Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА

на тему:

Комп'ютерна підсистема для контролю роботи над домашнім завданням

ПОЯСНЮВАЛЬНА ЗАПИСКА

Виконав студент 2 курсу групи 1КІ-20мсз
Спеціальності 123 Комп'ютерна інженерія
Попов В. Д.

Керівник к.т.н., доц. каф. ОТ
Снігур А. В.
" " 2022 р.

Рецензент д.т.н., проф., зав. каф. ЗІ
Лужецький В. А.
" " 2022 р.

Допущено до захисту

д.т.н., проф. Азаров О.Д.

" 17 " 06 2022 р.

ВНТУ 2022

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

Освітній рівень — бакалавр

Спеціальність — 123 Комп'ютерна інженерія

ЗАТВЕРДЖУЮ

Завідувач кафедри обчислювальної техніки

О.Д. Азаров

" 08 " 02 2022 р.

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ

студенту **Попову Віталію Дмитровичу**

1 Тема роботи «Комп'ютерна підсистема для контролю роботи над домашнім завданням» керівник роботи Снігур А.В. к.т.н., доцент, затверджено наказом вищого навчального закладу від 24 березня 2022 року № 66

2 Строк подання студентом роботи 17.06.2022 року.

3 Вихідні дані до роботи: технічний опис програмного застосунку, мова програмування C#, віконний додаток, середовище розробки Microsoft Visual Studio.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналіз технологій об'єктно-орієнтованого програмування для розробки віконних додатків, програмна реалізація, тестування розробленого програмного забезпечення.

5 Графічний матеріал — діаграма класів.

6 Консультанти розділів роботи приведені в таблиці 1.

Таблиця 1— Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Снігур А.В. к.т.н., доцент		

7 Дата видачі завдання 10.02.22 .

8 Календарний план виконання МКР приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів МКР	Строк виконання	Підпис
1	Постановка задачі	10.03.22	
2	Пошук матеріалів по технологіям розробки Android IoT застосунків	10.03.22— 25.03.22	
3	Структурне проектування установки	26.03.22— 28.03.22	
4	Обгунтування та вибір засобів реалізації системи	29.03.33— 04.04.22	
5	Розробка Android застосунку	05.04.22— 29.04.22	
6	Підготовка матеріалів тестування та розробка інструкції користувача	30.04.22— 27.05.22	
7	Оформлення пояснювальної записки та ілюстративного матеріалу	28.05.22— 06.06.22	
8	Перевірка якості виконання бакалаврської роботи та усунення недоліків	07.06.22	

Студент Попов Віталій Дмитрович

Керівник к.т.н., доц. Снігур А.В.

АНОТАЦІЯ

Пояснювальна записка містить 87 сторінок, 5 таблиці, 30 рисунків, 7 лістингів та 12 посиланнь.

Бакалаверська дипломна робота присвячена розробці підсистеми для контролю роботи над домашнім завданням. Вона поєднує у собі як модуль контролю над виконанням практичної частини завдання певної дисципліни так і теоретичної частини у вигляді контролю проходження тестування є актуальністю та своєчасністю.

В роботі проведено аналіз відомих програмних засобів для запису подій, що здійснює користувач під час навчання, та програмні засоби для тестування, врахував всі їх переваги та недоліки

В результаті роботи отримано підсистему, що реалізовує прогресивний досвід щодо контролю виконання домашніх завдань студентами.

ABSTRACT

The explanatory note contains 87 pages, 5 tables, 30 figures, 7 listings and 12 references.

The bachelor's thesis is devoted to the development of a subsystem for homework control. It combines both the module of control over the implementation of the practical part of the task of a particular discipline and the theoretical part in the form of control of testing is relevant and timely.

The paper analyzes the known software for recording events performed by the user during training, and software for testing, taking into account all their advantages and disadvantages

As a result, a subsystem was implemented that implements progressive experience in monitoring the performance of homework by students.

ЗМІСТ

					08-23.БДР.021.00.000 ПЗ		
Змн.	*Лист*	*№ докум.*	*Підпис*	*Дата*			
Розроб.		Попов В.Д.			Програмне забезпечення організації та оптимізації часу і виробничого процесу на .NET 5.0 в середовищі Visual Studio 2019 Пояснювальна записка	*Літ.*	*Арк.* 6 / *Акрушів* 54
Перевір.		Снігур А.В.					
Реценз.		.				*ВНТУ, гр. 1КІ–20мсз*	
Н. Контр.		Швець С.І.					
Затверд.		Азаров О.Д.					

ВСТУП

Цінність інформації та питома вага інформаційних послуг у житті сучасного суспільства різко зросли. Це дає підстави говорити про те, що головну роль у процесі інформатизації відіграє власне інформація, яка сама собою не виробляє матеріальних цінностей. Під інформацією (з загальних позицій) розумітимемо відомості про фактичні дані та сукупність знань про залежності між ними, тобто засіб, за допомогою якого суспільство може усвідомлювати себе і функціонувати як єдине ціле. Природно припустити, що інформація має бути науково-достовірною, доступною у сенсі можливості її отримання, розуміння та засвоєння; дані, з яких інформація отримується, повинні бути суттєвими, відповідними сучасному науковому рівню.

Інформаційне середовище включає безліч інформаційних об'єктів та зв'язків між ними, засоби та технології збору, накопичення, передачі, обробки, продукування та розповсюдження інформації, власне знання, а також організаційні та юридичні структури, що підтримують інформаційні процеси. Суспільство, створюючи інформаційне середовище, функціонує у ньому, змінює, удосконалює його. Сучасні наукові дослідження переконують у тому, що вдосконалення інформаційного середовища суспільства ініціює формування прогресивних тенденцій розвитку продуктивних сил, процеси інтелектуалізації діяльності членів суспільства у всіх його сферах, включаючи сферу освіти, зміну структури суспільних взаємин та взаємозв'язків.

Одним із пріоритетних напрямків процесу інформатизації сучасного суспільства є інформатизація освіти — впровадження засобів нових інформаційних технологій у систему освіти. Це дає можливість:

— удосконалення механізмів управління системою освіти на основі використання автоматизованих банків даних науково-педагогічної інформації, інформаційно-методичних матеріалів, а також комунікаційних мереж;

— вдосконалення методології та стратегії відбору змісту, методів та організаційних форм навчання, що відповідають завданням розвитку особистості студента в сучасних умовах інформатизації суспільства;

— створення методичних систем навчання, орієнтованих в розвитку інтелектуального потенціалу студента, формування умінь самостійно набувати знання, здійснювати інформаційно-навчальну, експериментально – дослідницьку діяльність, різноманітні види самостійної діяльності з обробки інформації;

— створення та використання комп'ютерних тестувальних, контролюючих та оцінювальних систем.

Базуючись на розглянутому, розробка, яка полягає у необхідності вирішення проблеми автоматичного тестування знань студентів, які дозволять отримати додаток для вирішення цього завдання, на основі якого буде можливість коректно організовувати та оптимізовувати процес тестування є **актуальною задачею.**

Об’єкт дослідження — процеси у програмному забезпеченні тестування знань студентів.

Предмет дослідження — методи та засоби для розробки підсистеми для контролю роботи над домашнім завданням.

Метою роботи є розробка програмного забезпечення тестування знань студентів на .NET 4.7 в середовищі Visual Studio_2019, яке дозволить вводити події та отримати увесь розпорядок користувача.

Задачі дослідження бакалаврської роботи:

— проаналізувати методи та технології розробки додатків ООП;

— проаналізувати існуючі аналоги;

— проаналізувати та обрати засоби реалізації системи;

— виконати моделювання;

— виконати розробку віконного додатку за допомогою фреймворку .NET 4.7 для подальшого використання;

— створити систему що буде аналізувати введені данні та повідомляти користувача про недоліки;

— протестувати розроблений додаток.

Методи дослідження бакалаврської роботи є використання принципів об’єктно-орієнтованого програмування мовою C#,CSS для реалізації поставленої мети.

Практичне значення отриманих результатів полягає в можливості використовування розробленої системи для контролю роботи над домашнім завданням**.**

Апробація результатів бакалаврської роботи: зроблено доповідь на L науково–технічній конференції підрозділів Вінницького національного технічного університету.

1 АНАЛІТИЧНИЙ ОГЛЯД ІСНУЮЧИХ ПІДХОДІВ ТА ПРОГРАМНИХ ЗАСОБІВ ДЛЯ ЗАПИСУ ПОДІЙ, ЩО ЗДІЙСНЮЄ КОРИСТУВАЧ ТА ПРОГРАМ ДЛЯ ТЕСТУВАННЯ

1.1 Огляд предметної області

Для активізації оволодіння новими видами діяльності, знаннями та вміннями щорічно в галузі освіти з'являються сотні комп'ютерних навчальних систем (КНС), які з успіхом застосовуються як засіб навчання та надають допомогу і у викладанні, і у вивченні того чи іншого предмета.

Це досягається за рахунок швидкого проведення розрахунків, наочного відображення та зберігання результатів, занесення та швидкості видачі інформації, активізації візуального мислення та візуальної пам'яті учня, можливості нетрадиційної постановки завдань та оперативного їх виконання. Комп'ютери можуть бути з успіхом використані на всіх стадіях навчального заняття: вони значно впливають на контрольно-оціночні функції, надають йому ігровий характер, сприяють активізації навчально-пізнавальної діяльності учнів. Комп'ютерні технології дозволяють досягти якісно більш високого рівня наочності пропонованого матеріалу, значно розширюють можливості включення різноманітних вправ у процес навчання, а безперервний зворотний зв'язок, підкріплений ретельно продуманими стимулами вчення, пожвавлює навчальний процес, сприяє підвищенню його динамізму, що в кінцевому рахунку веде до досягнення чи не головної мети власне процесуальної сторони навчання - формуванню позитивного ставлення учнів до матеріалу, що вивчається, інтересу до нього, задоволення результатами кожного локального етапу в навчанні. Комп'ютерне навчання нині, не змінюючи традиційної структури навчально-пізнавальної діяльності, наповнює її зміст новими елементами. Викладачі побачили у комп'ютері потужний засіб навчання та намагаються його застосувати у навчальному процесі. Але в розробників КНС виникає безліч проблем: від неможливості використання певних методів подання інформації до

складності програмування педагогічних задач. Так що мріючи створити універсальний засіб, часто обмежуються фрагментарними вставками програмних продуктів у навчальний процес. Найчастіше це виникає через відсутність необхідної інформації про можливості КНС. Тому необхідно визначити структуру та найважливіші суттєві вимоги, що пред'являються до КНС.

Більшість КНС побудовані за схемою: виклад матеріалу, тренування, контроль.

Програма може і не містити всі перелічені пункти, а зводитись тільки до одного з них, і тоді вона стає вузько направленою: демонстраційною, контролюючою або тренажером. Та й важко придумати щось інше, оскільки загальна структура визначається зрештою поставленими навчальними цілями.[4]

У зв'язку зі сказаним у структурі більшості КНС має містити такі блоки:

— інформаційно-теоретичний блок, що охоплює основні відомості дисципліни або розділу;

— інформаційно-довідковий блок або інформаційно-довідкову систему, що дозволяє організувати швидкий доступ до основних ухвал;

— блок контролю знань та обробки його результатів.

Інформаційно-теоретичний блок — блок пред'явлення навчальної інформації — містить набір екранів, де відображається весь обсяг навчального матеріалу КНС. За простотою підготовки матеріалу цього блоку стоїть надзвичайно складна проблема отримання виграшу від електронного помічника. Враховуючи, що читання книги або іншого друкованого тексту більш звичне та зручне, ніж робота за монітором, рекомендується уникати примітивної заміни паперової літератури на електронну.

Позбутися книжкового методу пред'явлення навчальної інформації можна шляхом використання наступних прийомів:

— інформацію видавати в короткій формі, а повний текст пред'являти лише за запитом, при цьому студенти швидко вивчать коротку форму, а повна версія зацікавить просунутих студентів;

— ширше використовувати структурування навчального матеріалу та представляти його на екрані не текстом, а у вигляді схем, графіків, мнемонічних процедур, за необхідності використовувати посилання підручники, навчальні посібники, довідники;

— урізноманітнити форму та порядок пред'явлення інформації: наприклад, подвійне підсвічування виділяти ключові слова, використовувати можливості фігурного заповнення екрана текстовим матеріалом, висвічувати не відразу весь текст, а із затримкою в часі, що відповідає раціональному темпу читання.

Перерахований набір не претендує на повноту, але психологічно дуже важливо здійснити перехід від цікавості формою до глибшого інтересу за змістом.

У сучасній освітній практиці тестування переважно використовується для вимірювання навчальних досягнень учнів, проведення якісного та кількісного аналізу результатів їхньої навчально-пізнавальної діяльності.

Комп'ютерні навчальні тести повинні відрізнятися за своєю структурою від контролюючих тестових програм, відповідно до своїх цілей та завдань у навчальному процесі. На відміну від контролюючих тестів, де незмінно присутня оцінка знань, умінь і навичок, яка нерідко призводить до стресу, страху втрати рейтингу перед товаришами по навчанню і перед собою, навчальні тести повинні спиратися принцип накопичення успіху при освоєнні знань. Інакше висловлюючись, в навчальному тестуванні має бути присутнім не «віднімальна», а накопичувальна система балів, що підвищує мотивацію до навчання з допомогою посилення емоційної компоненти. Крім того, уникнення звичайної оцінки знань знижує стрес при постійному застосуванні навчальних та розвиваючих тестів у навчальному процесі.

1.2 Аналіз сучасних програм тестування знань

Перед виробленням вимог до проектованого додатку слід вивчити існуючі аналоги. На ринку їх є достатньо багато, тому візьмемо найпопулярніші із них.

1.2.1 Пакет програм SunRav

SunRav — програма для створення тестів, проведення тестування та обробки його результатів. [5]

У пакет SunRav TestOfficePro входять програми для створення тестів, проведення тестування та обробки результатів тестування. З його допомогою можна організувати та провести тестування та іспити в освітніх закладах (вузи, коледжі, школи), а підприємства та організації можуть здійснювати атестацію та сертифікацію своїх працівників.

Інтерфейс програми «SunRav» показано на рис. 1.1.

