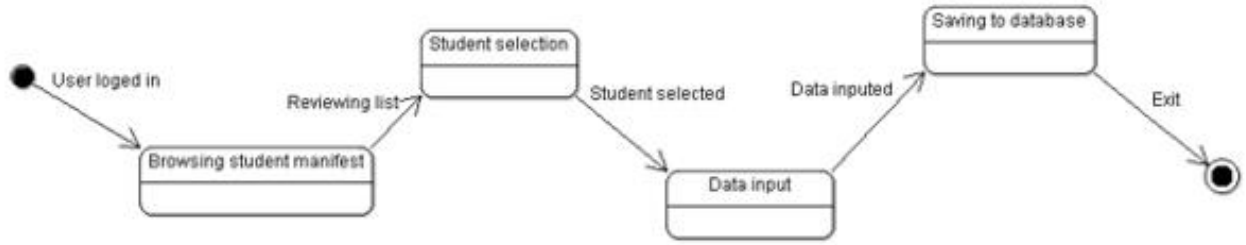

Рисунок 2.4 – Діаграма станів для перегляду студентського маніфесту

2.2. Проектування структури бази даних

Правильно спроектована база даних забезпечує доступ до найактуальнішої і точної інформації.

Є декілька принципів створення правильної структури бази даних. Перший принцип полягає в тому, що дублікат інформації (так звані надлишкові дані) це погано, тому що даремно витрачається пам’ять і збільшується ймовірність помилок і невідповідностей . Другий принцип полягає в тому, що правильність і повнота інформації має важливе значення. Якщо база даних містить неправильну інформацію, то будь-які звернення до неї теж будуть містити неправильну інформацію. Процес створення структури бази даних складається з наступних етапів:

Визначення мети бази даних - це допомагає підготувати вас для решти кроків.

Організація необхідної інформації – збір усіє необхідної інформації, яка буде використовуватись у системі.

Розділення інформації в таблицях – перетворення інформації у таблиці бази даних.

Перетворення елементів у стовпці таблиці – кожен елемент стає полем і відображається у вигляді стовпчика в таблиці. Наприклад, таблиця співробітників можуть включати в себе такі поля, як прізвище та дата.

Вкажіть первинні ключі – виберіть первинний ключ кожної таблиці. Первинний ключ являє собою стовпець, який використовується для

однозначної ідентифікації кожного рядка. Прикладом може бути ідентифікатор продукту або ID замовлення.

Налаштування відношень між таблицями – організація зв’язків між таблицями.

Аналіз структури – перевірка структури на наявність помилок. Внесення змін у міру необхідності.

Застосування правил нормалізації – застосування правил нормалізації даних для того, щоб побачити чи таблиці побудовані правильно. Внесення змін в таблиці в міру необхідності.

Проектування здійснюється за двома підходами. Перший підхід – визначення основних задач та створення для них бази. Другий підхід представляє з себе визначення предметної області та аналіз усіх даних. Об’єднання цих двох підходів є основним способом проектування структури бази даних.

Проектування бази даних слід розпочинати зі створення DFD-моделі предметної області (див. рис. 2.5).

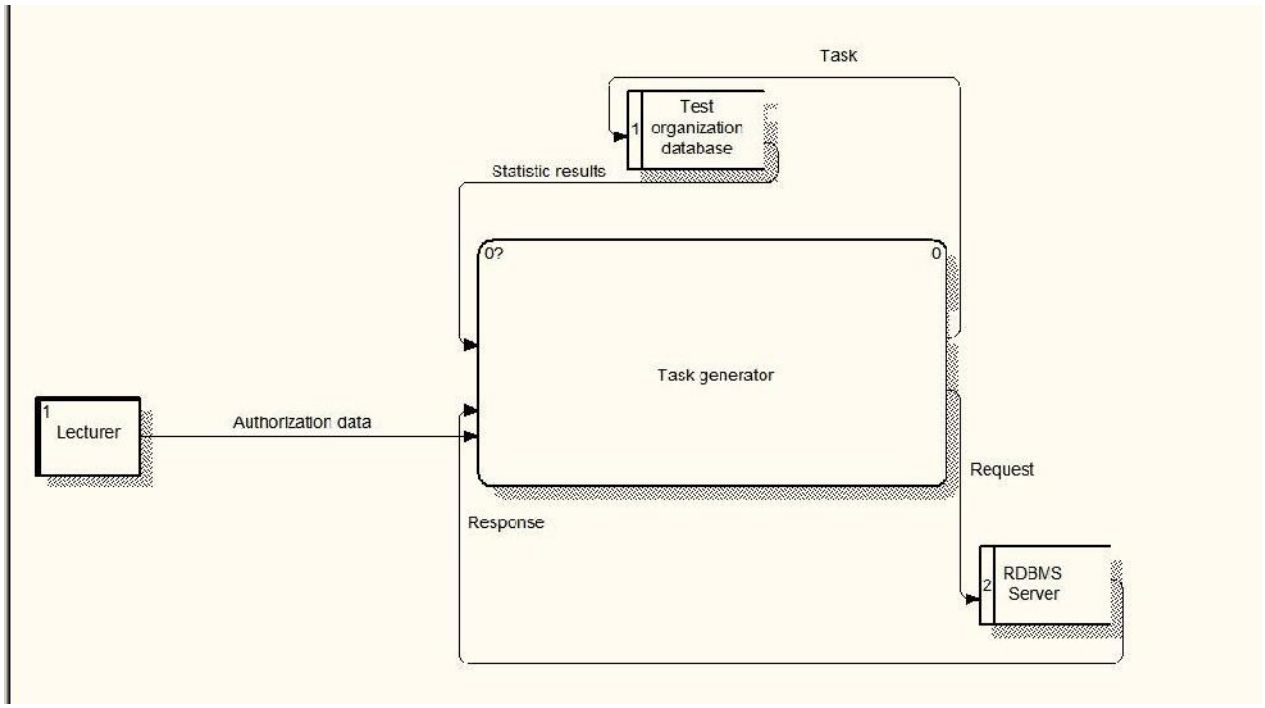

Рисунок 2.5 – Діаграма потоків даних

Діаграма декомпозиції зображена на рисунку 2.6.

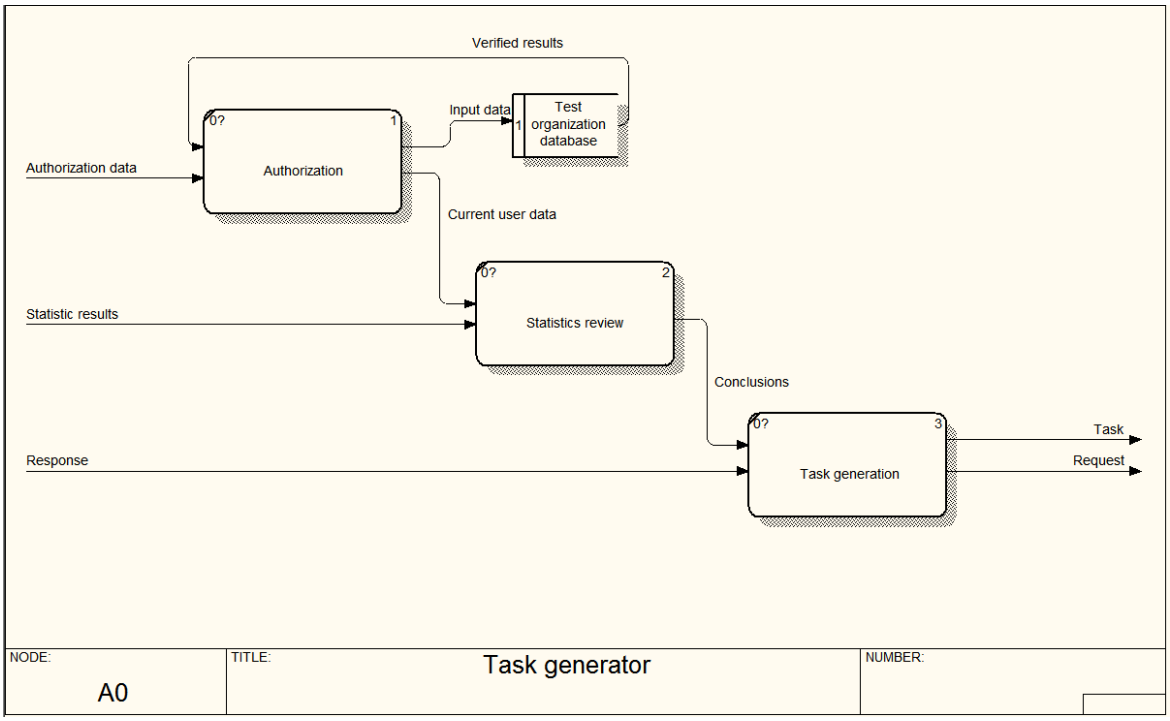

Рисунок 2.6 – Діаграма декомпозиції

Як показано на рисунку 2.6 робота програмної системи здійснюється наступним чином:

1. Входи (input), потоки, які поступають у систему і переробляються нею у вихідні величини, на діаграмі зображені ліворуч:

- ✓ Authorization data (дані для авторизації);
- ✓ Statistic results (статистика);
- ✓ Response (відповідь від БД).

2. Виходи (output), продукти діяльності системи, тобто результати перетворення вхідних величин тощо, на діаграмі зображені праворуч:

- ✓ Task (завдання);
- ✓ Request (запит у бд);

Процес створення завдання відбувається за допомогою викладача, який взаємодіє з програмною системою. Викладач авторизується, вводячи свої персональні дані. Далі, у нього є вибір з трьох пунктів меню: перегляд завдань, перегляд студентського маніфесту, перегляд статистики. При виборі перегляду завдань, викладач може створити або редагувати завдання. Щоб створити

завдання, викладач задає умову та прикладає у додатку файл з ERD діаграмою. Після успішного створення завдання усі дані записуються в базу даних.

Наступним кроком є виявлення основних сутностей та зв’язків між ними. Для цього створено діаграму елементів та зв’язків (див. рис. 2.7).

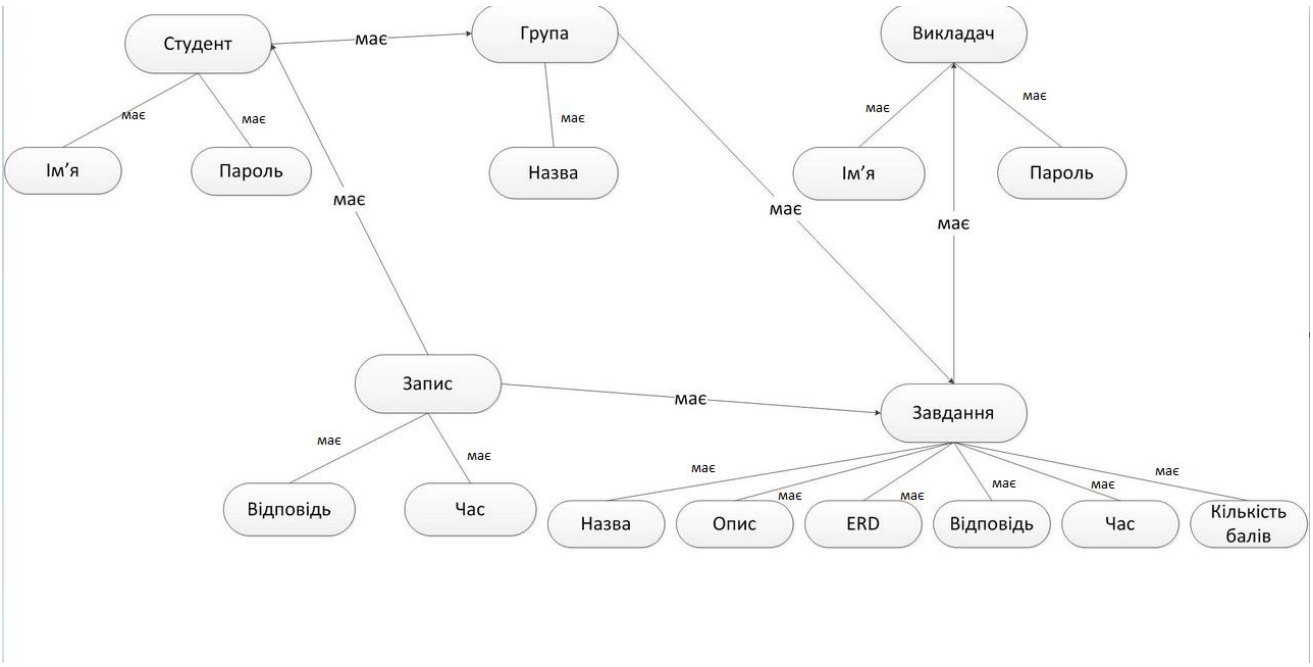

Рисунок 2.7 – Діаграма елементів та зв’язків

Після цього було створено таблицю ідентифікаторів, яку подано у таблиці 2.1.