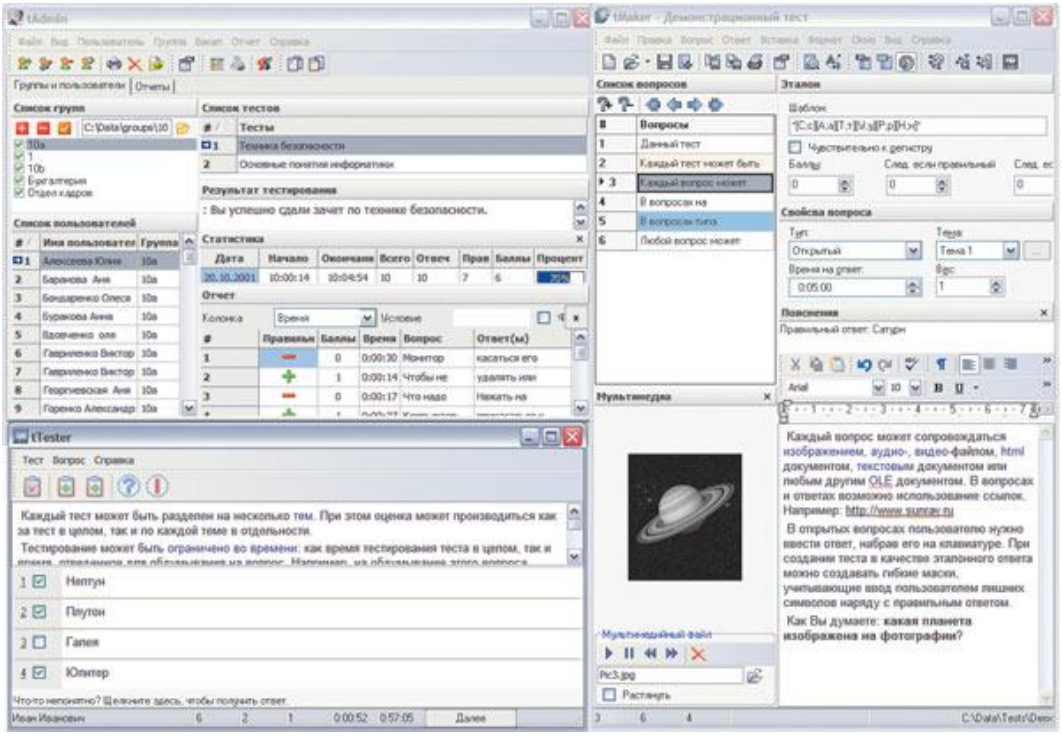

Рисунок 1.1 — Інтерфейс програми «SunRav»

На сьогоднішній день пакет програм SunRav Software є потужним інструментом комп'ютерного тестування та створення електронних книг і підручників. В таблиці 1.1 представлені недоліки та переваги даного пакету .

Таблиця 1.1 — Переваги та недоліки системи тестування SunRav

Переваги	Недоліки
Тести та результати тестування шифруються методами стійкої криптографії, що унеможливлює підробку результатів	Недостатня тех. підтримка
На тест можна встановити паролі	Перевантажений функціонал
Запитання та варіанти відповіді можна форматувати, використовуючи для цього вбудований текстовий редактор, близький за своїми функціями до MS WORD	Висока ціна (корпоративна ліцензія на ПЗ SunRav TestOfficePro / XT - 30000 грн.
Тестування можна обмежити за часом як для тесту, так і для кожного питання	Немає українського інтерфейсу

1.2.2 Система тестування INDIGO

Система тестування INDIGO — це професійний інструмент автоматизації процесу тестування та обробки результатів, який призначений на вирішення широкого спектра задач[6]:

— тестування та контроль знань учнів;

— визначення професійного рівня співробітників;

— проведення психологічного тестування;

— проведення опитувань;

— організація олімпіад та конкурсів.

Система INDIGO є потужним та гнучким засобом для проведення випробувань. Можлива робота із системою як через локальну мережу, так і через Інтернет. Є вбудований редактор тестів, система правил для проходження тестів, розроблені засоби для підвищення безпеки даних.

На рисунку 1.2 представлена форма вибору відповіді на запитання проходженні тесту на платформі INDIGO. З малюнка видно, що елементи управління оформлені класичним стилем, внаслідок чого інтерфейс відповіді на запитання у INDIGO менш зручний, ніж у psytests.org.

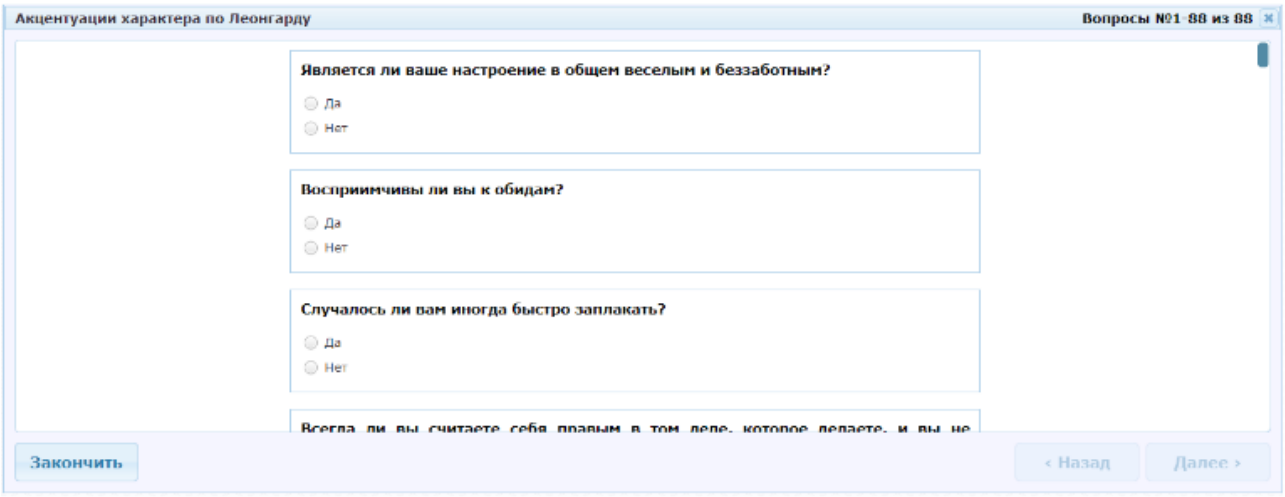

Рисунок 1.2 — Форма вибору відповіді питання при проходженні тесту на платформі INDIGO

Функціональні можливості:

— система тестування встановлюється на головний комп'ютер (сервер тестування) за допомогою інсталяційного пакета;

— система може працювати як ізольованому комп'ютері, і у локальної мережі чи через Інтернет;

— центр тестування можна розгорнути на Вашому комп'ютері або у хмарі наших Інтернет-серверах;

— усі дані зберігаються централізовано у базі даних системи;

— адміністратори працюють через програму клієнта;

— одночасно можуть працювати скільки завгодно адміністраторів із різних комп'ютерів;

— користувачі працюють через веб-браузери (Google Chrome, Яндекс.Браузер, Mozilla Firefox, Opera, Safari, Internet Explorer, Microsoft Edge та інші). Існує підтримка web-браузерів на мобільних пристроях (адаптивний інтерфейс);

— є підтримка інтеграції з Active Directory, 1С:Підприємством або з довільною інформаційною системою організації.

Таблиця 1.2 — Переваги та недоліки системи тестування INDIGO

Переваги	Недоліки
Можливість гнучкого налаштування підрахунку результатів	Висока ціна (стандартна ліцензія 12000 грн.)
Існує можливість проведення аналізу виконаних робіт, перегляд помилок	Перевантажений інтерфейс
Простота інтерфейсу, все просто і зрозуміло	Немає української мови
Можна швидко виводити результати як по одному студенту, так і по всій групі	Для користування програмою необхідна установка певного програмного забезпечення
Система гнучка в налаштуванні	
Зручність в адмініструванні та веденні статистики	

Отже, як можна побачити із опису системи тестування INDIGO призначена для закладів, які мають високі бюджети можуть та можуть дозволити собі цей продукт.

1.2.3 Конструктор тестів Keepsoft

Універсальна програма для перевірки знань. Додаток можна застосовувати для проведення тестування вдома та у навчальних закладах.

Програма дозволяє використовувати необмежену кількість тем, запитань та відповідей. [7]

Інтерфейс конструктору тестів «Keepsoft» показано на рис. 1.3.

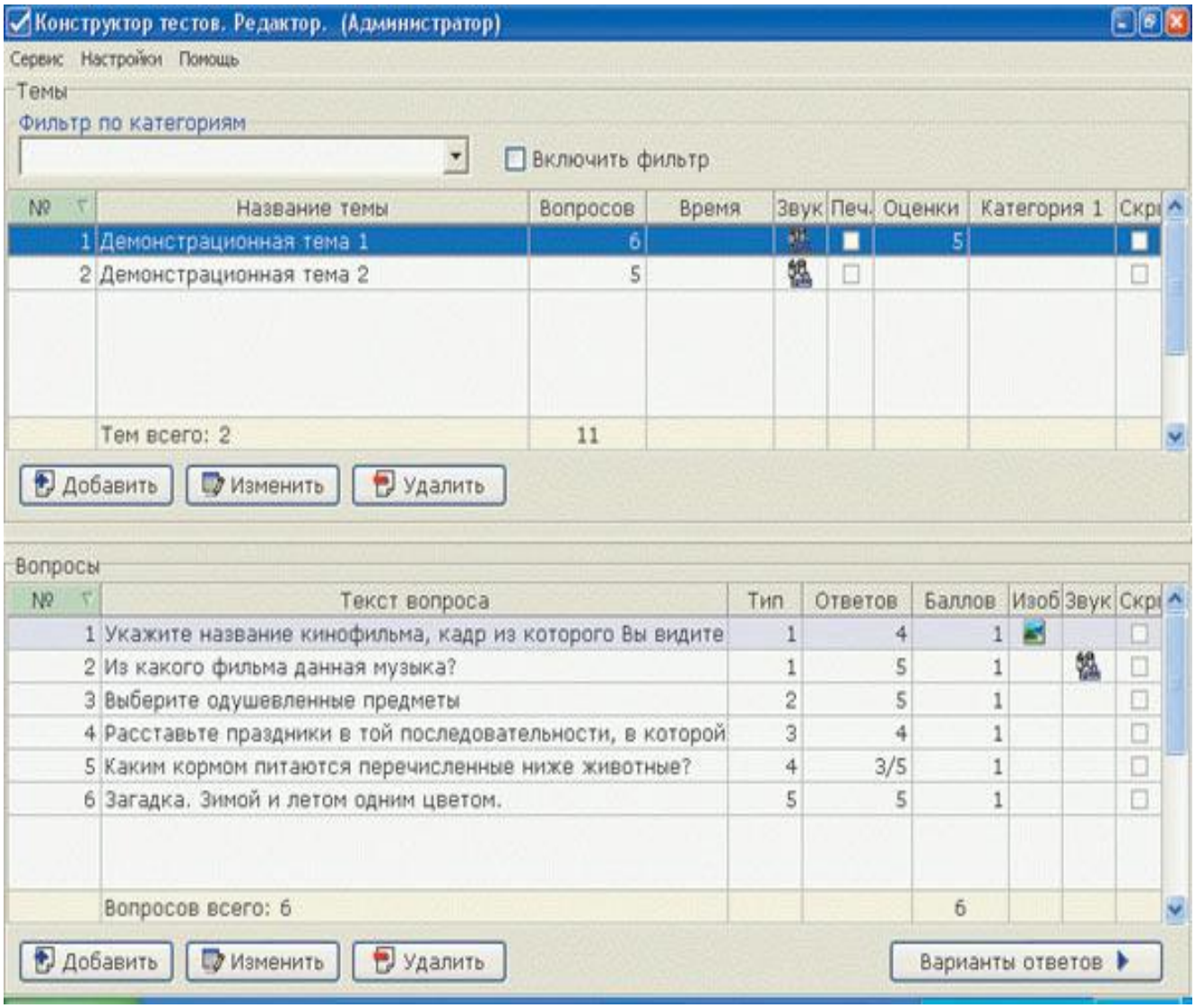

Рисунок 1.3 — Конструктору тестів «Keepsoft»

Дану програму можна завантажити з сайту розробника та ознайомитись із програмою безкоштовно протягом 30 днів. Для подальшої роботи з програмою

потрібна реєстрація з оплатою. Зареєстрованим користувачам надаються персональний реєстраційний ключ та безкоштовна технічна підтримка по e-mail.

Нижче в таблиці 1.3 представлені недоліки та переваги конструктору тестів «Keepsoft».

Таблиця 1.3 — Переваги та недоліки конструктору тестів «Keepsoft»

Переваги	Недоліки
Питання можуть містити музику (файли WAV, MID. RMI), зображення (файли JPG, BMP, ICO, EMF, WMF), відеоролики (файли AVI)	Застарілий дизайн тестів
Підтримуються питання всіх п'яти перерахованих вище типів	Немає повного перегляду тесту
Завдання ціни кожному питанню у балах	Немає української мови
Обмеження часу на відповідь	
Синхронізація бази даних; за допомогою цієї функції можна легко обмінюватися даними з іншими користувачами та переносити дані з комп'ютера на комп'ютер	

1.3 Використання середовища .NET 4.7 та мови програмування C# для створення додатків

На даний момент мова програмування C# одна з найпотужніших, що швидко розвиваються і затребуваних мов в IT галузі. На сьогодні на ній пишуться найрізноманітніші програми: від невеликих програм під віндовс до

великих веб-порталів і веб-сервісів, які обслуговують мільйон користувачів щодня.

C# вже зріла мова. І кFк і вся платформа .NET вже пройшла великий шлях. Перша версія мови вийшла разом з релізом Microsoft Visual Studio .NET в лютому 2002 року. Поточною версією мови є версія C# 9.0, яка вийшла 10.11.2019 року разом з релізом .NET 4.7.

C# має з Сі-подібний синтаксис і близька в цьому відношенні до C++ і Java. Тому, якщо ви знайомі з одним з цих мов, то опанувати C# буде легше.

C# є об'єктно-орієнтованою і в цьому плані багато перейняв у Java і C++. Наприклад, C# підтримує поліморфізм, успадкування, перевантаження операторів, статичну типізацію. Об'єктно-орієнтований підхід дозволяє вирішити завдання з побудови великих, але в той же час гнучких, масштабованих і розширюваних додатків. C# активно розвивається, і з кожною новою версією з'являється все більше нових функціональностей, як, наприклад, лямбда-вирази, динамічне зв'язування, асинхронні методи, тощо.

Коли говорять C#, нерідко мають на увазі технології платформи .NET (Windows Forms, WPF, ASP.NET, Xamarin). І, навпаки, коли говорять .NET, нерідко мають на думці саме C#. Однак, хоча ці поняття пов'язані, ототожнювати їх не варто. Мова C# бла створеною спеціально для роботи з фреймворком .NET, проте саме поняття .NET набагато ширше. Можна виділити наступні її основні риси:

— підтримка декількох мов є основою платформи є середовище виконання Common Language Runtime (CLR), завдяки чому .NET підтримує кілька мов: поряд з C # це також VB.NET, C++, F#, а також різні діалекти інших мов, прив'язані до .NET, наприклад, Delphi. NET. При компіляції код на будь-якій з цих мов компілюється в збірку спільною мовою CIL (Common Intermediate Language) — свого роду асемблер платформи .NET. Тому за певних умов ми можемо зробити різні модулі однієї програми на різних мовах;

— кросплатформеність є платформою з можливістю руху (з деякими обмеженнями). Наприклад, остання версія платформи на даний момент — .NET 4.7 підтримується на більшості сучасних ОС Windows, MacOS, Linux, використовуючи різні технології на платформі .NET, можна розробляти програми на мові C# для самих різних платформ — Windows, MacOS, Linux, Android, iOS, Tizen;

— потужна бібліотека класів представляє спільну для всіх підтримуваних мов бібліотеку класів. І який б додаток ми не збиралися писати на C# — калькулятор, чат чи складний веб-сайт — так чи інакше ми використаємо бібліотеку класів .NET;

— різноманітність технологій є середовище виконання CLR і базова бібліотека класів є основою для цілого стека технологій, які розробники можуть використати при побудові тих чи інших програм наприклад, для роботи з базами даних в цьому стеку технологій призначені технології ADO.NET та Entity Framework Core, для побудови графічних додатків з красивим та складним інтерфейсом користувача — технологія WPF і UWP, для створення більш простих графічних додатків — Windows Forms, для розробки мобільних додатків — Xamarin, для створення веб-сайтів і веб-додатків — ASP.NET і т.д;

— до цього варто додати активно розвивається і набирає популярних Blazor — фреймворк, який працює поверх .NET і який дозволяє створювати веб-додатки як на стороні сервера, так і на стороні клієнта, а в майбутньому буде можливість створювати мобільні додатків та десктоп-додатки;

— продуктивність згідно ряду тестів веб-додатки на .NET 4.7 в ряді категорій суттєво випереджають веб-додатки, побудовані з допомогою інших технологій, такі додатки на .NET 4.7 загалом відрізняються високою продуктивністю;

— також ще слід відзначити таку особливість мови C# і фреймворку .NET, як автоматичне збирання сміття, а це означає, що в більшості випадків не

доведеться, на відміну від інших мов, звільняти пам’ять вручну, вище зазначене середовище CLR саме викличе збирач сміття та очистить пам'ять.

Варто відзначити, що .NET довгий час розвивався головним чином як платформа для Windows під назвою .NET Framework.

В 2019 вийшла крайня версія даної платформи — .NET Framework 4.8. Вона більше не розвивається.

З 2014 Microsoft став розвивати альтернативну платформу — .NET Core, яка вже призначалася для різних платформ і повинна була увібрати в себе всі можливості застарілого .NET Framework і додати нову функціональність. Потім Microsoft послідовно випустив ряд версій цієї платформи: .NET Core 1, .NET Core 2, .NET Core 3. Логічним розвитком .NET Core 3.0 стала платформа .NET 4.7. Тому слід розрізняти .NET Framework, який призначений лише для Windows, і кросплатформенних .NET 4.7.

Також варто згадати про платформу Mono, яка була створена ще в 2004 році і представляла опен сорс версію платформи .NET Framework для Linux і MacOS. Використовуючи Mono, можна було створювати кросплатформенні додатки на C#. Mono як підтримується і донині. Наприклад, Xamarin — технологія для створення мобільних додатків для Android та iOS за допомогою мови C# використовує Mono.

Нерідко додаток, створений з використанням C#, називають керованим кодом (managed code). Це означає, що для цієї програми створено на основі платформи .NET і тому керується загальномовним середовищем CLR, яке завантажує додаток і при необхідності очищує пам'ять. Але є також додатки, наприклад, створені на мові C++, які компілюються не в спільну мову CIL, як C# або F#, а в звичайний машинний код. В такому випадку .NET не керує додатком та не очищує сміття.

Як вище писалося, код на C# компілюється в додаток або складається з розширеннями exe або dll на мові CIL. Далі при запуску на виконання подібного програми відбувається JIT-компіляція (компіляція в реальному часі) в машинний

код, який потім виконується процесором. При цьому, додаток може бути великим і містити велику кількість інструкцій, в поточний момент часу компілюватиметься лише та частина програми, до якої безпосередньо йде звертання. Якщо ми звернемося до іншої частини коду, то вона буде скомпільована з CIL в машинний код. При тому вже скомпільована частина програми зберігається до завершення роботи програми. У підсумку це підвищує продуктивність.

1.2.4 Система відстежування подій Plerdy

Відстежування подій і цілей у системі Plerdy дозволяє налаштувати аналітику і дізнатися, чи взаємодіяв користувач з певним програмним забезпеченням, елементом, посиланням тощо.

У цій системі легше відстежувати макро-події. Ви одразу ж помічаєте, коли користувачі відправляють форму, підписуються або залишають сторінку. Можна використовувати інструмент відстежування подій і цілей для виявлення тонкощів та мікро-подій, які також впливають на досвід користувачів. Все велике завжди починається з малого.

Рисунок 1.4 — Системf Plerdy

Plerdy записує всі важливі взаємодії користувачів і зберігає їх, щоб надати вам детальну статистику відстежування дій користувачів, як тільки вам це знадобиться. Просто створіть подію і розпочніть аналіз.

1.2.5. Аналіз програмного забезпечення і систем відстеження поведінки користувачів

Існує багато різних систем та сервісів відстеження поведінки користувачів. Smartlook — аналітична веб-консоль для маркетологів та дизайнерів. Сервіс обробляє аналітичні дані із сайту, програми. Можна створити безкоштовний обліковий запис на перші 10 днів.

Переваги:

— відображає переміщення миші та кліки на сайті, мобільному додатку;

— надає доступ до теплових карток;

— можливість відстежувати події;

— сегментація потенційних клієнтів;

— аналітика даних.

UsabilityHub — це онлайн сервіс, за допомогою якого можна оцінити інтерфейс сайту. Доступні три види тестування: Navflow (на орієнтація користувача), Fivesecondtest (оцінка дизайну), Clicktest (змодельована карта кліків). Можна підключити безкоштовний пакет із набором базових тестів тривалістю до 2-х хвилин. Також доступні пакети Basic, Pro, Enterprise з необмеженим доступом до всіх інструментів.

FullStory показує, які дії робили користувачі на веб-сайті. Сервіс використовують компанії у сфері SaaS, а також e-commerce. Крім аналітики, інструмент збирає докладну інформацію про сеанси, пропонує ефективні UX-рішення. Зареєструвавшись можна отримати доступ до безкоштовної демоверсії.

Usertesting також дозволяє переглядати записи сеансів користувачів, щоб краще розуміти їхні потреби, знаходити біль та усувати. Сервіс тестування знаходить слабкі сторони, спираючись на думку і коментарі користувачів. Є безкоштовна пробна демоверсія.

Hotjar входить до ТОП сервісів для аналізу юзабіліті сайту. Що може сервіс:

— доступ до теплових карт (карта кліків, скролла, рухи мишею);

— записи сесій;

— збір зворотного зв’язку

— створення опитувань або використання готових шаблонів;

— інтеграція коїться з іншими сервісами.

Інструмент має багато позитивних відгуків та рекомендацій фахівців. На сайті можна оформити тимчасову безкоштовну передплату на 35 щоденних сеансів або купити пакет з великою кількістю можливостей.

CrazyEgg — онлайн-сервіс, що допомагає зрозуміти, як гість взаємодіє із сайтом. Можна формувати карти кліків та скрола. Додатковий інструмент Confetti призначений для сегментації аудиторії. Безкоштовної версії немає, можна оформити пробну передплату за $10.

Five second tests. Інструмент на базі UsabilityHub. Це тестування, де користувачеві дається 5 секунд для перегляду дизайну, після чого він повинен відповісти на кілька запитань. Завдання — визначити перше враження про вашу сторінку. Команда стверджує, що 5 секунд достатньо для звичайного користувача, щоб зрозуміти, чи зміг ваш дизайн передати основний меседж. Щоб скуштувати безкоштовно, необхідно зареєструватися.

Загалом усі відомі програми можна поділити на такі підгрупи.

WEB-додатки та онлайн-тести — це програмне забезпечення функціонує як звичайне тестування. Людині пропонується пройти тест, тобто відповісти на певні питання. Програма на 106 підставі цих відповідей знаходить відповідність у своїй базі даних чи за допомогою різних алгоритмів визначає відсотковий склад відповідей та вибирає найконкурентнішу. Проблема таких програм у тому, що вони не є повноцінними інформаційними системами. Часто це взагалі окрема WEB-сторінка із набором скриптів. Такі програми не можуть навчатися й аналізувати величезні обсяги інформації.

Статистичні системи — це програмне забезпечення може оперувати великими масивами даних та створювати статистику. Наприклад, можна

показати вибірку з найуживаніших слів та словосполучень. Подальший аналіз даних здійснює людина та на його підставі робить відповідні висновки. Повноцінний комп'ютерний аналіз та навчання системи відсутні. Як приклад відповідних систем можна навести такі ресурси: · Istio.com – дає змогу отримати розширений аналіз тексту за словами, виводить топ найуживаніших слів та іншу інформацію щодо тексту. · Info.seocafe.info – ресурс дає змогу визначити основні параметри тексту. Призначений передусім для SEO-оптимізації (рис. 5).

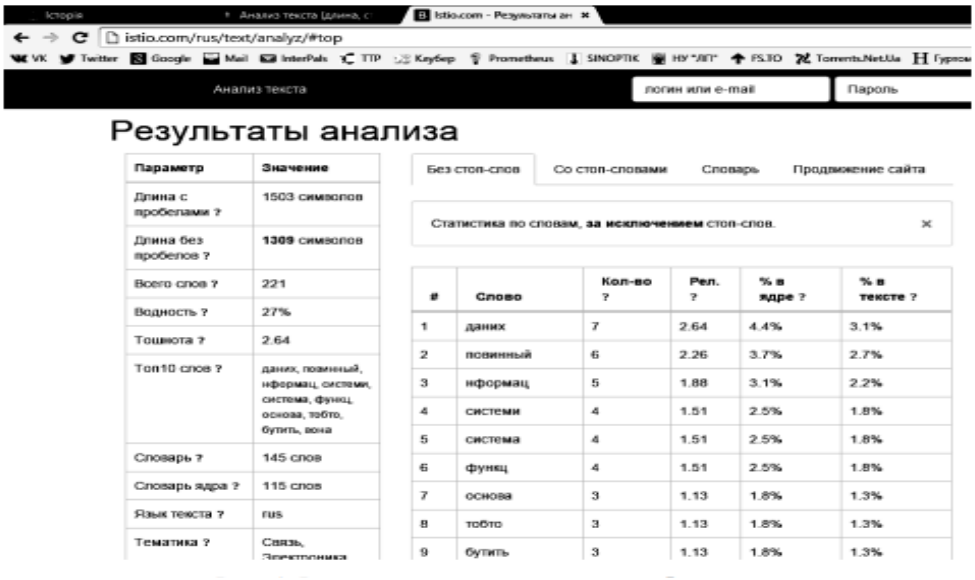

Рис. 4.— Результати аналізу тексту на Istio.com

Переваги таких ресурсів – простота використання: користувач просто вводить необхідний текст, а система автоматично його опрацьовує. Але недоліком є те, що ці системи не дають змоги з'ясувати психологічний чи емоційний стан людини, адже призначені для статистичного опрацювання тексту. На основі цієї класифікації загалом потрібно врахувати такі особливості розроблювакної системи. Система повинна опрацьовувати невеликі обсяги даних. Необхідний механізм збереження даних для їх подальшого опрацювання. Система має бути доступною широкому загалу. Фінальний програмний продукт

повинен функціонувати як окреме програмне забезпечення (не як додаток для певної соціальної мережі чи розширення для браузера).

Система повинна чітко Визначати програми, що запускаються. Враховуючи усе описане вище, можна визначити актуальність роботи. Вона буде корисна як звичайним рядовим користувачам так і викладачам.

2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ РОБОТИ ПІДСИСТЕМИ ДЛЯ ВИЗНАЧЕННЯ ТЕРМІНІВ РОБОТИ З ПЗ ДЛЯ ВІДПОВІДНОЇ ДИСЦИПЛІНИ ВКЛЮЧНО ТЕСТУВАННЯМ ЗНАНЬ

2.1 Розробка структури віконного додатку

Елементи керування графічного інтерфейсу додають інтерактивності програмам та роблять їх привабливішими для користувача.

Інтерфейсу користувача (UI) використовують UI елементи, щоб створити візуальну мову і забезпечити узгодженість продукту, що зробить його комфортним для користувачів і простим в навігації без значних зусиль з боку користувачів.

UI елементи зазвичай поділяються на одну з наступних чотирьох категорій:

— елементи керування введенням (Input Controls) – дозволяють користувачам вводити інформацію до системи;

— компоненти навігації (Navigation Components) – допомагають користувачеві переміщатися всередині продукту або веб-сайту. Загальні навігаційні компоненти містять панелі вкладок і головне меню програми;

— інформаційні компоненти (Informational Components) – дають інформацію користувачу;

— контейнери (Containers) – складаються зі зв’язаного разом контенту.

В нашому проекті для побудови споживацького інтерфейсу я використовував Windows Forms.

Windows Forms — є платформою користувача інтерфейсу, що служить для створення класичних додатків Windows. Вона покликана забезпечувати один з найефективніших способів створювання класичних додатків з допомогою візуального конструктора через Visual Studio.

На рис. 2.1 показана діаграма програми «Набір психологічних тестів» з правами адміністратора системи.