Таблиця 2.1

Таблиця ідентифікаторів

Об’єкт	Властивість	Тип	Розмірність	Ідентифікатор
Студент	Ім’я Пароль	String String	20 16	Student Name Password
Група	Назва	String	20	Group Name
Викладач	Ім’я Пароль	String	20	Teacher Name

Продовження таблиці 2.1

		String	16	Password
Запис	Відповідь Час	 String String	 50 10	Record Answer Time
Завдання	Назва Опис ERD Відповідь Час Значення	 String String String String String Float	 30 300 50 100 20 20	Task Name Description ERD Answer Time Value
Студент	Ім’я Пароль	 String String	 20 16	Student Name Password
Група	Назва	 String	 20	Group Name

Наступним кроком буде планування індексів. Індекси – це спеціальні таблиці, які використовуються пошуковою системою бази даних для швидшого отримання шуканої інформації. Вони використовуються як вказівники на дані в таблицях. Використання індексів пришвидшують виконання запитів пошуку, однак запити додавання на редагування даних виконуються повільніше.

Всі зовнішні ключі являються індексами:

- group_id – зовнішній ключ таблиць Student та Task_Group;
- teacher_id – зовнішній ключ таблиці Task;
- task_id – зовнішній ключ таблиць Record і Task_Group;
- student_id – зовнішній ключ таблиці Record.

Також для оптимізації пошуку було вирішено індексувати наступні поля:

- value – поле важливості завдання в таблиці Task;
- name – поле імені студента в таблиці Student.

Наступним кроком є створення діаграми класів UML (див. рис. 2.8).

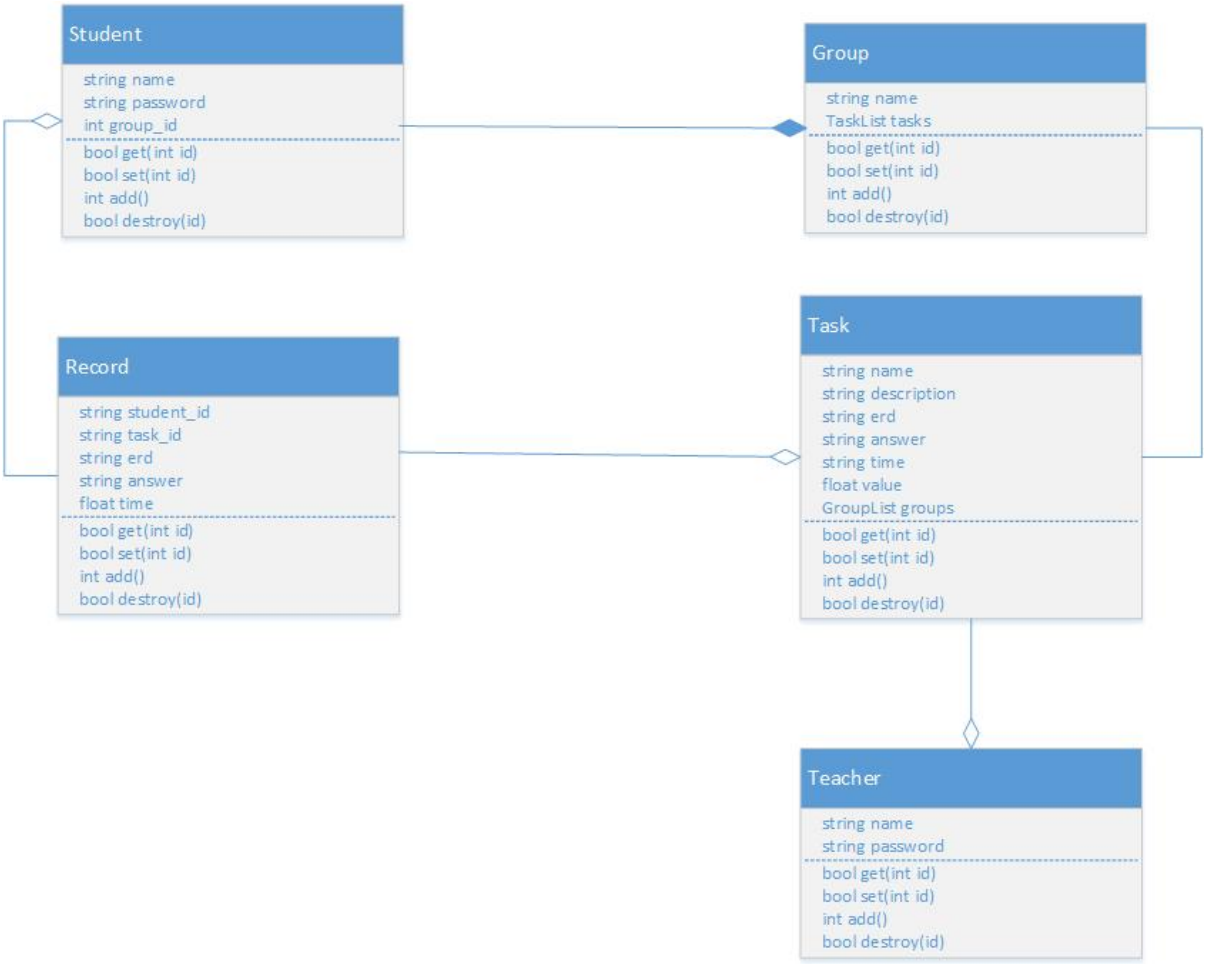

Рисунок 2.8 – Діаграма класів UML

Для проектування бази даних використовується діаграма “сутність-зв'язок” (ERD). Вона показує основні об’єкти(сутності) та властивості(атрибути) та зв’язки між ними.

Тому на підставі аналізу предметної області у роботі спроектовано структуру бази даних у вигляді ERD діаграми (див. рис. 2.9).

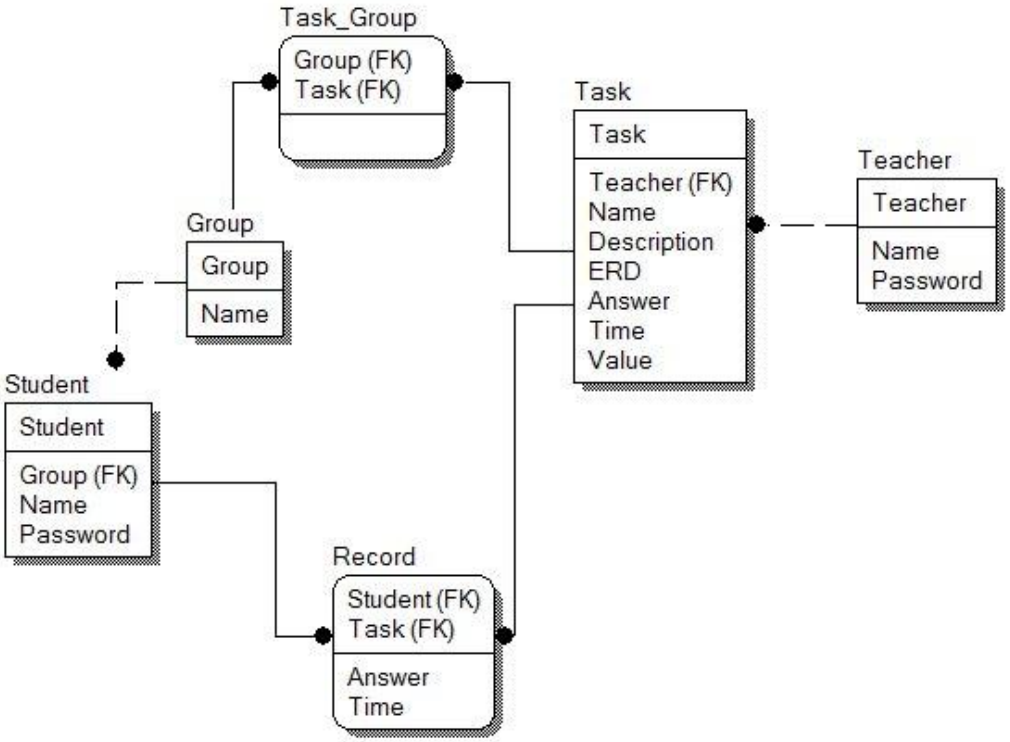

Рисунок 2.9 – ERD-діаграма

Проектування структури бази даних дозволяє визначити та повністю описати всі відношення та поля і є завершальною ланкою перед етапом програмної реалізації бази даних.

Висновок до другого розділу

Спроектовано підсистему “Викладач” програмної системи тестування студентів по SQL запитах, проведено аналіз технічної сторони виконання поставленого завдання. Розроблено архітектуру системи, здійснено проектування бази даних.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1. Програмна реалізація проекту

3.1.1. Обгрунтування вибору мови програмування

C# є елегантною об'єктно-орієнтованою мовою програмування з захищеними типами, яка дозволяє розробникам створювати безпечні і надійні додатки, які працюють на платформі .NET Framework. C# можна використовувати для створення додатків для Windows, веб-служб XML, розподілених компонентів, клієнт-серверних додатків, додатків баз даних і багато, багато іншого. Visual C# надає розширений редактор коду, зручні інструменти для створення дизайну користувацького інтерфейсу, вбудований відлагоджувач, і багато інших інструментів, які роблять його простим для розробки додатків на мові C# і .NET Framework.

Синтаксис мови C# є дуже виразним, але також досить простий і легкий в освоєнні. Синтаксис C# зменшує складність того, що було складним в C++ і надає потужні функції, такі як типи значень Nullable, перерахування, делегати, лямбда-вирази і прямий доступ до пам'яті, які не зустрічаються в Java. C# підтримує загальні методи і типи, які забезпечують більш високий рівень типу і продуктивності, а також ітератори, що дозволяють при реалізації колекцій класів визначати власну поведінку ітерації, яке може легко використовуватися в клієнтському коді. Вирази Language-Integrated Query (LINQ) роблять строго типізований запит зручною мовною конструкцією першого класу.

Як і всі об'єктно-орієнтовані мови програмування, C# підтримує інкапсуляцію, наслідування та поліморфізм. Всі змінні і методи інкапсулюються в визначення класів. Клас може наслідувати з одного батьківського класу, але підтримує безліч інтерфейсів. Для методів, які скасовують віртуальні методи в батьківському класі, необхідно ключове слово override, щоб виключити випадкове повторне визначення.

Дещо нова конструкція мови C# спрощує розробку компонентів програмного забезпечення. Така конструкція включає в себе такі особливості, як:

- делегати – підтримують повідомлення про події;
- властивості, що виступають в ролі методів доступу для закритих змінних;
- атрибути з декларативними метаданими;
- вбудовані коментарі XML-документації;
- LINQ, що забезпечує можливості запитів в різних джерелах даних.

Для забезпечення взаємодії з іншим програмним забезпеченням Windows, таким як об'єкти COM або власні бібліотеки DLL Win32, в мові C# існує процес, який називається "Interop". Процес Interop дозволяє програмам на C# виконувати практично будь-які дії, які може виконувати оригінальна програма на C++. Мова C# підтримує вказівники і поняття "небезпечного" коду для тих випадків, коли прямий доступ до пам'яті має важливе значення.

У вихідному файлі C# може бути визначено будь-яке число класів, структур, інтерфейсів і подій.

Подібно Java, C# не підтримує множинне успадкування. Замість цього використовується рішення Java – інтерфейси. Інтерфейси, реалізовані класом вказатують певні функції, які клас гарантовано реалізує. Інтерфейси дають змогу уникнути небезпек множинного успадкування, зберігаючи при цьому можливість дозволити багатьом класам реалізувати той же набір методів.

Іншою корисною особливістю C# є так званий “garbage collector”. Збирач сміття .NET Framework управляє виділенням і звільненням пам'яті для програми. При кожному створенні нового об'єкта середовище CLR виділяє пам'ять для об'єкта з керованої динамічно розподіленої пам'яті (купи). Поки в керованій купі є доступний адресний простір, середовище виконання продовжує виділяти простір для нових об'єктів. Але пам'ять має обмеження. В кінцевому рахунку, щоб звільнити деяку кількість пам'яті, збирач сміття повинен виконати процедуру очищення. Механізм оптимізації збирача сміття

визначає найкращий час для виконання збору, ґрунтуючись на вироблених виділеннях пам'яті. В ході виконання очищення збирач сміття відшукує в керованій купі об'єкти, які більше не використовуються додатком, і звільняє виділену для них пам'ять.