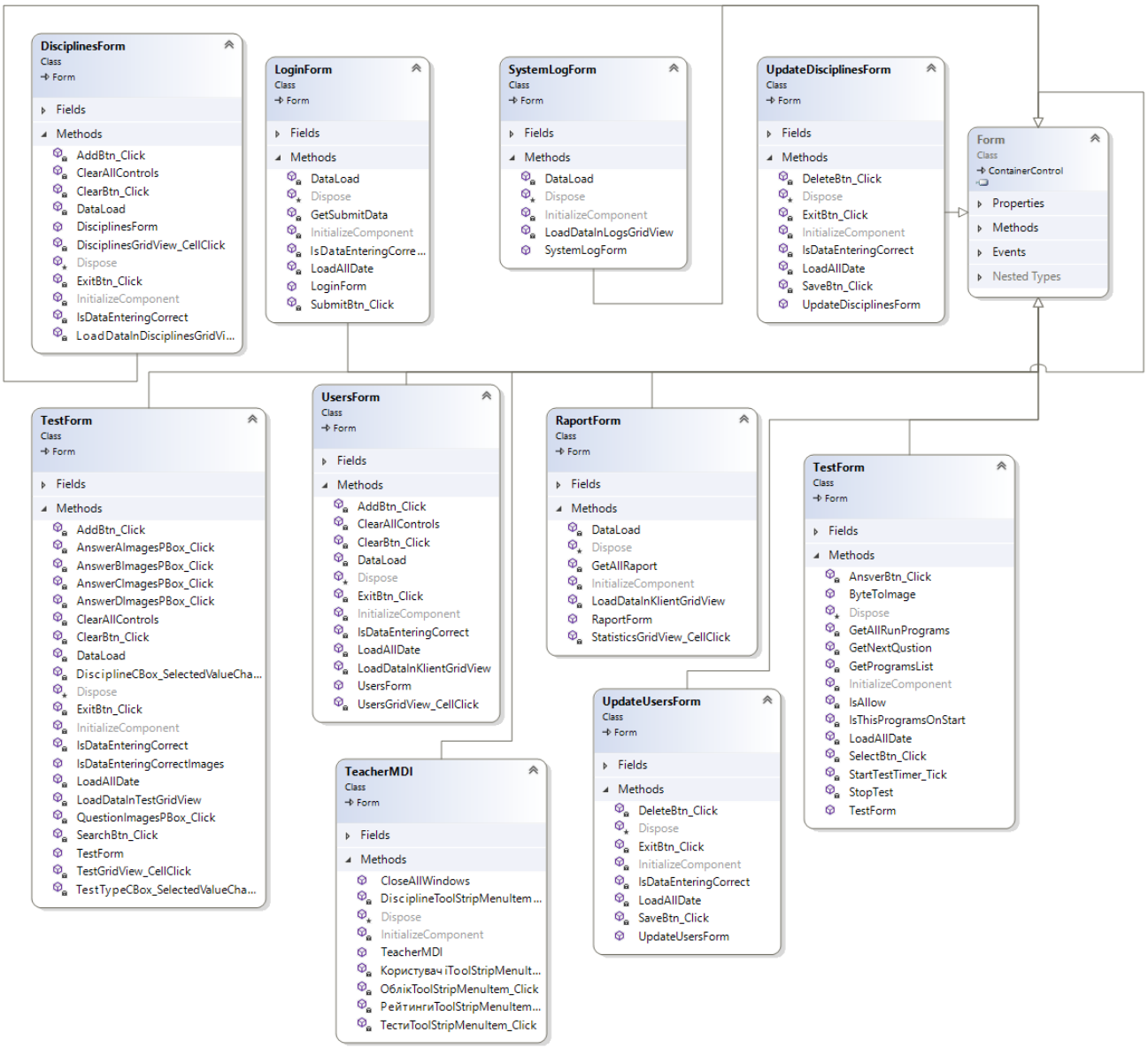

Рисунок 2.1 – Діаграма інтерфейсу програми «Набір психологічних тестів»

2.2 Особливості розробки бази даних. ER- діаграма з описанням сутностей

Схема "сутність-зв'язок" (також ER або ER-діаграма) - це вид блок-схеми, в якій показано, як різноманітні "сутності" (люди, концепції, об'єкти, і т. д.) зв'язані в системі. ER-діаграми зазвичай застосовуються для того, щоб проектувати і налагоджувати реляційні бази даних в сфері освіти, досліджувати та розробляти програмне забезпечення та інформаційні системи для бізнесу.

ER-діаграми (або ER-моделі) опираються на стандартний набір символів, в тому числі на прямокутники, ромбики, овали та лінії для того, щоб відобразити сутності, їх атрибути та зв'язки. Ці діаграми побудовані по тому самому принципу, що і граматичні структури: сутності мають роль іменників, а зв'язки — дієслова.

ER-діаграми — "побратими" схем структури даних (DSD), де замість зв’язків між сутностями відображають відношення між елементами всередині них. ER-діаграми часто використовують в поєднанні з діаграмами DFD, що схематично показує рух потоків інформації у рамках процесів або системи.

У ER-моделях та моделях даних найчастіше відзначають три рівні деталізації:

Концептуальна модель даних — схема вищого рівня при мінімальній кількості подробиць. Плюс цього підходу полягає в можливості відображати загальну структуру моделі і всю архітектуру її. Не такі масштабні системи можуть обійтись без даної моделі. І тут є зміст відразу перейти до логічної моделі.

Логічна модель даних: має в собі більш детальну інформацію, ніж концептуальна модель. На цьому рівні визначають докладніші операційні та транзакційні сутності. Логічна модель залежить від технологій, в яких буде застосовуватись.

Фізична модель даних: на базі кожної логічної моделі можна створити одну чи дві фізичні моделі. В фізичних повинно бути достатня кількість технічних подробиць для того, щоб складати і впроваджувати саму базу даних.

Слід зважати на той факт, що схожі рівні масштаба та деталізації зустрінуться і в інших видах схем (до прикладу, в діаграмах DFD), але ця класифікація буде відрізнятись від трьохсхемного підходу в розробці ПЗ, де розподіляється інформація за дещо іншим принципом. Хоча, іноді розробник застосовує ER-діаграми разом з додатковими ієрархіями, коли дизайн бази даних має мати більше інформаційних рівнів. До прикладу, розробники можуть додати

нові групи по принципу розширення вгору (суперкласи) також вниз (підкласи). А саме:

— тільки реляційні дані слід чітко визначити, що мета ER-діаграм — показувати зв'язки та відносини поміж елементами, тому вони мають відображати тільки реляційну структуру;

— тільки для структурованих даних, дані повинні бути чітко розбиті полями, стовпцями та рядками, інакше користь від ER-діаграми буде втрачена, це відноситься і до частково структурованих даних, так як лише деякі з них будуть придатні для роботи;

— складність в інтегруванні з базою даних, застосування цих ER-моделей для інтеграції зі вже існуючою базою даних — складне завдання через різницю в архітектурі.

Згідно завдання побудуємо ER діаграму.

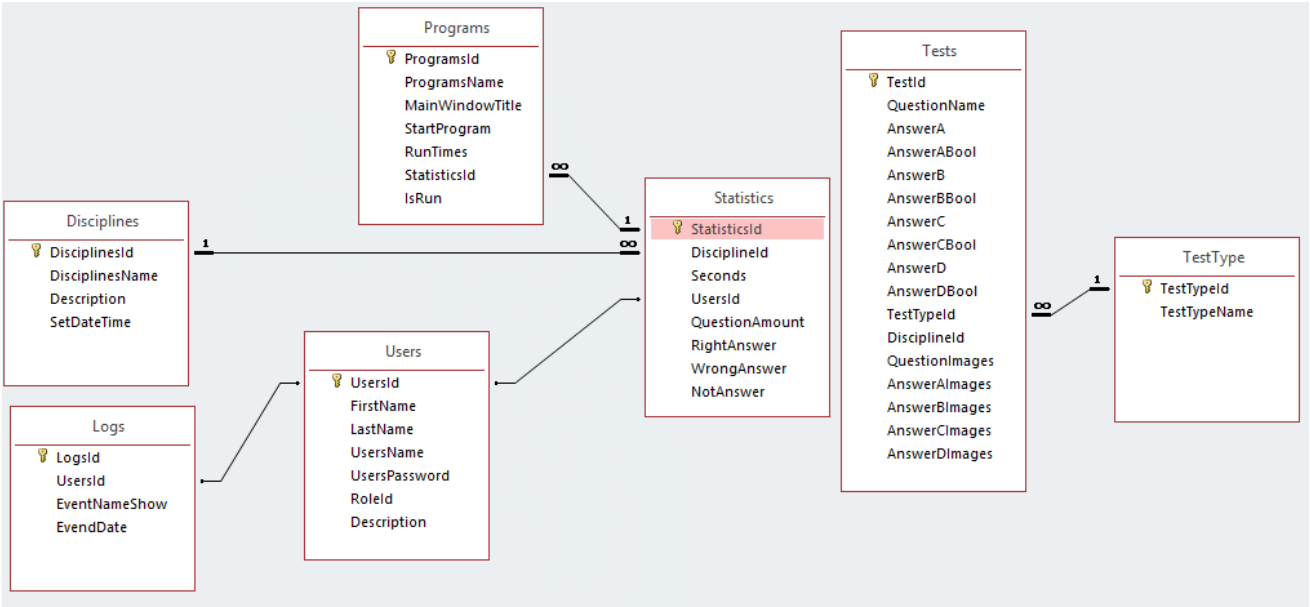

Рисунок 2.2 – ERD діаграма.

Як ми бачимо на рис. 2.2 наша база даних складається із 7 сутностей. Кожна сутність має свою таблицю, а саме:

— таблиця «TestType» — зберігає інформацію про типи тесту, а саме: картинка або текст;

— таблиця «Disciplines» — зберігає інформацію про всі дисципліни, по яких будуть проводитись тести;

— таблиця «Logs» — містить в собі інформацію щодо активності користувачів системи та їхні дії;

— таблиця «Programs» — зберігає інформацію про всі програми, що були запущені до і під час тестування знань;

— таблиця «Statistic» — зберігає інформацію про результати проходження тестування;

— таблиця «Tests» — містить інформацію про всі тести (запитання відповіді) для всіх дисциплін;

— таблиця «Users» — має в собі інформацію щодо всіх користувачів системи і їхні облікові дані.

2.3 Особливості розробки бізнес логіки додатку

Бізнес-логіка — в розробці інформаційних систем — набір правил, принципів, залежність поведінки об'єктів конкретної галузі (області людської діяльності, яку підтримує система). По-іншому можна сказати, що бізнес-логіка — це реалізація правил і обмежень операцій, які автоматизуються. Є синонімами терміну "логіка предметної області" (англ. domain logic). Бізнес-логіка визначає правила, яким підпорядковуються дані предметної області.

Простіше кажучи, бізнес-логіка — це реалізація предметної галузі в інформаційній системі. До неї відносяться: формули розрахунку щомісячних виплат з позик (у фінансовій індустрії), автоматизована відправка повідомлень електронної пошти керівнику проекту після закінчення частин завдання всіма

підлеглими (у системах управління проектами), відмова від готелю при скасуванні рейсу авіакомпанією (у туристичному бізнесі)) і т.д.

У фазі бізнес-моделювання і розробки вимог бізнес-логіка описується в вигляді:

— тексту;

— концептуальних аналітичних моделей предметної галузі (онтології);

— бізнес-правил;

— різноманітних алгоритмів;

— діаграм діяльності;

— графів та діаграм переходу станів;

— моделей бізнес-процесів.

У фазі аналізу та проектування системи бізнес-логіка втілюється в різних діаграмах мови UML або подібних до неї. У фазі програмування бізнес-логіка втілюється в коді класів та їх методів, у разі використання об'єктно-орієнтованих мов програмування, або процедур та функцій у разі застосування процедурних мов.

На жаргоні розробників програмного забезпечення «бізнес-логікою» також називаються програмні модулі, що її реалізують, та рівень системи, на якому ці модулі знаходяться (англ. business logic layer).

У багаторівневих (багатошарових) інформаційних системах цей рівень взаємодіє з нижчим рівнем інфраструктурних сервісів (англ. infrastructure layer), наприклад, інтерфейсом доступу до бази даних або файлової системи (англ. data-access layer, DAL) і рівнем сервісів додатка (англ. application services layer), який, своєю чергою, взаємодіє з рівнем користувальницького інтерфейсу (англ. user interface layer) чи зовнішніми системами.

BLL буде обробляти речі, які є частиною бізнес-сфери, а не частиною бази даних і не частиною користувальницького інтерфейсу.

В даному проекті реалізовано клас для формування звітів під назвою: «StatisticBLL».

Логіка додатка, іноді кажуть бізнес-логіка — в розробці інформаційних систем – набір правил, принципів, залежність поведінки об'єктів конкретної галузі (області людської діяльності, яку підтримує система). По-іншому можна сказати, що бізнес-логіка — це реалізація правил і обмежень операцій, які автоматизуються. Є синонімами терміну "логіка предметної області" (англ. domain logic). Бізнес-логіка визначає правила, яким підпорядковуються дані предметної області.

Простіше кажучи, бізнес-логіка — це реалізація предметної галузі в інформаційній системі. До неї відносяться: формули розрахунку щомісячних виплат з позик (у фінансовій індустрії), автоматизована відправка повідомлень електронної пошти керівнику проекту після закінчення частин завдання всіма підлеглими (у системах управління проектами), відмова від готелю при скасуванні рейсу авіакомпанією (у туристичному бізнесі)) і т.д.

У фазі бізнес-моделювання і розробки вимог бізнес-логіка описується в вигляді:

— тексту;

— концептуальних аналітичних моделей предметної галузі (онтології);

— бізнес-правил;

— різноманітних алгоритмів;

— діаграм діяльності;

— графів та діаграм переходу станів;

— моделей бізнес-процесів.

У фазі аналізу та проектування системи бізнес-логіка втілюється в різних діаграмах мови UML або подібних до неї. У фазі програмування бізнес-логіка втілюється в коді класів та їх методів, у разі використання об'єктно-орієнтованих мов програмування, або процедур та функцій у разі застосування процедурних мов.

На жаргоні розробників програмного забезпечення «бізнес-логікою» також називаються програмні модулі, що її реалізують, та рівень системи, на якому ці модулі знаходяться (англ. business logic layer).

У багаторівневих (багатошарових) інформаційних системах цей рівень взаємодіє з нижчим рівнем інфраструктурних сервісів (англ. infrastructure layer), наприклад, інтерфейсом доступу до бази даних або файлової системи (англ. data-access layer, DAL) і рівнем сервісів додатка (англ. application services layer), який, своєю чергою, взаємодіє з рівнем користувальницького інтерфейсу (англ. user interface layer) чи зовнішніми системами.

BLL буде обробляти речі, які є частиною бізнес-сфери, а не частиною бази даних і не частиною користувальницького інтерфейсу.

Модуль програми, що відстежує дії кросувача реалізований у Visual Studio. Він дозволяє з панелі керування запускати додаток для прикладу Visual Studio і визначати час його запуску і роботи. Таким чином для дисципліни програмування контролюється час роботи з відповідним додатком.

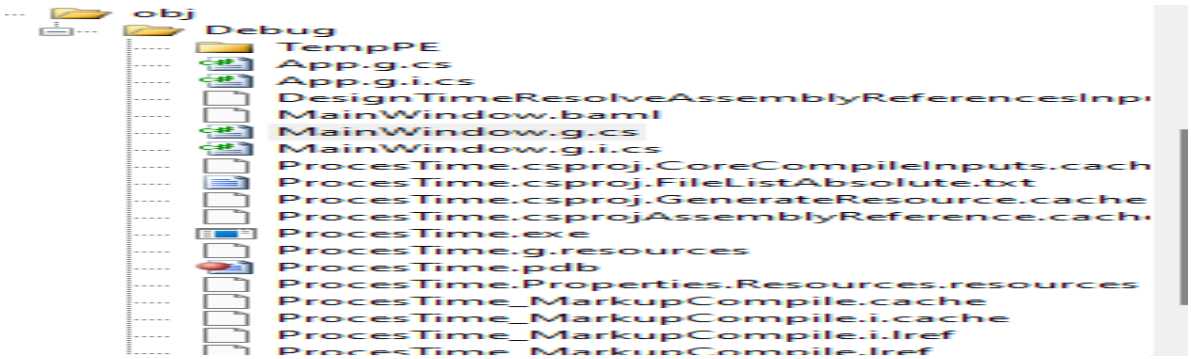

Рисунок 2.2 — Програма контролю за запуском додатком

У модулі, що відповідає за тестування реалізовано клас для формування звітів під назвою: «StatisticBLL».

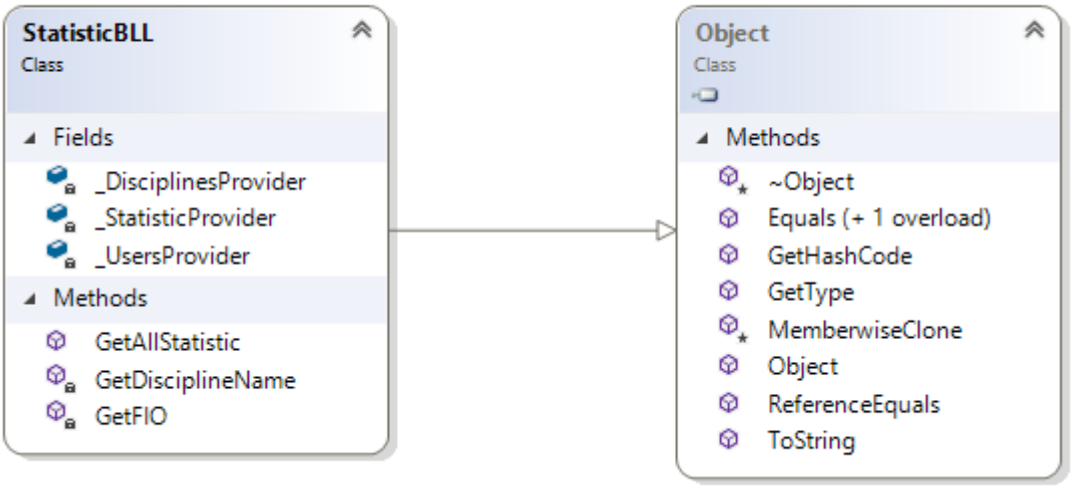

Рисунок 2.3 — Діаграма класу BLL

2.4 Особливості розробки класів для доступу до даних бази даних

Шар доступу до даних (Data Access Layer – DAL) в програмному забезпеченні є шаром комп'ютерної програми, що дає спрощений доступ до даних, які зберігаються в постійному сховищі певного типу, такого як реляційна база даних. Цей акронім зазвичай використовується в середовищі Microsoft.NET.

DAL може повернути посилання на об'єкт (в термінах об'єктно-орієнтованого програмування) разом з його атрибутами, як заміщення рядків полів з таблиці бази даних. Це дозволить створювати клієнтські (чи модулі користувача) модулі з вищим рівнем абстракції. Дана модель може реалізовуватись створенням класу разом з методами доступу до даних, що безпосередньо посилають на певний набір процедур бази даних. Інша реалізація потенційно отримує або записує записи або з файлової системи. DAL ховає складність сховища даних, яка лежить в основі даних.

Замість використувати такі команди як «створити», «видалити» або «оновити» в певній таблиці у базі, клас і кілька процедур, які зберігаються,

можуть створюватись в базі. Ці процедури можуть бути викликані методом всередині класу, що поверне об'єкт, який містить запитані значення. Або команди створити, видалити та оновити можуть виконуватись усередині простих функцій, що містяться в шарі доступу до даних.

Також методи бізнес-логіки з програми можуть співвідноситись до шару доступу до даних.

Для роботи в базі даних реалізовано 7 класів. Назва кожного класу починається з відповідної їй назви таблиці в базі даних та вкінці фіксується приставкою «Provider».

На рис. 2.4 приведено діаграму класів з методами для роботи з базою даних.

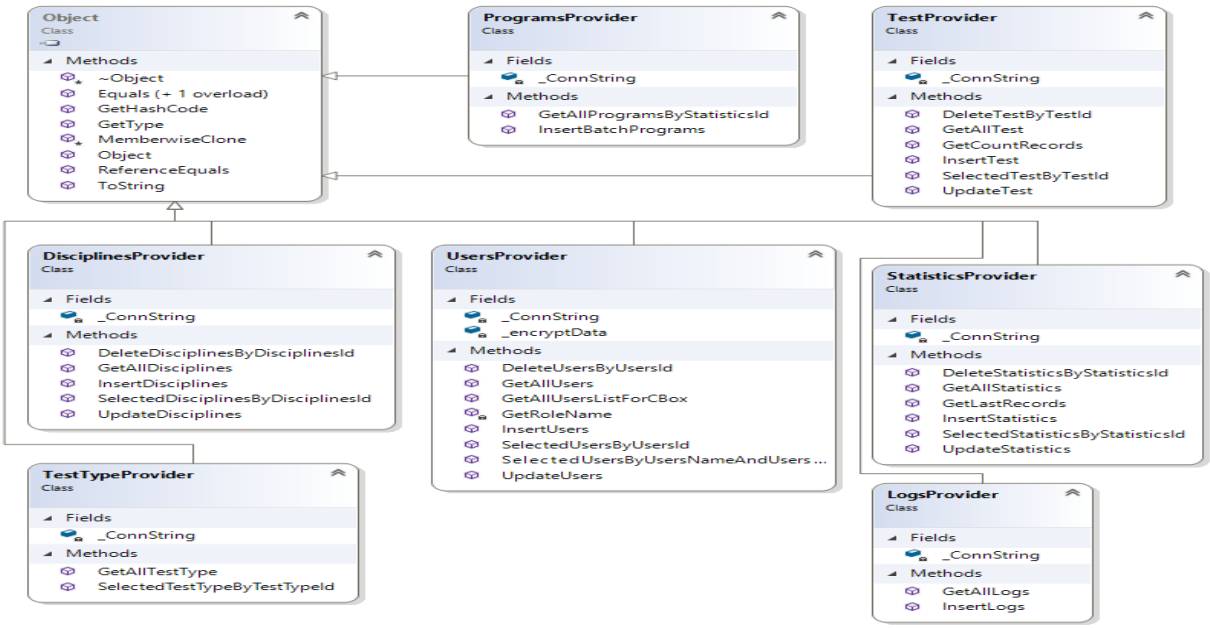

Рисунок 2.4 – Діаграма класів з методами для роботи з базою даних

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПЛАТФОРМИ

3.1 Варіантний аналіз і обгрунтування вибору програмних засобів

До того як почати розробку програмних засобів потрібно оглянути доступні програмні засоби та відібрати такі, які найкраще підійдуть для розробки даної програми.

Для розробки проведемо огляд таких мов програмування, як C++, Python, Java та C#.

C++ — це компільована мова програмування висока за рівнем, розроблена Б'ярне Страуструпом. Перша версія мови випущена у 1979 році. Найпридатніша версія мови C++ 17 вийшла в 2017 році. C++ отримала у спадок синтаксис та основні аспекти мови C. Можна сказати, C++ являється надмножиною мови C, на що теж вказує початкова назва — «C with classes» — «Сі з класами». Мова C++ підтримує парадигму об'єктно-орієнтованого програмування, характеризується високою швидкодією тому, що код програми компілюється в машинний код, а також має немалу кількість сторонніх бібліотек, які спрощують процес розробки. До мінусів мови відносять відсутність можливості автоматично керувати пам'яттю («збір сміття» — «garbage collection»), і це є потенційно небезпечним, адже допускається існування помилок, які призводять до втрати пам'яті (явище, коли частка пам'яті, виділеної для виконання програми, не звільняється після закінчення її використання). Натомість, ручне керування пам'яттю дає розробнику можливість більше контролювати роботу програми.

C# — це мова програмування високого рівня, розробники — Андерс Гейлсберг, Скот Вілтамут та Пітер Гольде в корпорації Microsoft. Перша версія мови була представлена в 2002 році. Остання робоча версія мови C# 7.0 вийшла в березні 2017 року. Мова програмування C# була побудована під впливом C++ та Java, має C-подібний синтаксис. Підтримує парадигму об'єктно-обієнтованого програмування. Має трохи нижчу швидкість, ніж мова C++, але вищу, ніж Java. Відрізняється наявністю чималої кількості сторонніх бібліотек, також є

автоматичне регулювання пам’ятті та вбудовану реалізацією деяких шаблонів проектування.

Java — це мова програмування високого рівня, представлена компанією «Sun Microsystems» в 1995 році. Остання версія мови Java Standard Edition 10 випущена в березні 2018 року. Містить С-подібний синтаксис. Підтримує парадигму об’єктно-орієнтованого програмування. Має трохи гіршу швидкодію в порівнянні з мовою С++ та мовою С#. Характеризується наявністю великої кількості сторонніх бібліотек. Java користується автоматичним збирачем сміття для керування пам'яттю в період життєвого циклу об'єкта. Програмісти вирішують, коли створити об'єкти, а віртуальна машина відповідає за звільнення пам'яті опісля того, коли об'єкт стає непотрібним. Коли до конкретного об'єкта вже нема посилань, збирач сміття автоматично прибирає його з пам'яті.

Python — це інтерпретована об'єктно-орієнтована мова програмування високого рівня, що містить строгу динамічну типізацію. Випущена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом з динамічною семантикою та динамічним зв'язуванням подають її привабливою для швидкого розробляння програм, а також її вважають засобом поєднання наявних компонентів. Python підтримує модулі та пакети модулів, і це сприяє модульності та повторному використовуванню коду. В мові програмування Python підтримано кілька парадигм програмування, а саме: об'єктно-орієнтована, процедурна, функціональна і аспектно-орієнтована. Python містить ефективні структури даних високого рівня та простий, але дуже ефективний метод об'єктно-орієнтованого програмування. Той факт, що це інтерпретована мова з динамічною обробкою типів, представляє її ідеальною для того, щоб писати скрипти та швидко розробляти прикладні програми, але самі прикладні програми можуть повільніше працювати, ніж якби вони писались на компільованій мові програмування.

В результаті проведення огляду для розробки обрали мову С# на якій будуть створені всі складові. Це обгрунтовується потужністю цієї мови

програмування, та зручністю роботи з об’єктно орієнтованними програмами.

3.2 Вибір середовища розробки

Для зручнішої розробки програмних засобів використовуються IDE (інтегровані середовища розробки). Інтегроване середовище розробки — комплекс програмних засобів для розроблення, який складається з текстового редактора коду, інструментів для компіляції та відлагодження проектів.

Для того, щоб порівняти середовища розробки для майбутнього вибору буде розглянуто середовище JetBrains CLion, Microsoft Visual Studio та Visual Studio Code.

Visual Studio Code — це текстовий редактор призначений для того, щоб розробляти застосунки для хмарних систем та веб-додатків. Visual Studio Code розповсюджується безкоштовно і являється кросплатформеним, отже підтримує усі сучасні операційні системи.

VS Code підтримує різнопланові плагіни, що з легістю інтегруються в нього, але він більше підійде для написання коду інтерпретованими мовами програмування, тому з VS Code буде не дуже зручно працювати.

Microsoft Visual Studio — це інтегроване середовище розробки, що розроблене компанією Microsoft. Підтримує мови програмування: C, C++, C#, F#, VisualBasic та інші. Дане середовище підійде тільки для операційної системи «Windows».

CLion — це IDE для розробки на мовах програмування C та C++, що розроблене компанією JetBrains. Також являється кросплатформеним та підтримує усі операційні системи.

Було обрано середовище Microsoft Visual Studio через доступність, підтримку всього необхідного для виконання, у тому числі .NET 4.7 Framework.

3.3 Розробка програмних модулів системи

Для того, щоб ефективно розробити додаток процес було розділено на два етапи.

Перший етап — створення програмної частини, що стосується ролі «викладач», другий етап — ролі «студент».

Після проходження аутентифікації користувачем, система знаючи, яка в нього роль, відповідно відкриє функціонал програми відповідно до неї.

Для ідентифікації користувача системи було створено форму реєстрації (рис. 3.1).