C#, як частина платформи .NET, компілюється в Microsoft Intermediate Language (MSIL), яка є схожою на байт-код Java. MSIL дозволяє C# бути незалежним від платформи і працює, використовуючи just-in-time компіляцію. Тому програми, що працюють під .NET отримують збільшену швидкість при повторному використанні. Крім того, оскільки інші мови, які становлять платформу .NET (включаючи VB і Cobol) компілюються в MSIL, для класів є можливість успадковуватися через мови. MSIL, так як і байткод, є тим, що дозволяє C# бути незалежною від платформи.

Потенціал мови C# є дуже великим, якщо використання платформи .NET є доцільним. C# розроблений для того, щоб використати всі особливості та переваги платформи .NET на повну.

Система тестування знань студентів складається з двох модулів: модуля “Викладач” та модуля “Студент”, які пов’язані між собою Системою управління базами даних (СУБД). Кожен модуль програми можна розглядати як окремий додаток з клієнт-серверною архітектурою, оскільки він побудований на основі постійного зв’язку з СУБД. Загалом в системі фігурують дві бази даних: база даних для організації тестування та база даних для якої було створено завдання, в діалозі з якою студент буде здійснювати поставлені завдання. Завдання модуля “Викладач” полягає в створенні тестових завдань, відслідковування результатів тестування та конфігурування студентського маніфесту. Даний модуль було розроблено за допомогою мови програмування C# на платформі .Net. Вхідними для модуля даними є статистичні записи з усіма спробами студентів виконати завдання. Джерелом всіх вхідних даних являється база даних. Вихідними для модуля даними є список груп, студентів, завдань.

Система складається з бібліотек, форм, класів та файлу конфігурації. Кожна форма представляє вікно інтерфейсу користувача та функціонал який стоїть за ним. Загалом модуль містить такі бібліотеки:

1. Бібліотека DataLib — здійснює роботу з базами даних, містить в собі класи DB.cs, ManageDB.cs та відповідники сутностей в базі даних. За допомогою класу DB.cs відбувається з'єднання з базою даних, в якій буде здійснюватися тестування. Лістинг коду класу:

```
namespace DataLib
{
  public class DB

   {

SqlConnection myConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["DBConnection"].ConnectionString);

    SqlCommand comm = new SqlCommand();
    public DataTable Read(String SQL)
    {
      myConnection.Open();
      comm.Connection = myConnection;
      comm.CommandText = SQL;
      using (SqlDataReader reader = comm.ExecuteReader())
      {
        if (reader.HasRows)
        {
          DataTable dt = new DataTable();
          dt.Load(reader);
          myConnection.Close();
          return dt;
        }
        else
        {
          throw new Exception("NOT SELECT");
        }
      }
    }
```

```
    }
}
```

Код класу ManageDB.cs загалом схожий з кодом DB.cs та відрізняється використанням іншої стрічки підключення. За допомогою класу ManageDB.cs відбувається підключення до бази даних організації тестування.

Клас Student.cs дозволяє здійснювати базові операції додавання, зчитування, редагування та видалення даних з таблиці (CRUD). Цей клас містить поля відповідні з полями таблиці в базі даних. Лістинг коду цього класу міститься в додатку А.

Властивості цього класу:

string name — поле імені студента

string password — поле пароля студента

int group_id — поле id групи

Основні методи цього класу:

bool get(int id) — метод зчитування даних з таблиці в клас за заданим id

bool set(int id) — метод запису даних в таблицю за заданим id

int add() — метод додавання нового запису в таблиці з даними цього класу, повертає id запису

bool destroy(id) — метод видалення таблиці за заданим id запису

Клас Teacher.cs є відповідником таблиці в базі даних та дозволяє здійснювати базові операції додавання, зчитування, редагування та видалення даних. Цей клас містить поля відповідні з полями таблиці в базі даних. Лістинг коду цього класу міститься в додатку А.

Властивості цього класу:

string name — поле імені викладача

string password — поле пароля викладача

Основні методи цього класу:

bool get(int id) — метод зчитування даних з таблиці в клас за заданим id

bool set(int id) — метод запису даних в таблицю за заданим id

int add() — метод додавання нового запису в таблиці з даними цього класу, повертає id запису

bool destroy(id) — метод видалення таблиці за заданим id запису

Клас Group.cs є відповідником таблиці в базі даних та дозволяє здійснювати базові операції додавання, зчитування, редагування та видалення даних з таблиці. Цей клас містить поля відповідні з полями таблиці. Лістинг коду цього класу міститься в додатку А.

Властивості цього класу:

string name — поле назви групи

TaskList tasks — список завдань, приписаних до групи

Основні методи цього класу:

bool get(int id) — метод зчитування даних з таблиці в клас за заданим id

bool set(int id) — метод запису даних в таблицю за заданим id

int add() — метод додавання нового запису в таблиці з даними цього класу, повертає id запису

bool destroy(id) — метод видалення таблиці за заданим id запису

Клас Task.cs є відповідником таблиці в базі даних та дозволяє здійснювати базові операції додавання, зчитування, редагування та видалення даних з таблиці. Цей клас містить поля відповідні з полями таблиці. Лістинг коду цього класу міститься в додатку А.

Властивості цього класу:

string name — поле назви завдання

string description — поле опису завдання

string erd — поле шляху до зображення erd

string answer — зашифроване поле результату правильного запиту як відповіді на завдання

string time — значення часу, назначеного на виконання задвання

float value — значення кількості балів оцінки виконання завдання

GroupList groups — список груп, яким приписане це завдання

Основні методи цього класу:

bool get(int id) — метод зчитування даних з таблиці в клас за заданим id

bool set(int id) — метод запису даних в таблицю за заданим id

int add() — метод додавання нового запису в таблиці з даними цього класу, повертає id запису

bool destroy(id) — метод видалення таблиці за заданим id запису

Клас Record.cs є відповідником таблиці в базі даних та дозволяє здійснювати базові операції додавання, зчитування, редагування та видалення даних з таблиці. Цей клас містить поля відповідні з полями таблиці. Лістинг коду цього класу міститься в додатку А.

Властивості цього класу:

string student_id — поле id студента

string task_id — поле id завдання

string erd — поле шляху до зображення erd

float time — поле часу, витраченого на виконання завдання

string answer — зашифрована стрічка запиту

Основні методи цього класу:

bool get(int id) — метод зчитування даних з таблиці в клас за заданим id

bool set(int id) — метод запису даних в таблицю за заданим id

int add() — метод додавання нового запису в таблиці з даними цього класу, повертає id запису

bool destroy(id) — метод видалення таблиці за заданим id запису

Бібліотека Hash — бібліотека, яка відповідає за хешування даних алгоритмом SHA1. Ця бібліотека містить один клас HashSum.cs. Лістинг цього класу наведено нижче:

```
public class HashSum
   {
      private DataTable dt;
      public HashSum(DataTable dt, String secret = "PZS")
      {
         if (dt.TableName == "")
         {
```

```
        dt.TableName = secret;
    }
    this.dt = dt;
}
public HashSum(SqlDataReader reader, String secret = "PZS")
{
    DataTable dt = new DataTable();
    dt.Load(reader);
    dt.TableName = secret;
    this.dt = dt;
}
private byte[] SerializeData()
{
    // Serialize the table
    DataContractSerializer serializer = new DataContractSerializer(typeof(DataTable));
    MemoryStream memoryStream = new MemoryStream();
    XmlWriter writer = XmlDictionaryWriter.CreateBinaryWriter(memoryStream);
    serializer.WriteObject(memoryStream, this.dt);
    byte[] serializedData = memoryStream.ToArray();
    return serializedData;
}
private byte[] CalculateHashValue(byte[] SerializedData)
{
    // Calculte the serialized data's hash value
    SHA1CryptoServiceProvider SHA = new SHA1CryptoServiceProvider();
    byte[] hash = SHA.ComputeHash(SerializedData);
    return hash;
}

private String GetHash(byte[] hash)
{
    return Convert.ToBase64String(hash);
}
public String GetHashString()
{
    byte[] serializedData = this.SerializeData();
    byte[] hashValue = this.CalculateHashValue(serializedData);
    return this.GetHash(hashValue);
```

}

Конфігураційний файл містить в собі стрічки підключення до баз даних.

3.1.2. Вимоги до апаратного забезпечення

Мінімальні вимоги до сервера : комп'ютер з тактовою частотою процесора від 1 ГГц, 512 Мб оперативної пам'яті. Підтримується процесори як x86, так і x64 розрядності.

Рекомендовані вимоги до сервера : комп'ютер з тактовою частотою процесора від 2 ГГц, 1 ГБ оперативної пам'яті.

Для роботи системи необхідно наступне встановлене програмне забезпечення: Microsoft .Net Framework 3.5, MS SQL Server, Microsoft Visual C++ Redistributable x86.

Для роботи з клієнтською частиною системи потрібний комп'ютер, підключений по протоколу TCP/IP до мережі.

Мінімальні вимоги до комп'ютера клієнта: Pentium III 1 ГГц і вище, 512 Мб оперативної пам'яті. ОС Windows 7.

3.2. Програмна реалізація бази даних

Засобом для створення таблиць та подальшої реалізації бази даних було обрано Microsoft SQL Server Management Studio 2014. Цей програмний продукт використовується для роботи в СУБД Microsoft SQL Server 2014 та дозволяє здійснювати управління, налаштування та адміністрування компонентів. Поєднання середовищ Microsoft Visual Studio, Microsoft та SQL Server Management Studio забезпечує високий рівень підтримки та сумісності розробки БД та програмних систем, оскільки ці програмні продукти розроблені з умовою хорошої інтеграції та взаємодії між собою.

SQL Server Management Studio являє собою розширену утиліту для управління об’єктами в SQL серверу та об’єднує зручний графічний інтерфейс та потужні можливості скриптингу. Важливою компонентою системи є Object Explorer,

який забезпечує ієрархічний користувацький інтерфейс для перегляду та управління об’єктів.

Програмна система тестування знань студентів з SQL запитів передбачає розгортання двох баз даних: бази даних організації тестування викладачем та база даних для виконання тестування студентом. Перша база даних містить чітко виражену структуру, в той чай як друга база даних обирається на вибір викладача. Головним та єдиним обмеженням є необхідність створення завдання викладачем та виконання його студентом в одній і тій ж базі даних. В якості такої бази даних було використано стандартний шаблон AdventureWorksDW2014, який надається компанією Microsoft для вивчення функціоналу з тестовими даними. Він містить безліч таблиць та записів які ідеально підходять для проведення тестування, оскільки дає змогу створити безліч завдань, які не будуть пересікатись.

Отже, в базі даних організації тестування повинні бути реалізовані таблиці Group, Student, Teacher, Task, Record, Task_Group.

Спочатку створимо нову базу даних. Для цього в засобі SQL Server Management Studio, використовуючи Object Exporer, потрібно натиснути праву клавішу миші на підрозділі Databases поточного зв’язку з SQL Server та обрати варіант ‘New Database’. Ця послідовність зображена на рисунку 3.1. Ця функція дозволяє створювати нові бази даних для подальшої роботи з ними. Інші варіанти можуть передбачати імпорт/експорт та відновлення баз даних із файлів чи інших джерел.

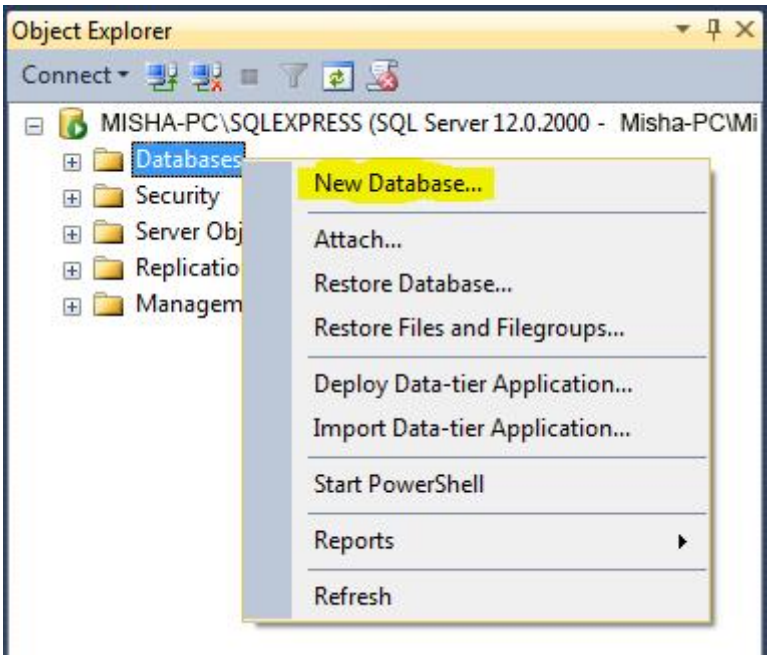

Рисунок 3.1 – Створення нової бази даних

Результат приведе до виклику вікна конфігурації об’єкту нової бази даних як на рисунку 3.2. Необхідно ввести назву та підтвердити виконання.