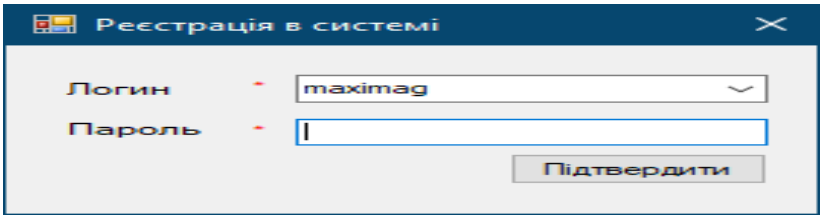

Рисунок 3.1 — Реєстрація користувача в системі

Для перевірки ролі користувача, була реалізовано два методи програми: GetSubmitData (рис 3.2) та DataLoad (рис 3.3).

```
1 reference
private void GetSubmitData() {
  try {
    if (IsDataEnteringCorrect()) {
      List<Users> listUsers = new List<Users>();
      listUsers = _UserProvider.SelectedUsersByUsersNameAndUsersPassword(UserNameCBox.Text, UserPasswordTbx.Text);
      if (listUsers.Count > 0) {
        CurrentUser = listUsers[0];
        DataLoad();
      } else {
        MessageBox.Show(NamesMy.MessageBoxExaption.ThisUserLoginAndUserPasswordNotExistInSystem, NamesMy.MessageBoxExaption.CaptionMessage);
      }
    }
  } catch {
    MessageBox.Show("Немає з'єднання!");
  }
}
```

Рисунок 3.2 — Код методу «GetSubmitData»

Метод «GetSubmitData» викликається після натискання клавіші «Підтвердити» користувачем. Після чого, метод Selected Users By Users Name And Users Password повертає список всіх користувачів із вибраним ім’ям користувача та введеним паролем. Якщо такого користувача система не знайде, тоді буде виведено повідомлення "Користувача з таким ім'ям або раролем не існує в системі". Інакше, в залежності від того, яка роль буде у вибраного

користувача, система виведе відповідне вікно. Код цього методу приведений на рис. 3.3.

```
1 reference
private void DataLoad() {
    _LogsProvider.InsertLogs(CurrentUser.UsersId, "Користувач ввійшов в систему", DateTime.Now);
    this.Visible = false;
    if (CurrentUser.RoleId == 1) {
        (new TeacherMDI()).ShowDialog();
    } else {
        (new TestsForm()).ShowDialog();
    }
    _LogsProvider.InsertLogs(CurrentUser.UsersId, "Користувач вийшов із системи", DateTime.Now);
    this.Close();
}
```

Рисунок 3.3 — Код методу «DataLoad»

Вікно, що відкриється після аутентифікації у системі користувача із роллю «викладач» (рис 3.4).

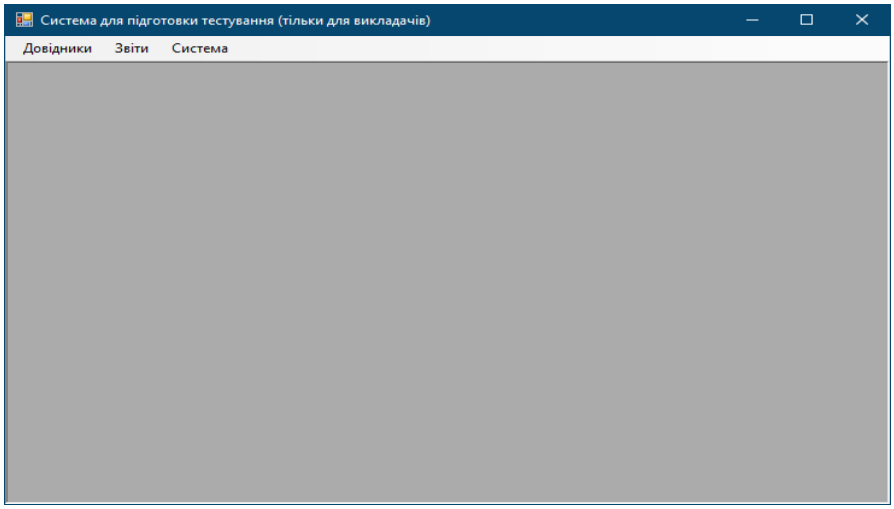

Рисунок 3.4 — Головне вікно програми для користувача із роллю «викладач»

Для користувача із роллю «студент» буде виведено вікно, де запропонується вибрати із списку дисципліну по якій буде проводитись тестування знань (рис. 3.5).

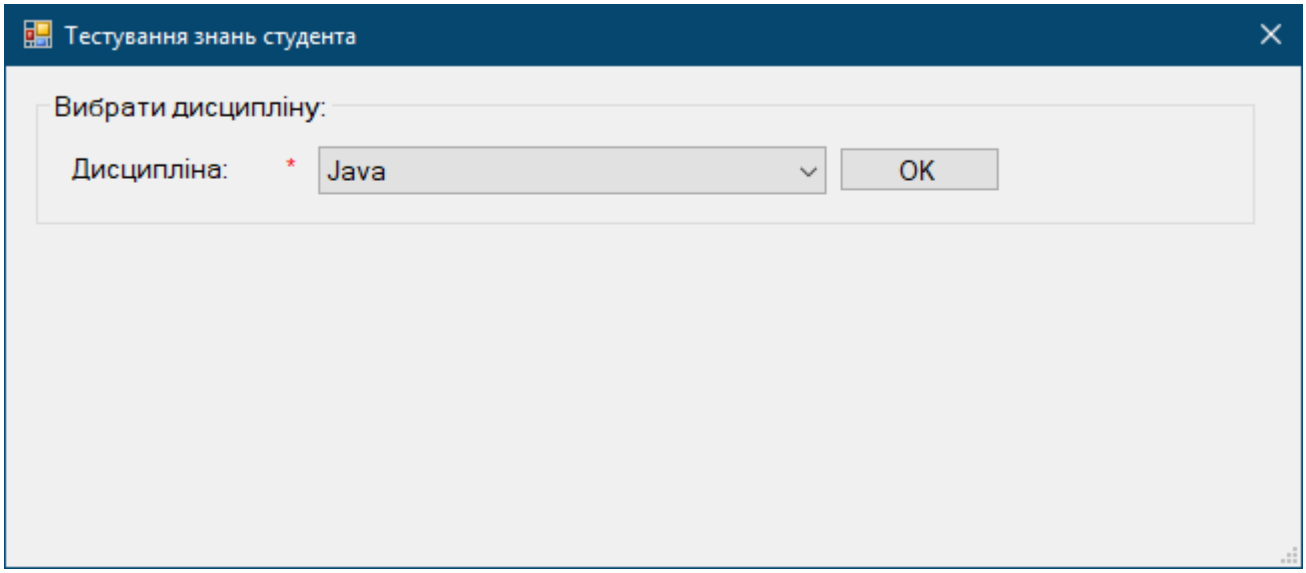

Рисунок 3.5 — Вікно програми для користувача із роллю «студент»

Після, того, як дисципліна вибрана, користувачу необхідно натиснути клавішу «ОК» для того, щоб програма розпочала тестування знань (рис 3.6).

Як можна побачити, зверху вікна система виводить запитання та варіанти відповіді на нього, після вибору варіанту відповіді, користувачу необхідно натиснути кнопку «Відповісти», після чого система виведе наступне питання. Код методу із перевіркою вибраної відповіді приведений на рисунку 3.7.

Як можна побачити із коду, насамперед в ньому перевіряється кількість вже заданих питань відносно всіх питань системи. Якщо кількість заданих питань менша за кількість всіх запитань, тоді викликається метод «GetNextQustion», інакше метод «StopTest» (рис. 3.8—3.9).

Також, програма відслідковує запущені програми до тестування знань та під час тестування знань. Всі ці дані програма записує у базу даних.

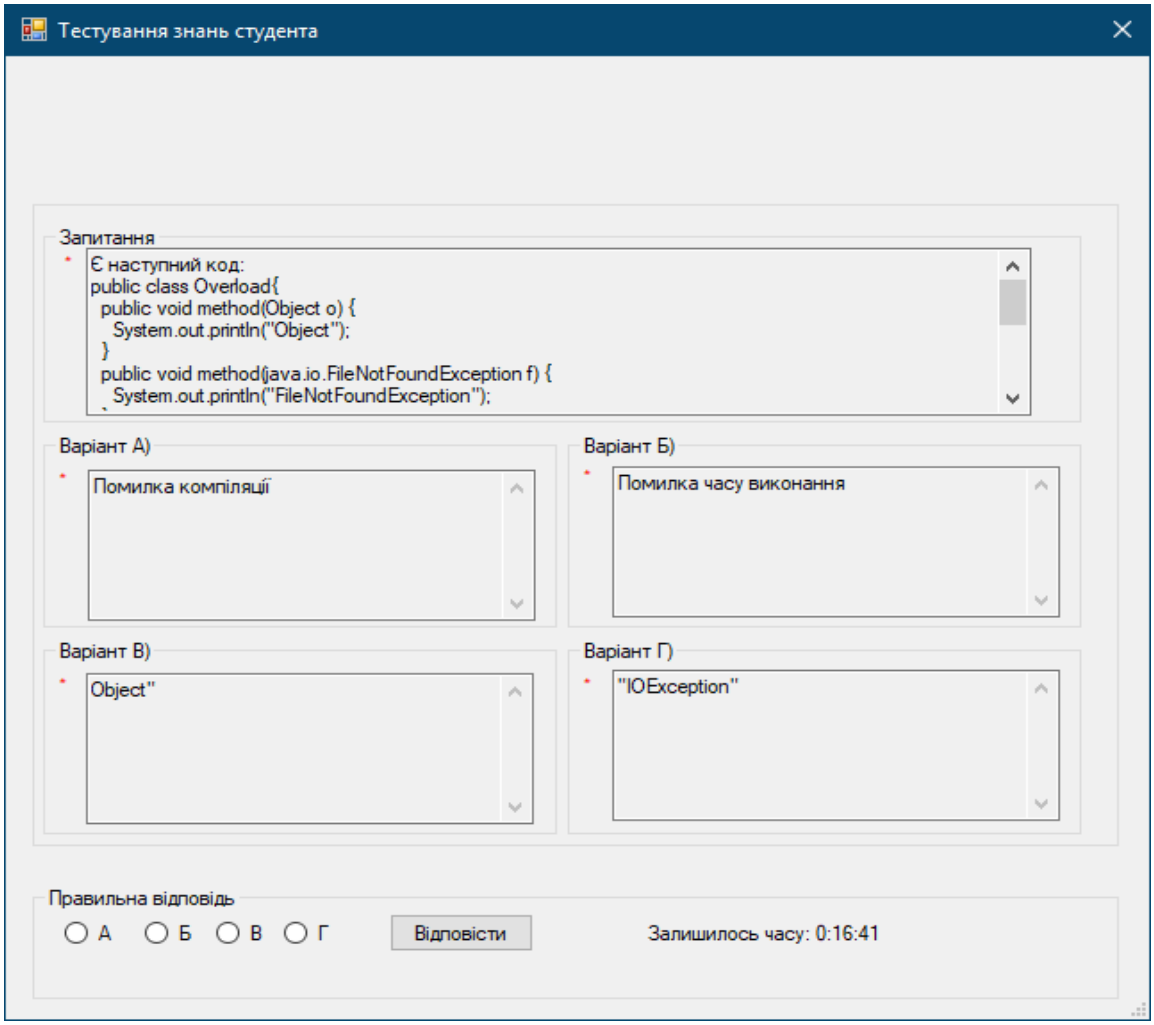

Рисунок 3.6 — Тестування знань

```
1 reference
private void AnsverBtn_Click(object sender, EventArgs e) {
    if (_AmountAnsver < _TestList.Count) {
        if (AnswerABool.Checked && AnswerABool.Checked == _SelectTest.AnswerABool) {
            _RightAnsver++;
        } else if (AnswerBBool.Checked  && AnswerBBool.Checked == _SelectTest.AnswerBBool) {
            _RightAnsver++;
        } else if (AnswerCBool.Checked  && AnswerCBool.Checked == _SelectTest.AnswerCBool) {
            _RightAnsver++;
        } else if (AnswerDBool.Checked  && AnswerDBool.Checked == _SelectTest.AnswerDBool) {
            _RightAnsver++;
        }
        GetNextQustion();
    } else {
        StopTest();
    }
}
```

Рисунок 3.7 — Код події «AnsverBtn_Click»

```
2 references
private void GetNextQustion() {
    _SelectTest = _TestProvider.SelectedTestByTestId(_TestList[_AmountAnsver].TestId);
    if (_SelectTest.TestTypeId == 1) {
        QuestionNameTBox.Text = _SelectTest.QuestionName;
        AnswerATBox.Text = _SelectTest.AnswerA;
        AnswerBTBox.Text = _SelectTest.AnswerB;
        AnswerCTBox.Text = _SelectTest.AnswerC;
        AnswerDTBox.Text = _SelectTest.AnswerD;
        AddTextGBox.Visible = true;
        GraphicsGBox.Visible = false;
    } else {
        QuestionImagesPBox.Image = ByteToImage(_SelectTest.QuestionImages);
        AnswerAImagesPBox.Image = ByteToImage(_SelectTest.AnswerAImages);
        AnswerBImagesPBox.Image = ByteToImage(_SelectTest.AnswerBImages);
        AnswerCImagesPBox.Image = ByteToImage(_SelectTest.AnswerCImages);
        AnswerDImagesPBox.Image = ByteToImage(_SelectTest.AnswerDImages);
        AddTextGBox.Visible = false;
        GraphicsGBox.Visible = true;
    }
    _AmountAnsver++;
}
```

Рисунок 3.8 — Код методу «GetNextQustion»

```
2 references
private void StopTest() {
    StartTestTimer.Stop();
    TimeSpan timeQuest = DateTime.Now - _StartTest;
    int wrongAnswer = _TestList.Count - _RightAnsver;
    int notAnsver = _TestList.Count - _AmountAnsver;
    int lastStatisticId = 0;

    _StatisticsProvider.InsertStatistics(Convert.ToInt32(DisciplineCBox.SelectedValue), timeQuest.TotalSeconds, LoginForm.CurrentUser.UsersId,
      _TestList.Count, _RightAnsver, wrongAnswer, notAnsver);
    lastStatisticId = _StatisticsProvider.GetLastRecords();
    _LogsProvider.InsertLogs(LoginForm.CurrentUser.UsersId, "Тести по дисципліні " + _selectedDisciplines.DisciplinesName, DateTime.Now);
    for (int i = 0; i < _RunProgramsList.Count; i++) {
        _RunProgramsList[i].StatisticsId = lastStatisticId;
        _RunProgramsList[i].IsRun = true;
    }
    _ProgramsProvider.InsertBatchPrograms(_RunProgramsList);

    for (int i = 0; i < _StartProgramsList.Count; i++) {
        _StartProgramsList[i].StatisticsId = lastStatisticId;
    }
    _ProgramsProvider.InsertBatchPrograms(_StartProgramsList);

    string message = "Кількість правельних відповідей: " + _RightAnsver.ToString() + " з " + _TestList.Count.ToString() + " \r\n";
    MessageBox.Show(message);
    AddTextGBox.Visible = false;
    GraphicsGBox.Visible = false;
    AnsverGBox.Visible = false;
    DisciplineGBox.Visible = true;
    _RightAnsver = 0;
}
```

Рисунок 3.9 — Код методу «StopTest»

Якщо користувач запускає сторонню програму, його буде про це негайно проінформовано відповідним повідомленням (рис. 3.10).