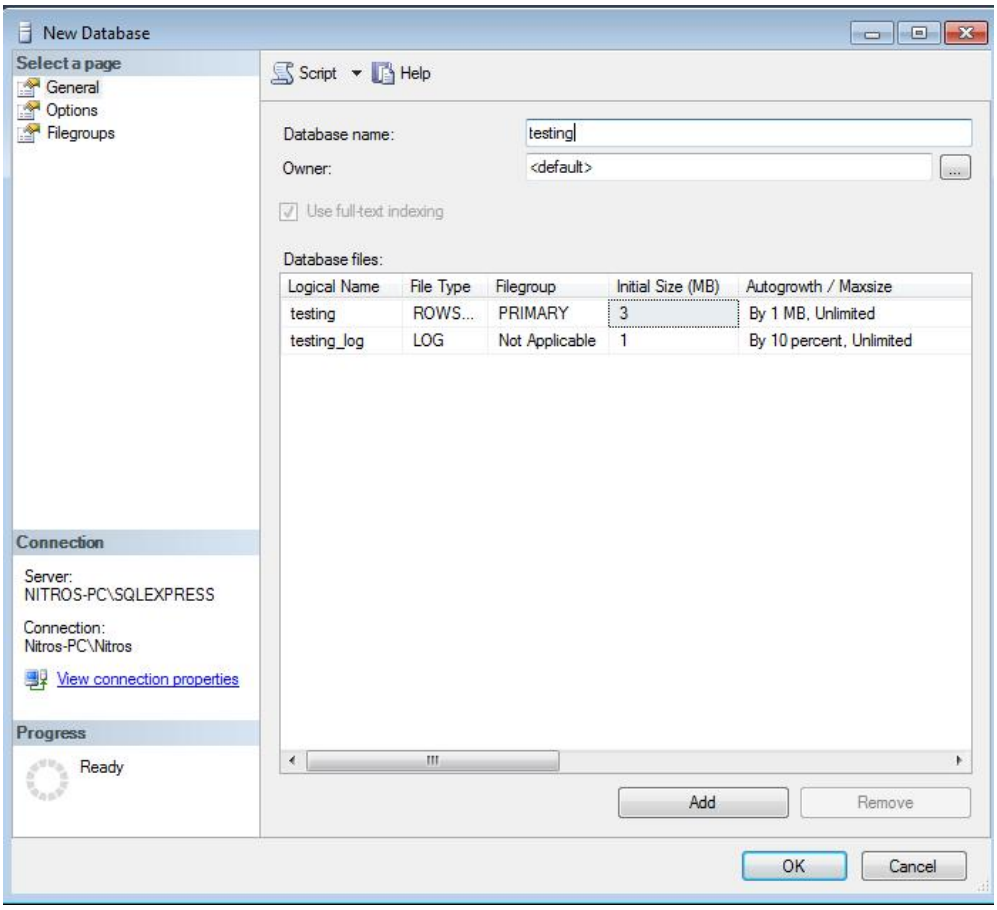

Рисунок 3.2 – Вікно налаштування нового об’єкту бази даних

В результаті буде отримано та виконано такий SQL-скрипт:

```
USE [master]
GO
CREATE DATABASE [testing]
CONTAINMENT = NONE
ON  PRIMARY
GO
USE [testing]
GO
```

Виконання цього скрипта призведе до створення нової бази даних Testing та дасть змогу використовувати наступні скрипти уже в її маштабах.

Наступним кроком буде створення таблиці Group. Для цього викличем редактор скриптів за допомогою комбінацій клавіш Ctrl+N. На рисунку 3.2 зображено вікно редактора скриптів.

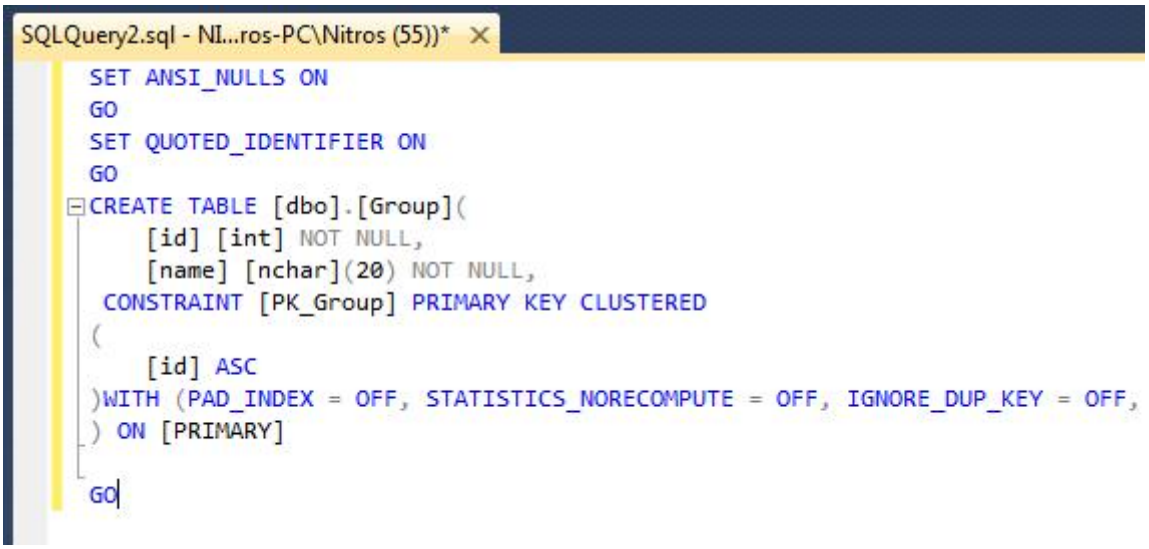

Рисунок 3.2 – Вікно редактора скриптів

Введемо та виконаєм наступний SQL-скрипт:

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Group](
        [id] [int] NOT NULL,
        [name] [nchar](20) NOT NULL,
 CONSTRAINT [PK_Group] 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
```

Виконання цього скрипта призведе до створення таблиці Group з первинним ключем id та з полями і типами цієї сутності згідно з таблицею ідентифікаторів та ERD.

Наступним буде створення таблиці Student. Використано такий SQL-запит:

```
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
```

Виконання цього скрипта призведе до створення таблиці Student з первинним ключем id та з полями і типами цієї сутності згідно з таблицею ідентифікаторів та ERD.

Тепер буде створено таблицю Teacher. Ця таблиця буде використовуватись в іншому модулі програмної системи тестування знань студентів, однак оскільки база даних є спільною створення такої таблиці є обов’язковим. Буде використано такий SQL-скрипт:

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Teacher](
```

```
	[id] [int] NOT NULL,
	[name] [nchar](20) NOT NULL,
	[password] [nchar](16) NOT NULL,
 CONSTRAINT [PK_Teacher] 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
```

Виконання цього скрипта призведе до створення таблиці Teacher з первинним ключем id та з полями і типами цієї сутності згідно з таблицею ідентифікаторів та ERD.

Далі створим таблицю Record. Буде використано такий SQL-скрипт:

```
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
```

Виконання цього скрипта призведе до створення таблиці Record з первинним ключем id та з полями і типами цієї сутності згідно з таблицею ідентифікаторів та ERD.

Далі створим таблицю Task_Group. Вона буде виступати в якості проміжної в зв’язку many-to-many між таблицями Task та Group. Буде використано такий SQL-скрипт:

```
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Task_Group](
	[id] [int] NOT NULL,
	[group_id] [int] NOT NULL,
	[task_id] [int] NOT NULL,
 CONSTRAINT [PK_Group_Task] 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
```

Виконання цього скрипта призведе до створення таблиці Task_Group з первинним ключем id та з полями і типами цієї сутності згідно з таблицею ідентифікаторів та ERD.

Наступним кроком буде створення зв’язків між таблицями. Виконаєм таку послідовність SQL команд:

```
ALTER TABLE [dbo].[Record] WITH CHECK ADD CONSTRAINT [FK_Record_Student] FOREIGN KEY([student_id]) REFERENCES [dbo].[Student] ([id]) GO
ALTER TABLE [dbo].[Record] CHECK CONSTRAINT [FK_Record_Student] GO
```

Цей скрипт виконується відносно таблиці Record та створює зв'язок зовнішнього ключа student_id з первинним ключем id таблиці Student. Це дозволить зв’язати ці дві таблиці між собою в відношенні one-to-many.

```
ALTER TABLE [dbo].[Record] WITH CHECK ADD CONSTRAINT [FK_Record_Task] FOREIGN KEY([task_id]) REFERENCES [dbo].[Task] ([id]) GO
ALTER TABLE [dbo].[Record] CHECK CONSTRAINT [FK_Record_Task] GO
```

Цей скрипт виконується відносно таблиці Record та створює зв'язок зовнішнього ключа task_id з первинним ключем id таблиці Task. Це дозволить зв’язати ці дві таблиці між собою в відношенні one-to-many.

```
ALTER TABLE [dbo].[Student] WITH CHECK ADD CONSTRAINT [FK_Student_Group] FOREIGN KEY([group_id]) REFERENCES [dbo].[Group] ([id]) GO
ALTER TABLE [dbo].[Student] CHECK CONSTRAINT [FK_Student_Group] GO
```

Цей скрипт виконується відносно таблиці Student та створює зв'язок зовнішнього ключа group_id з первинним ключем id таблиці Group. Це дозволить зв’язати ці дві таблиці між собою в відношенні one-to-many.

```
ALTER TABLE [dbo].[Task_Group] WITH CHECK ADD CONSTRAINT [FK_Task_Group_Group] FOREIGN KEY([group_id]) REFERENCES [dbo].[Group] ([id]) GO
ALTER TABLE [dbo].[Task_Group] CHECK CONSTRAINT [FK_Task_Group_Group] GO
ALTER TABLE [dbo].[Task_Group] WITH CHECK ADD CONSTRAINT [FK_Task_Group_Task] FOREIGN KEY([task_id]) REFERENCES [dbo].[Task] ([id]) GO
ALTER TABLE [dbo].[Task_Group] CHECK CONSTRAINT [FK_Task_Group_Task] GO
```

Цей скрипт виконується відносно таблиці Task_Group, яка являється проміжною в звязку many-to-many між таблицями Task та Group. Цей зв'язок забезпечується створенням двох зв’язків one-to-one за допомогою зовнішніх ключів task_id та group_id з первинними ключами id таблиць Task та Group відповідно.

Висновок до третього розділу

Обгрунтовано вибір мови програмування та додаткових програмних засобів, описно вимоги до програмного продукту, описана програмна реалізація системи з приведеними скриптами.

РОЗДІЛ 4. ТЕСТУВАННЯ ТА ДОСЛІДНА ЕКСПЛУАТАЦІЯ

4.1. Тестування

Для того, щоб перевірити програмний продукт на відповідність вимогам та впевнитись, що немає ніяких помилок було проведено тестування. Врахувавши специфіку розроблюваного продукту було складено та проведено наступні види тестування:

- функціональне тестування;
- GUI тестування;

Функціональне тестування дозволяє провести перевірку на відповідність продукту функціональним вимогам. Тести проводяться над усіма функціями програмної системи та Функціональне тести охоплюють всі розроблені функції, та розроблені з акцентом на тестування найбільш ймовірних типів помилки в програмній системі.

Для проведення функціонального тестування було розроблено тестові випадку у вигляді таблиці, що містить таку інформацію:

- № (номер тестового випадку);
- Description (опис об’єкту тестування);
- Steps to reproduce (кроки відтворення тестового випадку);

- Expected result (очікуваний результат тестування);
- Actual result (реальний результат тестування);
- Passed/Failed (вказується, чи пройдений тест, чи ні.);
- Environment (середовище, в якому здійснювалося тестування).

Було проведено 14 тестів, з них усі виявились успішними. Отже, усі функції програми відповідають функціональним вимогам.