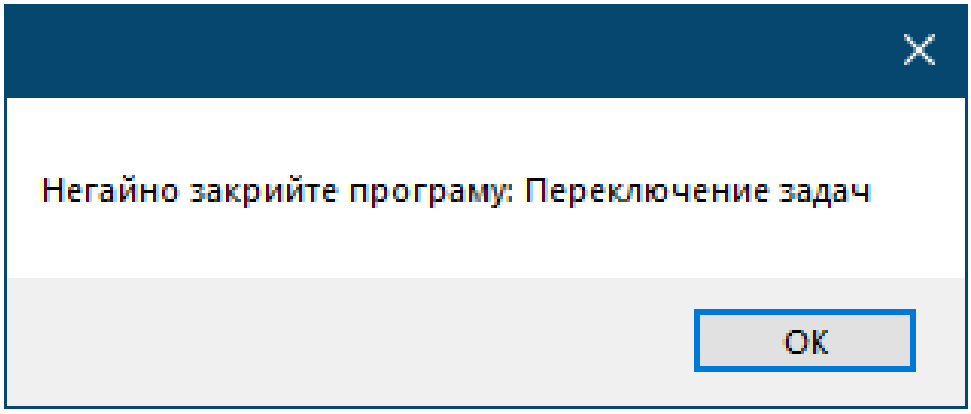

Рисунок 3.10 — Вікно, про запущену програму

Час, скільки була відкрита програма, теж буде записано.

В кінці проходження тестування програма виведе результат у діалоговому вікні (рис. 3.11).

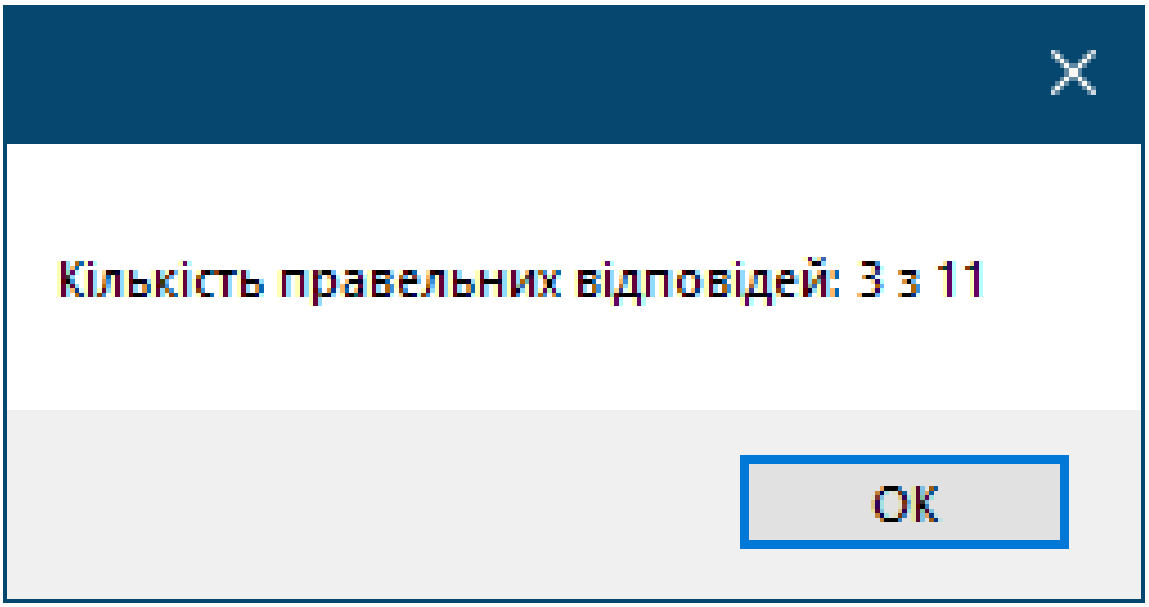

Рисунок 3.11 — Діалогове вікно із результатом тестування знань

В свою чергу користувач із роллю «викладач» має всій набір функціоналу системи, а саме:

— можливість додавання користувачів системи;

— можливість створення дисциплін;

— можливість перегляду логів системи;

— можливість створення тестування для кожної дисципліни;

— можливість формування звітності по пройдених тестах.

На рисунку 3.12 показано можливість створення нової дисципліни, по якій буде проводитись тестування.

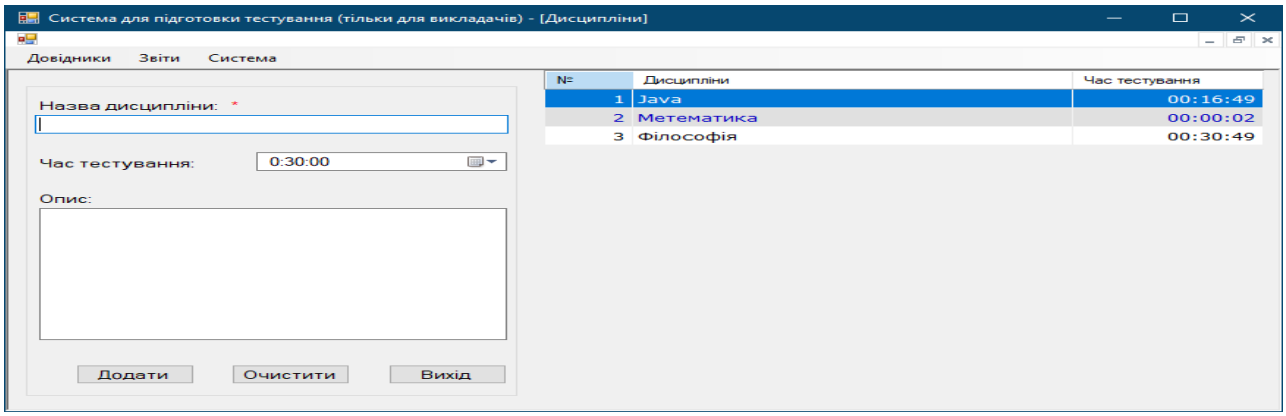

Рисунок 3.12 — Вікно для створення та редагування даних дисципліни

Для створення нової дисципліни було реалізовано подію із викликом методів, що приведений на рисунку 3.13.

```
1 reference
private void AddBtn_Click(object sender, EventArgs e) {
    if (IsDataEnteringCorrect()) {
        _DisciplinesProvider.InsertDisciplines(DisciplinesNameTBox.Text, DescriptionTBox.Text, SetDateTimeDTP.Value);
        DataLoad();
        ClearAllControls();
    }
}
```

Рисунок 3.13 — Код події «AddBtn_Click»

Як можна побачити, перед занесенням даних в базу даних, за допомогою методу «IsDataEnteringCorrect» проводиться валідація введених даних користувачем програми. Код методу «IsDataEnteringCorrect» приведений на рисунку 3.14.

```
1 reference
private bool IsDataEnteringCorrect() {
    bool isCorrect = true;
    if (_validation.IsDataEntering(DisciplinesNameTBox.Text)) {
        DisciplinesNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
    } else {
        DisciplinesNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
    }
    return isCorrect;
}
```

Рисунок 3.14 — Код методу «IsDataEnteringCorrect»

Також, якщо дані було введено із помилками, в програмі реалізована можливість їх редагування або видалення. Для цього необхідно в правій частині екрану вибрати відповідну дисципліну, після чого програма відкриє відповідне вікно (рис. 3.15).

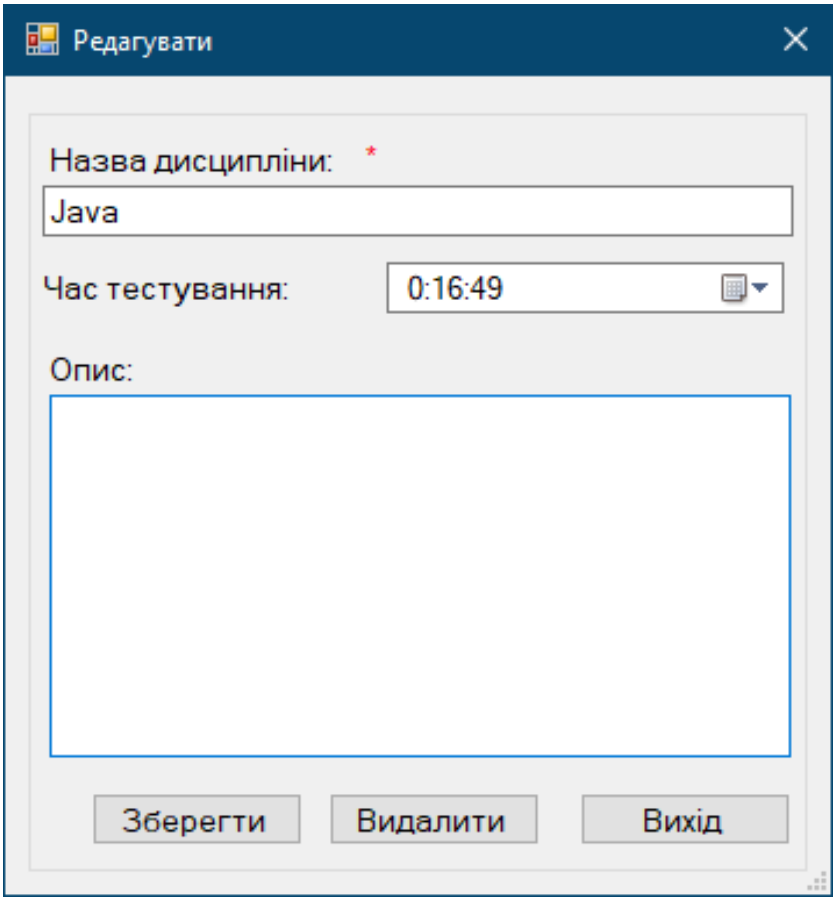

Рисунок 3.15 — Вікно для редагування даних дисципліни

Для додавання тестів по вибраній дисципліні необхідно відкрити відповідне вікно, для цього необхідно перейти по меню програми «Довідники» — > «Тести» (рис. 3.16).

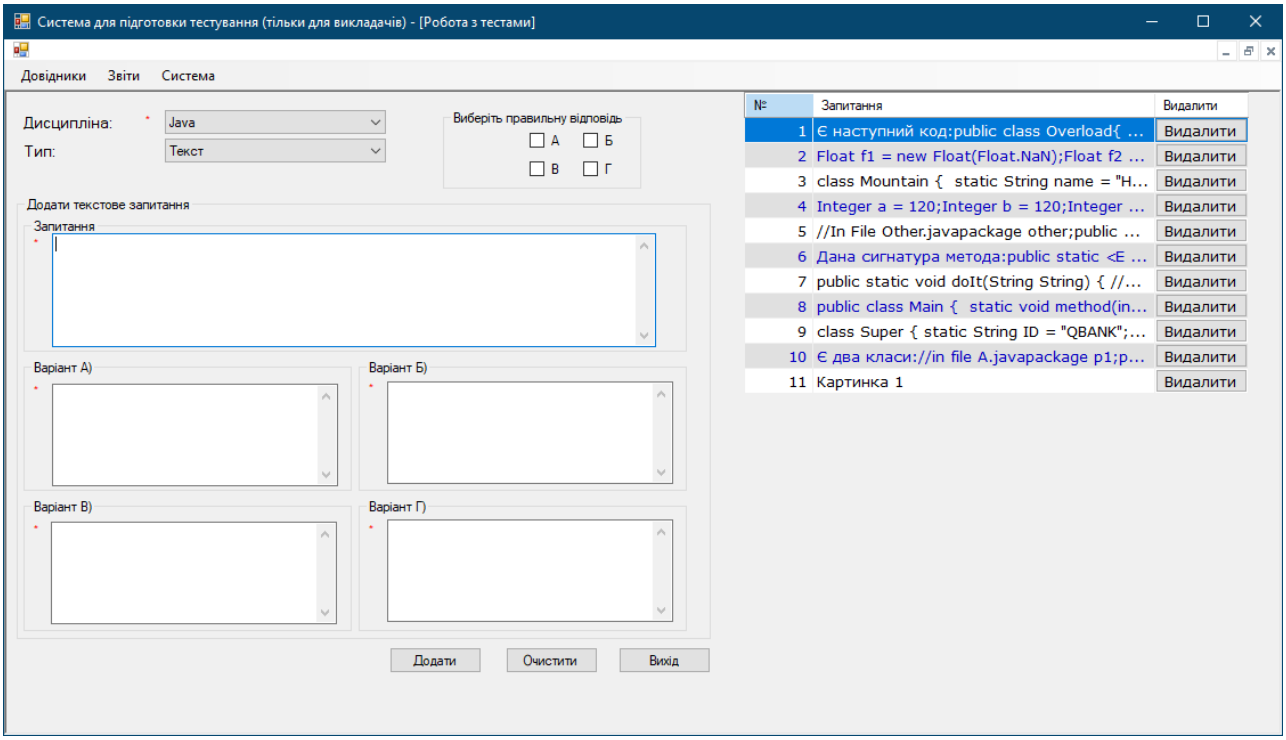

Рисунок 3.16 — Вікно для додавання тестів

Код додавання нового тесту приведений на рисунку 3.17.

```
1 reference
private void AddBtn_Click(object sender, EventArgs e) {
    if (Convert.ToInt32(TestTypeCBox.SelectedValue) == 1) {
        if (IsDataEnteringCorrect()) {
            _TestProvider.InsertTest(QuestionNameTBox.Text, AnswerATBox.Text, AnswerABoolCBox.Checked, AnswerBTBox.Text, AnswerBBoolCBox.Checked,
              AnswerCTBox.Text, AnswerCBoolCBox.Checked, AnswerDTBox.Text, AnswerDBoolCBox.Checked,
              Convert.ToInt32(TestTypeCBox.SelectedValue), Convert.ToInt32(DisciplineCBox.SelectedValue),
             _QuestionImages, _AnswerAImages, _AnswerBImages, _AnswerCImages, _AnswerDImages);
            DataLoad();
            ClearAllControls();
        }
    } else {
        if (IsDataEnteringCorrectImages()) {
            _TestProvider.InsertTest(QuestionNameTBox.Text, AnswerATBox.Text, AnswerABoolCBox.Checked, AnswerBTBox.Text, AnswerBBoolCBox.Checked,
              AnswerCTBox.Text, AnswerCBoolCBox.Checked, AnswerDTBox.Text, AnswerDBoolCBox.Checked,
              Convert.ToInt32(TestTypeCBox.SelectedValue), Convert.ToInt32(DisciplineCBox.SelectedValue),
             _QuestionImages, _AnswerAImages, _AnswerBImages, _AnswerCImages, _AnswerDImages);
            DataLoad();
            ClearAllControls();
        }
    }
}
```

Рисунок 3.17 — Код події «AddBtn_Click» для додавання нового тесту

Як можна побачити із коду програми: у ній окрім валідації даних на введення, є ще й можливість додавання картинок замість тексту. Ця можливість є досить зручною, оскільки дозволяє додавати окрім тексту ще й картинки. Також, вважаю доцільним привести код реалізації події таймера (рис. 3.18).

```
1 reference
private void StartTestTimer_Tick(object sender, EventArgs e) {
    GetAllRunPrograms();
    TimeSpan ts = _EndDate - DateTime.Now;
    TimeLbl.Text = "Залишилось часу: " + ts.Hours + ":" + ts.Minutes + ":" + ts.Seconds;
    if (DateTime.Now > _EndDate) {
        StopTest();
    }
}
```

Рисунок 3.15 — Код події «StartTestTimer_Tick»

Як можна побачити, в ньому стоїть умова перевірки часу виконання тестування. Як тільки час тестування буде вичерпано, тестування буде зупинено.