Таблиця 4.1

Функціональні тестові випадки

Варіанти використання	**Тестові випадки**	**Тестові дані**
Authorization	1	5
View Task list	4	20

Продовження таблиці 4.1

View student manifest	1	1
View task results	1	1
Загалом	7	27

Таблиця 4.2

Специфікація тестів для функціонального тестування

	Тестові випадки	**Тестові дані**
1	Verify the authorization	5
2	Verify task list	1
3	Verify student manifest	1
4	Verify test results	1
5	Verify task edit	20
6	Verify task generation	20
7	Verify task deletion	1
8	Verify student information editor	20
9	Verify student information deletion	1
10	Verify ERD attach	5
	Загалом	75

GUI тестування

Для тестування графічного інтерфейсу користувача проводиться GUI тестування. Цей вид тестування має на меті перевірити відповідність роботи інтерфейсу користувача вимогам. Дії потенційного користувача відтворюються для перевірки очікуваного результату з реальним. Для цього було складено чек-ліст та проведено ручне тестування інтерфейсу. Так як програмною системою водночас буде користуватися дуже мала кількість користувачів, автоматизоване тестування в даному випадку є затратним та зайвим. Чек-ліст наведений у додатку Б.

Було проведено 12 тестів, з них усі виявились успішними. Отже, інтерфейс користувача організований вірно та відповідає заявленим вимогам.

Таблиця 4.3

GUI тестові випадки

Варіанти використання	**Тестові випадки**	**Тестові дані**
Authorization	1	5
View Task list	5	20
View student manifest	5	1
View task results	1	1
Загалом	12	27

Таблиця 4.4

Специфікація тестів для GUI тестування.

	Тестові випадки	**Тестові дані**
1	Check -> “Authorization”	5
2	Check -> “Task list button”	1
3	Check -> “Student manifest button”	1
4	Check -> “Task results button”	1
5	Check -> “Add task button”	20
6	Check -> “Edit task button”	20
7	Check -> “Delete task button”	1
8	Check -> “Create task”	20

9	Check -> “Update task”	1
10	Check -> “Create student”	5
11	Check -> “Create group”	5
12	Check -> “Return button”	1
	Загалом	81

4.2. Розгортання програмного продукту

Рекомендується запускати на комп’ютерах з файловою системою NTFS, оскільки вона є більш захищеною ніж FAT32. Для роботи програмної системи необхідне таке апаратне та програмне забезпечення:

Системні вимоги до сервера:

1) Вимоги до апаратного забезпечення:

• Оперативна пам’ять - 8Гб;

• Об’єм дискового простору - 250Гб.

2) Вимоги до програмного забезпечення:

• Операційна система - OS Windows 7 і вище;

• .NET Framework 4.0;

• MS SQL Server 2013.

Системні вимоги до клієнта:

Вимоги до апаратного забезпечення:

• Оперативна пам’ять – 512 мб.

• Жорсткий диск 50 гб.

Вимоги до програмного забезпечення:

• Операційна система - OS Windows 7 і вище;

• .NET Framework 4.0.

4.3. Інструкція користувача

4.3.1. Компоненти ПЗ

Програмна система розроблена на мові програмування C# у середовищі Microsoft Visual Studio 2013 на платформі .NET і експлуатується під

управлінням ОС Windows. Усі класи програмної системи задокументовані інформаційно та семантично.

Для функціонування системи тестування необхідне встановлення на СУБД сервера двох баз даних, файли яких постачаються разом з програмною системою.

Набір файлів модуля «Студент» системи тестування знань студентів представлений в таблиці 4.5.

Таблиця 4.5

Набір файлів, необхідних для функціонування модуля «Викладач»

№	Файл	Призначення	Належить проекту
1	Hash.dll	Бібліотека для хешування паролів та запитів	Бібліотеки програмної системи тестування знань студентів
2	DataLib.dll	Бібліотека для з'єднання з базами даних та діалогу з ними	
3	SQLtesting.exe	Основний виконуваний файл додатку модуля «Викладач»	Модуль «Викладач» програмної системи тестування знань студентів
4	SQLtesting.config	Конфігураційний файл	

4.3.2. Встановлення ПЗ

Для роботи програмної системи необхідно встановити Microsoft SQL Server та Microsoft .NET Framework 4.

На стороні сервера, використовуючи СУБД Microsoft SQL Server 2014 розгорнути 2 бази даних: Testing та AdventureWorksDW2014. Для цього використати середовище Microsoft SQL Server Management Studio 2014, в якому

після підключення до СУБД необхідно обрати імпорт бази даних в панелі інструментів. Джерелом даних слід вказати файли баз даних, які надаються разом з програмною системою.

Бібліотеки та модуль «Викладач» програмної системи тестування знань студентів розмістити в одній папці та використовувати запускаючи виконуваний файл SQLtestvykl.exe.

4.3.3. Налаштування

Відповідно до створених баз даних, користувачів та налаштувань з'єднання, необхідно записати стрічки підключення до двох баз даних в файл конфігурації SQLtestvykl.config в стрічки параметри 'Testing' та 'AdventureWorks'.

4.3.4. Базові функції ПЗ

Для перегляду списку завдань в головному меню виберіть пункт – "Список завдань".

Для створення завдання в головному меню виберіть пункт – "Список завдань", потім нажміть на піктограму додавання завдання "  ".

Для редагування завдання в головному меню виберіть пункт – "Список завдань", потім нажміть на піктограму редагування завдання "  ".

Для видалення завдання в головному меню виберіть пункт – "Список завдань", потім нажміть на піктограму видалення завдання "  ".

Для перегляду студентського маніфесту в головному меню виберіть пункт – "Студентський маніфест".

Для перегляду результатів в головному меню виберіть пункт – "Результати".

4.3.5. Аналіз помилок

При виникненні помилок “Server connection timed out” та “Cannot connect to server” необхідно провірити з’єднання клієнта та сервера з мережею, активність СУБД сервера.

4.3.6. Організація інтерфейсу з користувачем

Схематично організацію взаємодії користувача і машини представлено на рисунку 4.1.

Рисунок 4.1 – Організація взаємодії комп'ютера і користувача

До теперішнього моменту склалися наступні типи діалогів: питання-відповідь, меню, екранна форма, командний код, пряме маніпулювання (графічний інтерфейс).

Діалог типу «питання-відповідь» або регламентований діалог є першим в історії взаємодії користувача і комп’ютера. При цьому виді діалогу запити користувача і відповіді системи мають наперед задану структуру і кодуються з використанням лінгвістичного забезпечення ІТ (ІС). Залежно від ситуації ініціатива ведення діалогу може належати комп'ютеру або користувачу.

Щось схоже на такий вид діалогу реалізовано в умовах обмежень на мову або предметну область. При обмеженнях на предметну область створюються 2 словника:

• Функціональний. Містить функціональні слова, які означають за змістом характер необхідної задачі: наприклад, питальні слова що, скільки, які, коли, а також імперативи знайти, визначити і т.д. Склад цих слів відповідає

функціональним можливостям системи. Дані слова є ключовими, і їх зміст не може змінюватися. Вони, як правило, пов'язані з відповідною процедурою або з іншими такими ж словами;

• Специфічний. Містить зовнішні імена полів БД, так звану професійну лексику користувачів системи. Ці слова вводяться при налаштуванні системи і відображають предметну область (наприклад, прізвище, номер залікової книги і т.д. - для навчальних закладів), а також жаргон (сленг) користувача. Вони також є ключовими в тому сенсі, що визначають подальшу обробку запиту.

Графічний інтерфейс (пряме маніпулювання) – стиль взаємодії людини з комп'ютером, який включає в себе безперервне подання об'єктів. На відміну від інших стилів взаємодії, наприклад, командної мови, суть прямого маніпулювання полягає в тому, щоб дозволити користувачеві маніпулювати об'єктами за допомогою певних дій. Прикладом прямого маніпулювання є зміна розміру графічної форми, наприклад, прямокутника, перетягуючи його кути або краї за допомогою миші.

Наявність реально існуючих метафор для об'єктів і дій можуть полегшити користувачеві вивчати і використовувати інтерфейс (можна сказати, що інтерфейс є більш природним або інтуїтивним). Швидкий зворотній зв'язок дозволяє користувачеві робити менше помилок і виконувати задачі за менший час, тому що вони можуть бачити результати дії до її завершення, таким чином, оцінюючи вірогідний кінцевий варіант і компенсуючи помилки.

Пряме маніпулювання тісно пов'язане з інтерфейсами, які використовують вікна, іконки, меню і вказівний пристрій (WIMP GUI), оскільки вони майже завжди включають пряме маніпулювання. Проте, прямі маніпуляції не слід плутати з іншими термінами, оскільки не мається на увазі використання вікон або графічного виведення. Наприклад, прямі маніпуляції можуть бути застосовані до інтерфейсів для сліпих користувачів, використовуючи комбінацію тактильних і звукових пристроїв та програмного забезпечення.

Крім того, можна спроектувати інтерфейс WIMP, який умисно не припускає використання прямого маніпулювання. Наприклад, більшість версій віконних інтерфейсів (наприклад, Microsoft Windows) дозволяють користувачам змінювати положення вікна, перетягуючи його за допомогою миші, але не перемальовуючи повне вікно на проміжних позиціях під час пересування. Замість того, наприклад, прямокутний контур вікна може малюватись під час переміщення, коли повний вміст вікна буде перемальований тільки після того, як користувач відпустив кнопку миші. Це було необхідно на старих комп'ютерах, які не відчували брак потужності пам'яті і/або процесора, щоб швидко перемалювати дані за вікном, що тягнеться, але більше не використовується за замовчуванням в нових версіях Microsoft Windows.

Нижче наведено інтерфейс програми з детальним описом усіх меню та функцій (див. рис. 4.2).

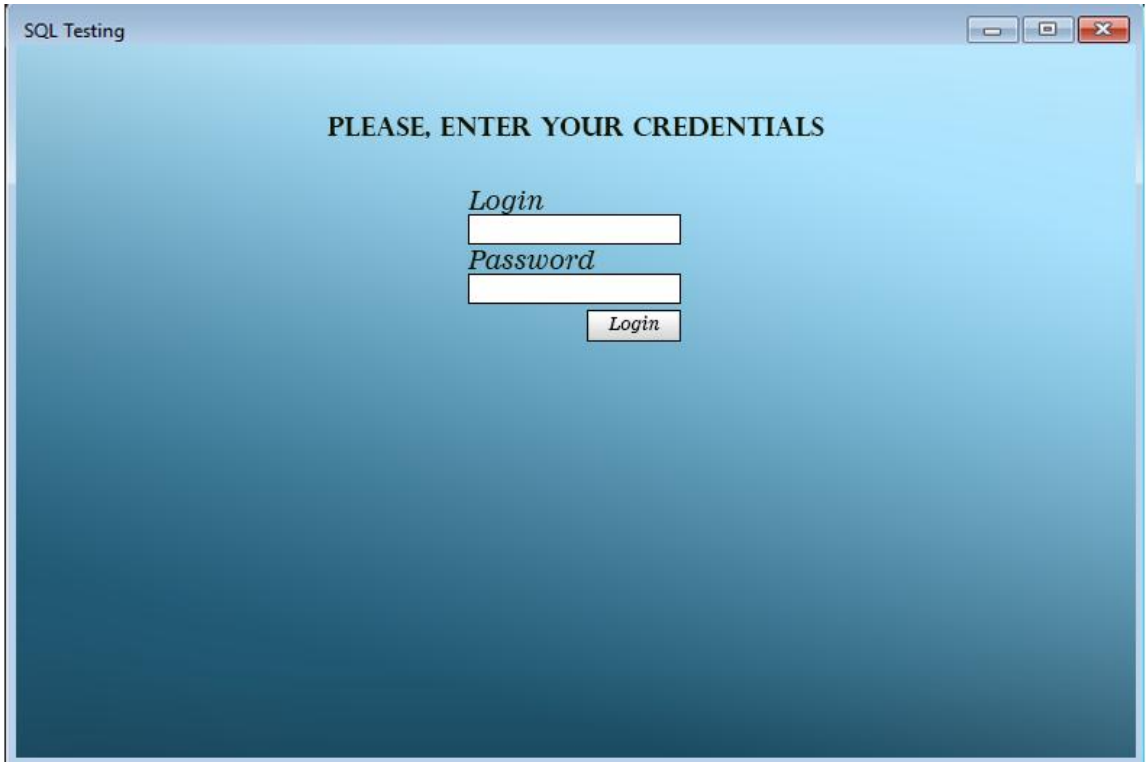

Рисунок 4.2 – Вікно авторизації

На даному вікні користувач може здійснити вхід у систему, увівши свої персональні дані у поля “Логін” та “Пароль”. Також є можливість завершення роботи програми при натисканні відповідної кнопки “Хрестик”.

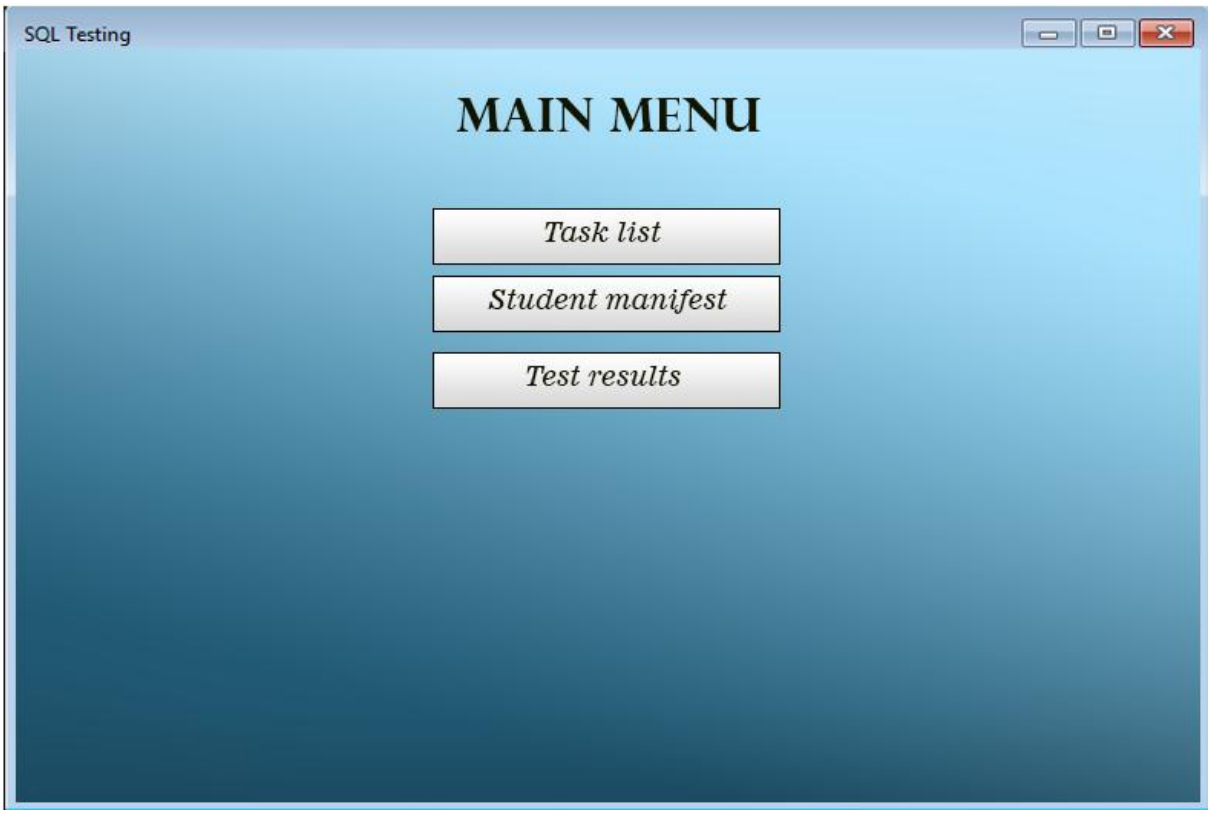

Рисунок 4.3 – Головне меню

У головному меню програми користувач може здійснити наступні дії: При виборі пункту “Перегляд завдань”, користувач зможе переглянути список завдань та здійснювати над ними певні операці; При виборі пунку “Перегляд студентського маніфесту” користувачу у новому вікні відкриється список студентів та груп; При виборі пункту “Перегляд результатів” користувачу у новому вікні відкриються результати тестів. Також користувач може виконати вихід з системи, натиснувши “хрестик”.

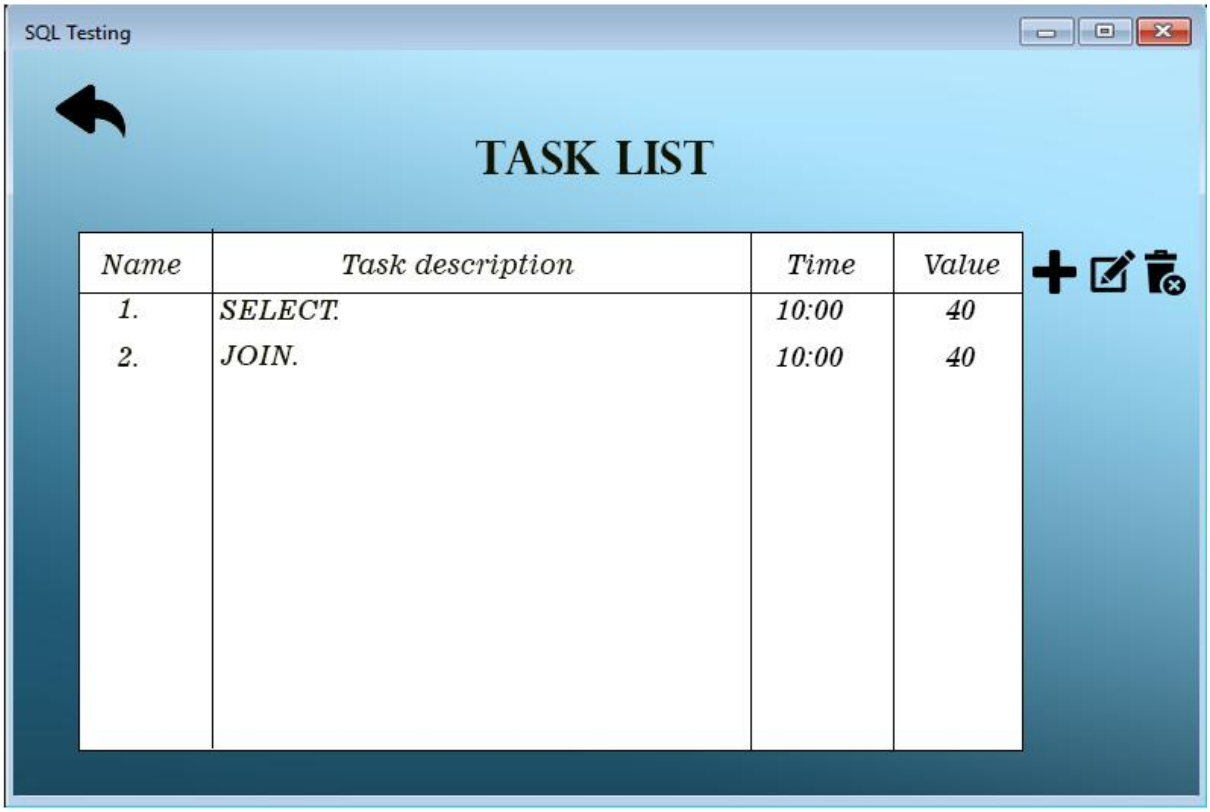

Рисунок 4.4 – Вікно перегляду списку завдань

У цьому вікні користувач бачить список завдань, зображений у таблиці. У правій стороні вікна є дві піктограми:

1. “ ” – додавання нового завдання;
2. “ ” – відкриває вікно редагування завдання;
3. “ ” – видаляє обране завдання зі списку.

Також, користувач може повернутися до попереднього меню натиснувши кнопку “ ”.

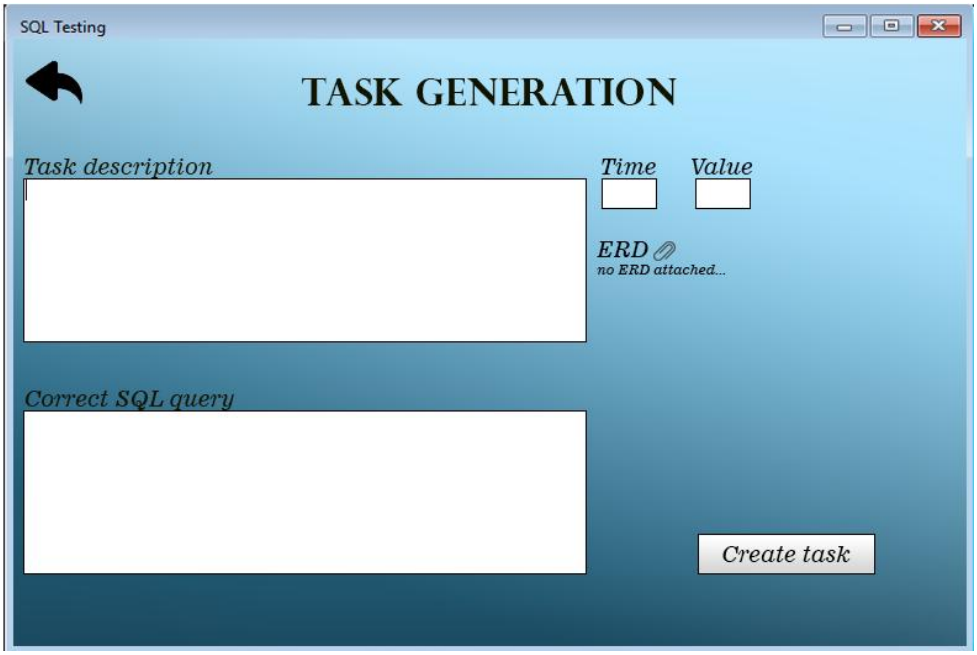

Рисунок 4.5 – Вікно створення завдання

Обравши функцію створення завдання, користувачу відкривається вікно зі створенням завдання. Тут, користувач може написати текст завдання у відповідній формі та завантажити ERD-діаграму. Також, користувач може повернутися до попереднього меню натиснувши кнопку “ ↰ ”.

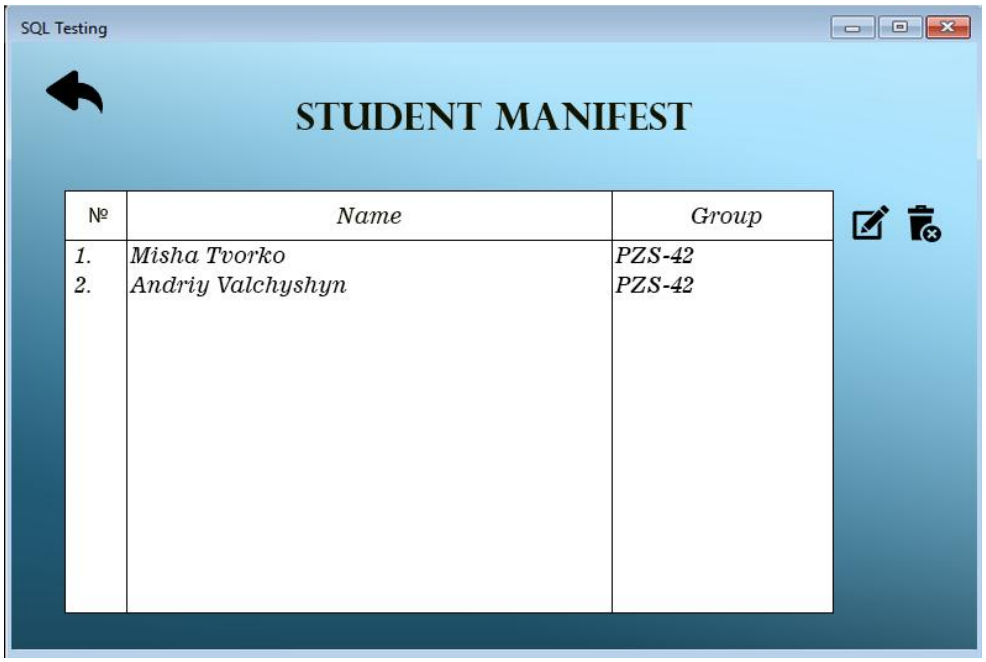

Рисунок 4.6 – Вікно перегляду студентського маніфесту

Тут, користувач може переглянути список студентів та груп.

Також, користувач може повернутися до попереднього меню натиснувши кнопку “ ”.

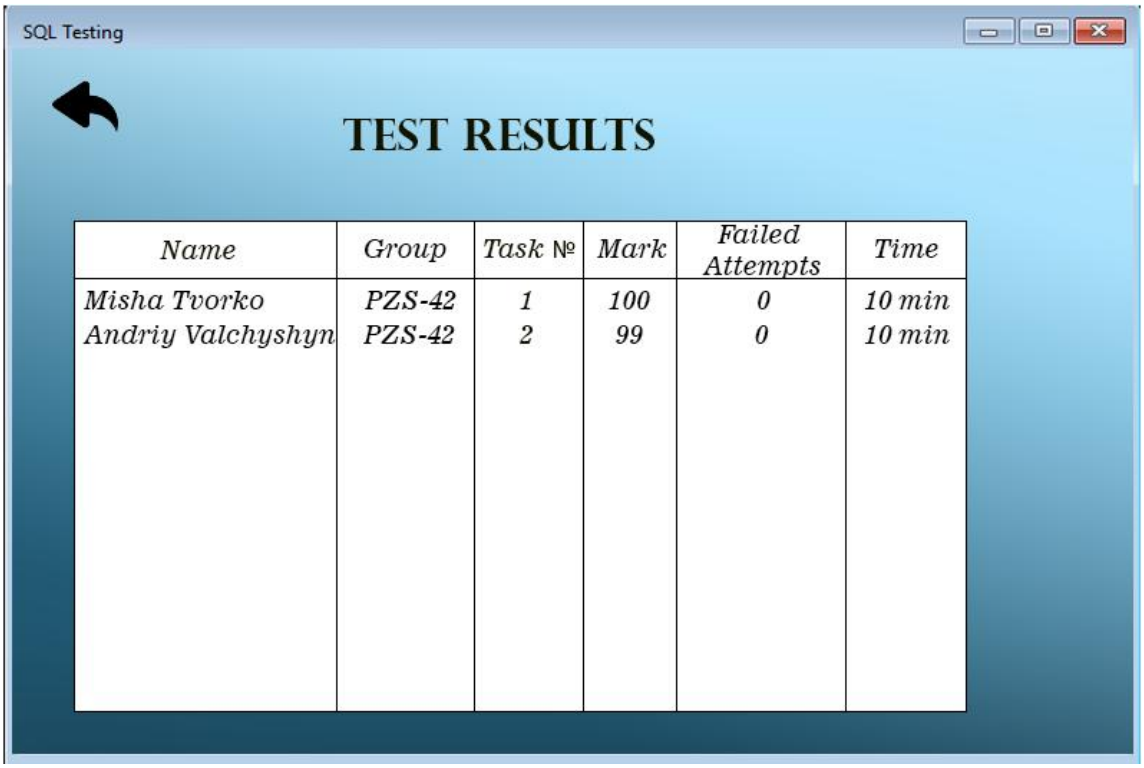

Рисунок 4.7 – Вікно перегляду результатів

Тут, користувач може переглянути результати тестувань. Також, користувач може повернутися до попереднього меню натиснувши кнопку “ ”.

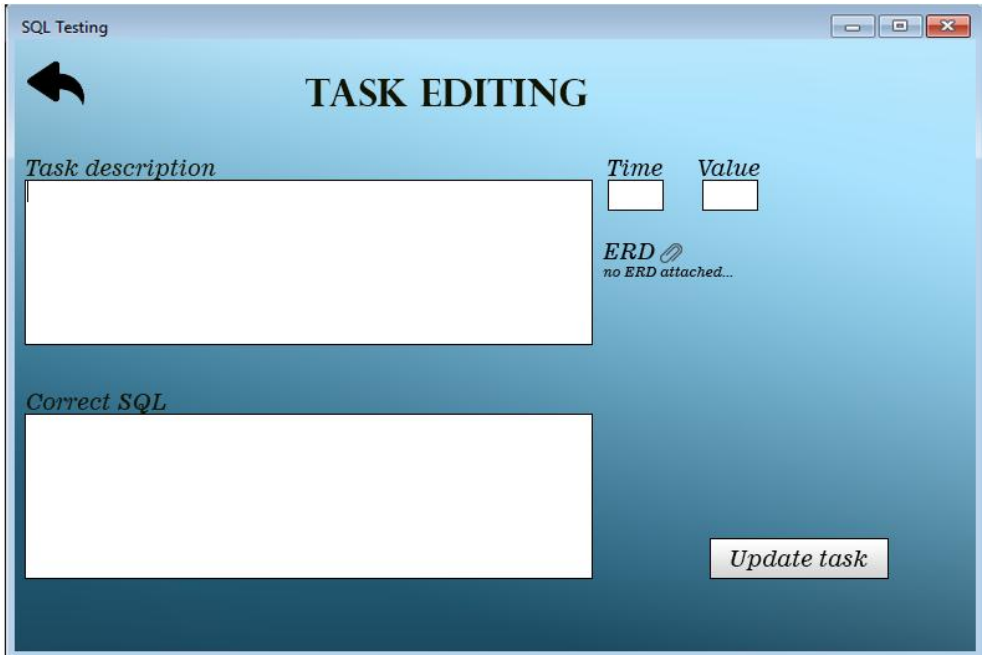

Рисунок 4.8 – Вікно редагування завдання

Обравши функцію редагування завдання, користувачу відкривається вікно із редагуванням завдання. Тут, користувач може написати текст завдання у відповідній формі та завантажити ERD-діаграму. Також, користувач може повернутися до попереднього меню натиснувши кнопку “ ↰ ”.

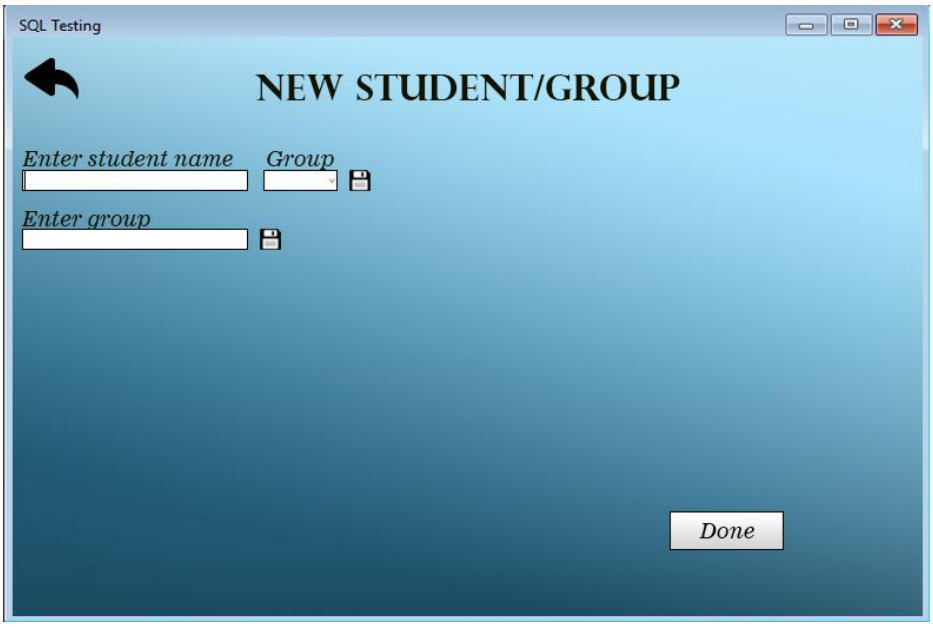

Рисунок 4.9 – Вікно додавання студента та групи

Обравши функцію додавання студента та групи, користувач може додати нову групу, студента з відповідною групою та зберегти інформацію у базу даних. Також, користувач може повернутися до попереднього меню натиснувши кнопку “ ↰ ”.

Висновок до четвертого розділу

Проведено аналіз програмного продукту та вибрано необхідні методи тестування. Розроблено тест-кейси та чек-листи, згідно з якими проводилось тестування програмної системи. Описано необхідне програмне та апаратне забезпечення, необхідне для розгортання програмного продукту, приведені назви необхідних файлів та їх опис. Створено інструкцію користувача та наведено вигляд інтерфейсу програми з детальним описом усіх функцій.

ВИСНОВОК

Тестування – важливий процес у визначенні рівня знань студентів. Можливість автоматизувати процес створення тестів для викладача має дуже велике значення, адже це дозволить зменшити затрати часу на підготовку тестових завдань, полегшить спостереження за процесом тестування, дасть змогу вести статистику завдань та слідкувати за результатами тестувань.

Мета розробленої системи полягає якраз у полегшенні здійснення тестування зі сторони викладача, а саме: створення завдань, відслідковування результатів, перегляд та організація студентського маніфесту.

Система проектувалась на основі досліджених трьох аналогів, програмної системи для тестування знань “Айрен”, “MyTestStudentPRO” та веб-системи “Exambuilder”. Було враховано усі їх недоліки та переваги та застосовано ці знання для розробки продукту.

Аріхектура програмної системи розроблена із повною відповідністю до поставлених вимог, здійснено аналіз потоків даних для подальшого проектування.

Система створена з використанням мови програмування C# у середовищі розробки Microsoft Visual Studio 2013. Оскільки додаток повинен містити базу даних, обрано реляційну модель бази даних. Ця модель забезпечує найвищу надійність збереження даних, та високу швидкодію при роботі з ними. Архітектуру бази даних реалізовано за допомогою Microsoft SQL Server та Microsoft Management Studio. Це пояснено тим, що ці програмні продукти добре інтегровані з іншими продуктами від Microsoft, тому це дозволило зробити проектування та програмування дуже простим та призвело до максимального зниження кількості помилок.

Тестування програмної системи не показало ніяких помилок. Функціональне тестування пройшло успішно, вказавши повну відповідність поставленим вимогам. Тестування графічного інтерфейсу не виявило помилок,

що означає хорошу організацію та оптимізацію користувацького інтерфейсу підсистеми “Викладач”.

Створено документацію для користувача, яка містить опис програмного продукту і його функцій. Розроблено інструкцію щодо розгортання програмного продукту, вказано вимоги до апаратного та програмного забезпечення. Наведено список файлів необхідних для коректної роботи програми та їх опис.

Отже, підсистема “Викладач” програмної системи тестування студентів по SQL запитах є надійною та зручною системою, яка дозволяє автоматизувати процес проведення тестувань, полегшує роботу викладача, забезпечує запис результатів та збереження їх у базі даних. Дозволяє організувати студентських маніфест. Система захищена від несанкціонованого доступу, тому тести та правильні відповіді до них не будуть втрачені чи змінені стороннім користувачем. Робота системи є швидкою, з мінімальною кількістю збоїв.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Джеймс Р. Грофф, Пол Н. Вайнберг, Эндрю Дж. Оппель. SQL: полное руководство, 3-е издание = SQL: The Complete Reference, Third Edition. — М.: «Вильямс», 2014. — 960 с.
2. Крис Фиайли. SQL: Руководство по изучению языка. — М.: Peachpit Press, 2003. — 456 с.
3. К. Дж. Дейт. Введение в системы баз данных / Пер. с англ. — 8-е изд. — М.: Вильямс, 2005. — 1328 с.
4. Codd, Edgar F (June 1970). "A Relational Model of Data for Large Shared Data Banks". Communications of the ACM 13 (6): 377–87.
5. C. J. Date with Hugh Darwen: A Guide to the SQL standard : a users guide to the standard database language SQL, 4th ed., Addison Wesley, USA 1997.
6. Лайза Криспин, Джанет Грегори Гибкое тестирование: практическое руководство для тестировщиков ПО и гибких команд = Agile Testing: A Practical Guide for Testers and Agile Teams. — М. : «Вильямс», 2010. — 464 с. — (Addison-Wesley Signature Series). — 1000 прим.
7. Канер Кем, Фолк Джек, Нгуен Енг Кек Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений. — Киев : ДиаСофт, 2001. — 544 с. — ISBN 9667393879.
8. Калбертсон Роберт, Браун Крис, Кобб Гэри Быстрое тестирование. — М. : «Вильямс», 2002. — 374 с.
9. Синицын С. В., Налютин Н. Ю. Верификация программного обеспечения. — М. : БИНОМ, 2008. — 368 с.
10. Бейзер Б. Тестирование чёрного ящика. Технологии функционального тестирования программного обеспечения и систем. — СПб. : Питер, 2004. — 320 с.

11. Beizer, Boris (1990). Software Testing Techniques (Second ed.). New York: Van Nostrand Reinhold. pp. 21,430. ISBN 0-442-20672-0.
12. IEEE (1990). IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. New York: IEEE. ISBN 1-55937-079-3.
13. ISO/IEC/IEEE 29119-1:2013 – Software and Systems Engineering – Software Testing – Part 1 – Concepts and Definitions; Section 4.38.
14. Rodríguez, Ismael; Llana, Luis; Rabanal, Pablo (2014). "A General Testability Theory: Classes, properties, complexity, and testing reductions". IEEE Transactions on Software Engineering 40 (9): 862–894. doi:10.1109/TSE.2014.2331690.ISSN 0098-5589.
15. Hershey, William; Easthope, Carol (1972). A set theoretic data structure and retrieval language. Spring Joint Computer Conference, May 1972. ACM SIGIR Forum 7 (4). pp. 45–55. doi:10.1145/1095495.1095500
16. Childs, David L. (1968). "Description of a set-theoretic data structure". CONCOMP (Research in Conversational Use of Computers) Project. Technical Report 3. University of Michigan.
17. Дивак М.П. Системний аналіз та проектування КІС /М.П.Дивак// Навчальний посібник – Т.: Економічна думка. – 2004.
18. Дейт К. Введение в системы баз даннях /К.Дейт// – К.; М.; СПб.: Изд.дом “Вильямс”. – 2000. –560 с.
19. Буч Г., Рамбо Дж., Джекобсон А. Язык UML. Руководство пользователя. — Пер. с англ. — М.: ДМК, 2000. — 432 с.
20. Фаулер М., Скотт К. UML. Основы. — Пер. с англ. — СПб: Символ-Плюс, 2002. — 192 с., ил. ISBN 5-93286-032-4
21. Никаноров, С. П. Системний аналіз: етап розвитку методології рішення. – 2001. – Выпуск 12. – С. 62–87. -фрагмент
22. Губанов В.А. і др. Введення в системний аналіз: Навчальний посібник /Под ред. Л.А. Петросяна. - Л.: Изд-во ЛГУ, 1988.

23. Клир Дж. Системологія. Автоматизація рішення системних задач.ер. с англ. – М.:1990.. – С. 62–87.

ДОДАТОК А

Лістинг коду програми

```
using DataLib;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Configuration;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Task_Generation
{
   public partial class AdminMainForm : Form
   {
      public AdminMainForm()
      {
         InitializeComponent();
         try
         {
            using (StreamReader sr = new StreamReader(ConfigurationManager.ConnectionStrings["DefaultTask"].ConnectionString))
            {
               String line = sr.ReadToEnd();
               var listTC = JsonConvert.DeserializeObject<List<TaskControl>>(line);
               foreach (var item in listTC)
               {
                  Tasklist.Items.Add(item.description);
```

```
            }

        }
    }
    catch (Exception exept)
    {
        MessageBox.Show(exept.Message);
    };
}

private void AdminMainForm_Load(object sender, EventArgs e)
{

}

private void AddTaskBtnClick(object sender, EventArgs e)
{
    Task_Gen TaskForm = new Task_Gen();
    if (TaskForm.ShowDialog() == DialogResult.OK)
    {
        Tasklist.Items.Clear();
        try
        {
            using (StreamReader sr = new StreamReader(ConfigurationManager.ConnectionStrings["DefaultTask"].ConnectionString))
            {
                String line = sr.ReadToEnd();
                var listTC = JsonConvert.DeserializeObject<List<TaskControl>>(line);
                foreach (var item in listTC)
                {
                    Tasklist.Items.Add(item.description);
                }
            }
        }
        catch (Exception exept)
        {
            MessageBox.Show(exept.Message);
        };
```

```
        }
    }

    private void DeleteBtn_Click(object sender, EventArgs e)
    {
        List<TaskControl> task = new List<TaskControl>();
        using (StreamReader sr = new StreamReader(ConfigurationManager.ConnectionStrings["DefaultTask"].ConnectionString))
        {
            String line = sr.ReadToEnd();
            var listTC = JsonConvert.DeserializeObject<List<TaskControl>>(line);
            foreach (var item in listTC)
            {
                task.Add(item);
            }
        }

        while (Tasklist.SelectedItems.Count > 0)
        {
            int index = Tasklist.SelectedIndex;
            Tasklist.Items.Remove(Tasklist.SelectedItems[0]);
            task.RemoveAt(index);
            string jstr = JsonConvert.SerializeObject(task, Formatting.Indented);
            System.IO.File.WriteAllText(@"D:\task.json", jstr);
        }
    }

    private void TaskListLabel_Click(object sender, EventArgs e)
    {

    }
  }
}


namespace SQLDev
{
  public partial class UserLogin : Form
```

```
{
    public UserLogin()
    {
        InitializeComponent();
        try
        {
            using (StreamReader sr = new StreamReader(ConfigurationManager.ConnectionStrings["Students"].ConnectionString))
            {
                String line = sr.ReadToEnd();
                var groupList = JsonConvert.DeserializeObject<List<GroupControl>>(line);
                foreach (var group in groupList)
                {
                    GroupCombo.Items.Add(group.groupName);
                }

            }
        }
        catch (Exception exept)
        {
            MessageBox.Show(exept.Message);
        };
    }

    private void LogInBtn_Click(object sender, EventArgs e)
    {
        if (String.IsNullOrWhiteSpace(GroupCombo.Text))
        {
            GroupStat.ForeColor = Color.Red;
            GroupStat.Text = "Choose group!";
        }
        else
        {
            if (String.IsNullOrWhiteSpace(StudCombo.Text))
            {
                StudStat.ForeColor = Color.Red;
                StudStat.Text = "Choose student!";
            }
```

```
            else
            {
                TaskMenu menuForm = new TaskMenu(StudCombo.SelectedItem.ToString() + ", " +
GroupCombo.SelectedItem.ToString());
                menuForm.FormClosed += new FormClosedEventHandler(menuForm_FormClosed);
                menuForm.Show();
                this.Hide();
            }
        }
    }

    private void menuForm_FormClosed(object sender, FormClosedEventArgs e)
    {
        this.Close();
    }

    private void GroupCombo_SelectedIndexChanged(object sender, EventArgs e)
    {
        GroupStat.Text = "";
        StudCombo.Items.Clear();
        try
        {
            using (StreamReader sr = new
StreamReader(ConfigurationManager.ConnectionStrings["Students"].ConnectionString))
            {
                String line = sr.ReadToEnd();
                List<GroupControl> groupList = JsonConvert.DeserializeObject<List<GroupControl>>(line);
                foreach (var stud in groupList[GroupCombo.SelectedIndex].students)
                {
                    StudCombo.Items.Add(stud.name);
                }
            }
        }
        catch (Exception exept)
        {
            MessageBox.Show(exept.Message);
        }
    }
```

```
        private void StudCombo_SelectedIndexChanged(object sender, EventArgs e)
        {
            StudStat.Text = "";
        }
    }
}


namespace DataLib
{
    public class GroupControl
    {
        public string groupName;
        public List<StudentControl> students;
    }
}

namespace DataLib
{
    public class TaskControl
    {
        public string description { get; set; }
        public string imgPath { get; set; }
        public string correctSQL { get; set; }
    }
}

namespace DataLib
{
    public class StudentControl
    {
        public string name { get; set; }
    }
}


namespace SQLDev
{
    public partial class TaskView : Form
```

```
    {

        public TaskView(int taskNum)
        {
            InitializeComponent();
            try
            {
                using (StreamReader sr = new StreamReader(ConfigurationManager.ConnectionStrings["DefaultTask"].ConnectionString))
                {
                    String line = sr.ReadToEnd();
                    var listTC = JsonConvert.DeserializeObject<List<TaskControl>>(line);
                    var currTask = listTC[taskNum];
                    DescLabel.Text = currTask.description;
                    ERDpictureBox.Image = new Bitmap(currTask.imgPath);
                    ERDpictureBox.SizeMode = PictureBoxSizeMode.Zoom;
                }
            }
            catch (Exception exept)
            {
                MessageBox.Show(exept.Message);
            };
        }

        private void TaskView_Load(object sender, EventArgs e)
        {

        }

        private void TaskView_SizeChanged(object sender, EventArgs e)
        {

        }
    }
}

namespace SQLDev
```

```
{
    public partial class TaskMenu : Form
    {
        private Boolean[] Marks;
        private Int32 TaskCount = 0;
        Button[] buttonArray;
        public TaskMenu(string student)
        {
            InitializeComponent();
            StudLab.Text += student;
        }

        private void TaskMenu_Load(object sender, EventArgs e)
        {
            this.TaskCount = getTaskCount();
            CreatingNewButtons(this.TaskCount);
            this.Marks = new Boolean[this.TaskCount];
            for (Int32 i = 0; i < Marks.Length; i++)
            {
                this.Marks[i] = false;
            }
        }

        private void CreatingNewButtons(int taskCount)
        {
            int horizotal = 13;
            int vertical = 65;
            buttonArray = new Button[taskCount];

            for (int i = 0; i < buttonArray.Length; i++)
            {
                buttonArray[i] = new Button();
                buttonArray[i].Name = "taskBtn_" + i;
                buttonArray[i].Size = new Size(65, 65);
                buttonArray[i].Location = new Point(horizotal, vertical);
                buttonArray[i].Text = "Task " + (i+1).ToString();
                buttonArray[i].Click += TaskBtn_Click;
                if ((i+1)%5 == 0)
```

```
            {
                horizotal = 13;
                vertical = vertical + 70;
            }
            else
            {
                horizotal = horizotal + 70;
            }
            this.Controls.Add(buttonArray[i]);
        }
    }

    private int getTaskCount()
    {
        try
        {
            using (StreamReader sr = new StreamReader(ConfigurationManager.ConnectionStrings["DefaultTask"].ConnectionString))
            {
                String line = sr.ReadToEnd();
                var taskList = JsonConvert.DeserializeObject<List<TaskControl>>(line);
                return taskList.Count;
            }
        }
        catch (Exception exept)
        {
            MessageBox.Show(exept.Message);
            return 0;
        };
    }

    private void  TaskBtn_Click(object sender, EventArgs e)
    {
        string[] words = (((Button)sender).Name).Split('_');

        MainForm taskForm = new MainForm(Convert.ToInt32(words[1]));
        taskForm.Text += (Convert.ToInt32(words[1])+1).ToString();
        switch (taskForm.ShowDialog())
```

```
        {
            case DialogResult.Abort:
                break;
            case DialogResult.Cancel:
                break;
            case DialogResult.Ignore:
                break;
            case DialogResult.No:
                {
                    if (!this.Marks[Convert.ToInt32(words[1])])
                        this.buttonArray[Convert.ToInt32(words[1])].BackColor = System.Drawing.Color.Salmon;
                }
                break;
            case DialogResult.None:
                break;
            case DialogResult.OK:
                {
                    this.Marks[Convert.ToInt32(words[1])] = true;
                    this.StudentMark.Text = "Mark: " + GetStudentMark().ToString();
                    this.buttonArray[Convert.ToInt32(words[1])].BackColor =
System.Drawing.Color.LightGreen;
                }
                break;
            case DialogResult.Retry:
                break;
            case DialogResult.Yes:
                break;
            default:
                break;
        }
    }

    private void LogOutBtn_Click(object sender, EventArgs e)
    {
        DialogResult result = MessageBox.Show("Are you sure to log out?", "Log Out",
MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation);

        if (result == DialogResult.Yes)
```

```
            {
                UserLogin loginForm = new UserLogin();
                loginForm.FormClosed += new FormClosedEventHandler(loginForm_FormClosed);
                loginForm.Show();
                this.Hide();
            }
        }

        private void loginForm_FormClosed(object sender, FormClosedEventArgs e)
        {
            this.Close();
        }

        private void TaskMenu_SizeChanged(object sender, EventArgs e)
        {

        }
        private Int32 CountCorrectTasks()
        {
            Int32 counter = 0;
            for (Int32 i = 0; i < this.Marks.Length; i++)
            {
                if (this.Marks[i])
                {
                    counter++;
                }
            }
            return counter;
        }
        private Int32 GetStudentMark()
        {
            Double correct = CountCorrectTasks();
            Double result = correct / this.TaskCount;
            result *= 100;
            return Convert.ToInt32(result);
        }
    }
}
```

ДОДАТОК Б

Чек-ліст

1. №	Action	Date(s)	Status	Found bug(s)
1	Check -> “Authorization”	01.05.2016	P	-
2	Check -> “Task list button”	01.05.2016	P	-
3	Check -> “Student manifest button”	01.05.2016	P	-
4	Check -> “Task results button”	01.05.2016	P	-
5	Check -> “Add task button”	01.05.2016	P	-
6	Check -> “Edit task button”	01.05.2016	P	-
7	Check -> “Delete task button”	01.05.2016	P	-
8	Check -> “Create task”	01.05.2016	P	-
9	Check -> “Update task”	01.05.2016	P	-
10	Check -> “Create student”	01.05.2016	P	-
11	Check -> “Create group”	01.05.2016	P	-
12	Check -> “Return button”	01.05.2016	P	-