Для того, щоб ефективно розробити додаток процес було розділено на два етапи.

Перший етап — створення програмної частини, що стосується ролі «викладач», другий етап — ролі «студент».

Модуль, що відповідає за час роботи додатка реалізований у вигляді додаткової панелі. При натисненні на неї запускається відповідний додаток, для прикладу Visual studio, у відповідний файл записуєтья час запуску і роботи з додатком.

Рисунок 3.1 — Модуль контролю за запуском додатку

Після проходження аутентифікації користувачем, система знаючи, яка в нього роль, відповідно відкриє функціонал програми відповідно до неї.

Для ідентифікації користувача системи було створено форму реєстрації (рис. 3.1).

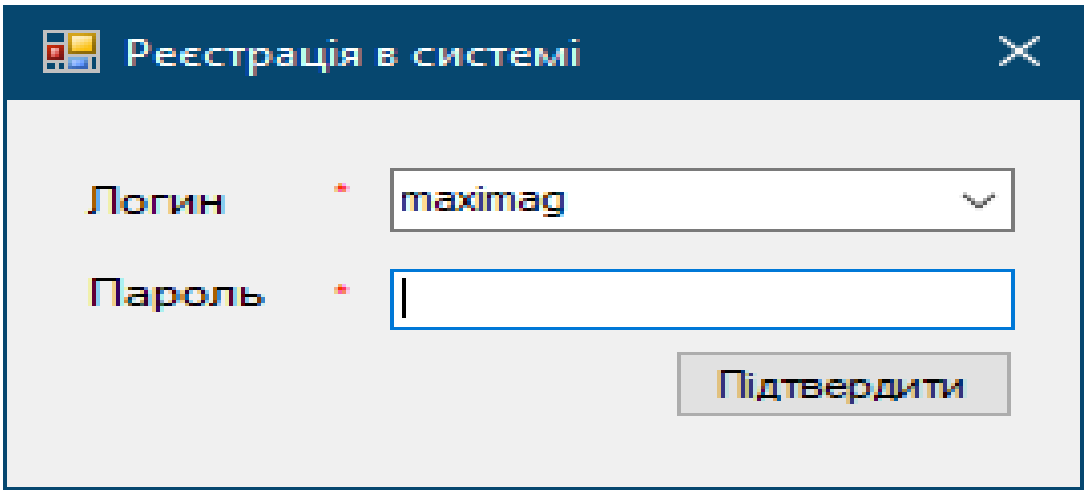

Рисунок 3.1 — Реєстрація користувача в системі

3.4 Тестування програмного модуля

Тестування — процес аналізу і дослідження, який дає змогу виявляти інформацію щодо якості продукту відносно умов, при яких він має застосовуватись. Метод тестування теж містить в собі пошук дефектів, помилок, несправностей. Також являється випробуванням програмних складових для оцінки готовності програмного продукту до використання. Результати тестування оцінюються за наступними характеристиками:

— відповідності вимогам, що надавалися розробниками та проектувальниками;

— відповідності вихідних даних;

— чи прийнятний час виконання функцій;

— практичності;

— відповідності вимогам замовника.

Сукупність тестів навіть для простих програмних компонентів може бути нескінченною, через це тактика тестування повинна полягати в тому, що проводяться тільки необхідні тести з врахуванням доступного часу та ресурсу. І як результат, програмні засоби будуть тестуватись стандартним виконанням програми з метою виявити баги, помилки або інші дефекти.

Є багато видів тестування: одні зазвичай роблять самі розробники, інші — спеціалізовані групи. У нашому випадку буде використовуватись тестування системи.

Тестування системи — виконання програмного забезпечення у його остаточній конфігурації, інтегровано з іншими програмними та апаратними системами.

Одним зі способів вивчення поставленого завдання є дослідити методики «чорної скриньки», основне місце програми тестів «чорної скриньки» — це інтерфейс ПЗ. Ці тести покликані демонструвати:

— як виконані функції програми;

— як надходять вихідні дані;

— як виробляються результати;

— як збережена цілісність зовнішньої інформації.

Тестуючи «чорну скриньку», розглядаються системні характеристики програм, ігнорується їх логічна структура всередині. Повне тестування, як правило, є неможливе. До прикладу, якщо у програмі є 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів буде становити 1010. Тестування «чорної скриньки» не зреагує на багато особливостей програмних помилок.

Тестування програмних засобів буде доречним провести використовуючи методику «чорної скриньки».

Вона основана на використанні шаблонів тестування, бо ж тест-кейсів. Це значить, що буде створено декілько ситуацій, в яких перевіряється

працездатність додатку, коректність основних функцій. Розроблені тест-кейси описано в таблиці 3.1.

Таблиця 3.1 Тест-кейси

<table>
<tr><td rowspan="3">Код тест-кейса</td><td colspan="3">Опис тест-кейса</td></tr>
<tr><td colspan="2">Хід тестування</td><td>Очікуваний результат</td></tr>
<tr><td>Дата тестування</td><td>Результат</td><td>Примітка</td></tr>
<tr><td rowspan="3">001</td><td colspan="3">Перевірка реєстрації користувача програми</td></tr>
<tr><td colspan="2">1. Запустити додаток
2. Провести автентифікацію користувача із роллю «викладач».</td><td>Вивід відповідного вікна із функціоналом для роботи користувача</td></tr>
<tr><td>17.04.2022</td><td>Пройдено</td><td>-</td></tr>
<tr><td rowspan="3">002</td><td colspan="3">Створення нової дисципліни, по якій буде проводитись тестування</td></tr>
<tr><td colspan="2">1. Запустити додаток
2. Провести автентифікацію
3. Додати дисципліну
4. Редагувати дані дисципліни
5. Видалити дисципліну</td><td>Список дисциплін відображається відповідно до введених користувачем даних</td></tr>
<tr><td>17.04.2022</td><td>Пройдено</td><td>-</td></tr>
<tr><td rowspan="3">003</td><td colspan="3">Створення нового тесту, по якому буде проводитись тестування</td></tr>
<tr><td colspan="2">1. Запустити додаток
2. Провести автентифікацію
3. Додати тест
4. Редагувати дані тесту
5. Видалити тест</td><td>Список тестів відображається відповідно до введених користувачем даних</td></tr>
<tr><td>17.04.2022</td><td>Пройдено</td><td>-</td></tr>
<tr><td rowspan="3">004</td><td colspan="3">Перевірка можливості виведення рейтингів тестування</td></tr>
<tr><td colspan="2">1. Запустити додаток
2. Провести автентифікацію
3. Виведення рейтингів тестування користувачів програми</td><td>Успішне виведення рейтингів проведення тестування</td></tr>
<tr><td>17.04.2022</td><td>Пройдено</td><td>-</td></tr>
</table>

3.5 Інструкція користувача

Дана програма відкриває широкі можливості для організації роботи тестування знань, оскільки дозволяє автоматизувати дії проведення тестування та зберігати результати тестування.

На початку необхідно запустити додаток «Система для підготовки тестування». ». Після запуску програми має бути виведене вікно, де користувачу запропонують ввести ім’я та пароль.

Після успішної автентифікації, користувач в залежності від його ролі відкриється відповідний фунціонал програми.

Якщо ж була вибрана роль «викладач», тоді програма такі можливості:

— можливість додавання користувачів системи;

— можливість створення дисциплін;

— можливість перегляду логів системи;

— можливість створення тестування для кожної дисципліни;

— можливість формування звітності по пройдених тестах.

Якщо ж було вибрано роль «студент», тоді програма виведе вікно для тестування знань. В цьому вікні є можливість вибрати дисципліну, та відповідно до неї прийти тестування. В кінці тестування програма видасть його результат.

Однією із важливих функцій програми є можливість створення рейтингів по результатах тестування. Також програма дозволяє переглянути всі програми, що були запущені користувачем, який проходив тестування до початку тесту та під час його проходження.

Програма є досить простою та зручною в користуванні.

ВИСНОВКИ

У ході виконання бакалаврської дипломної роботи проведено розгляд підсистеми для контролю роботи над домашнім завданням.

В першому розділі було проведено аналітичний огляд існуючих програм та сервісів контролю над домашнім завданням. Порівняли існуючі системи, виявили їх основні переваги і недоліки.

В другому розділі роботи проведено розробку структури контролю роботи над домашнім завданням та алгоритмів тестування знань.

В третьому розділі описано програмну реалізацію і тестування платформи Також приведено загальний опис програми розробленої системи. Результати тестування були успішними, помилок не виявлено.

Підсистема для контролю роботи над домашнім завдання дає можливість перевіряти якість виконання роботи студентом та визначити чи запускались при цьому потрібні додатки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Тестирование. Тесты, их классификация. [Электронный ресурс]. – Режим доступа: https://www.psychologos.ru/articles/view/testirovanie,свободный (дата обращения: 19.05.2021);

2. Психологические тесты онлайн. [Электронный ресурс]. – Режим доступа: https://psytests.org, свободный (дата обращения: 19.05.2021)

3. Программа для создания тестов и онлайн тестирования.[Электронный ресурс]. – Режим доступа: https://indigotech.ru;

4. Система тестирования сотрудников. [Электронный ресурс]. –Режим доступа: https://www.startexam.ru;

5. Pro C# 7: With .NET and .NET Core 8-th edition / Andrew Troelse, Philip Japikse. –2017.

6. Об'єктно-орієнтоване програмування [Електронний ресурс] Режим доступу: http://ruslan.rv.ua/python-essential/oop/oop_basis/

7. Об'єктно-орієнтований підхід до програмування [Електронний ресурс] Режим доступу: http://www.znannya.org/?view=csharp-oop

8. Офіційна документація .NET [Електронний ресурс] Режим доступу: https://docs.microsoft.com/ru-ru/dotnet/

9. Мова програмування C# і платформа .NET Core [Електронний ресурс] Режим доступу: https://metanit.com/sharp/tutorial/1.1.php

10. What's new in .NET 4.7 [Електронний ресурс] Режим доступу: https://docs.microsoft.com/en-us/dotnet/core/dotnet-five

11. .NET Core/5+ vs. .NET Framework for server apps [Електронний ресурс] Режим доступу: https://docs.microsoft.com/en-us/dotnet/standard/choosing-core-framework-server

12. Представляем .NET 4.7 [Електронний ресурс] Режим доступу: https://temofeev.ru/info/articles/predstavlyaem-net-5/

ДОДАТОК А

Міністерство освіти та науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ ВНТУ
д.т.н., проф.
_____ О. Д. Азаров
«19» квітня 2021 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання бакалаврської дипломної роботи

«Комп'ютерна підсистема для контролю роботи над домашнім завданням»

08–23.БДР.021.00.000 ПЗ

Науковий керівник к.т.н. доц. каф. ОТ
_____ Снігур А.В.
Студент групи 1КІ–20мсз
_____ Попов В.Д.

Вінниця 2021

1 Підстави для виконання бакалаврської дипломної роботи (БДР)наступні:

— актуальність розробки, яка полягає у необхідності вирішення проблеми можливості тестування знань, на основі якого буде можливість коректно оцінювати знання студентів та суттєво економи час викладачів;

— наказ про затвердження теми бакалаврської дипломної роботи.

2 Мета і призначення БДР наступні:

— метою БДР є розробка програмного забезпечення тестування знань на .NET 4.7 в середовищі Visual Studio 2019, яке дозволить проводити тестування знань студентів та суттєво економи час викладачів.

— призначення розробки полягає у виконанні бакалаврської дипломноїроботи із подальшим впровадженням та розвитком.

3 Вихідні дані для виконання БДР наступні:

— технічний опис програмного застосунку;

— мова програмування C#;

— віконний додаток;

— середовище розробки Microsoft Visual Studio.

4 Вимога до виконання БДР наступна:

— створення віконного додатку;

— можливість створювати та редагувати дисциплін та програм тестування для них;

5 Етапи БДР та очікувані результатиприведені в таблиці А.1. Робота виконується за вісім етапів.

6 Матеріали, що подаються до захисту БДР, наступні:

— пояснювальна записка БДР, графічні і ілюстративні матеріали, протокол попереднього захисту БДР на кафедрі, відгук наукового керівника, відгук опонента, протоколи складання державних екзаменів,

— анотації до БДР українською та іноземною мовами, нормоконтроль про відповідність оформлення ДР діючим вимогам.

Таблиця А.1 — Етапи виконання роботи

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Постановка задачі роботи	09.03.21	09.03.21	Вступ
2	Пошук матеріалів по програми для тестування знань	10.03.21	25.03.21	Розділ 1
3	Розробка структури та алгориитмів роботи тестування знань	26.03.21	28.03.21	Розділ 2
4	Обгрунтування та вибір засобів реалізації системи	29.03.21	04.04.21	Розділ 2
5	Розробка головного вікна, методів створення дисциплін та тестів	05.04.21	29.04.21	Розділ 3
6	Підготовка матеріалів та розробка алгоритму збереження та виведення даних програми	30.04.21	27.05.21	Розділ 3, Працююча система
7	Оформлення пояснювальної записки та ілюстративного матеріалу	28.05.21	06.06.21	Пояснювальна записка
8	Перевірка якості виконання бакалаврської роботи та усунення недоліків	07.06.21	07.06.21	Презентація

7 Порядок контролю виконання та захисту БДР:

— виконання етапів графічної та розрахункової документації ДР контролюється науковим керівником згідно зі встановленими термінами;

— захист ДР відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

8 Вимоги до оформлення БДР:

— викладені в ГОСТ 2.104-95 ЕСКД «Єдина система конструкторської документації»,

— ГОСТ 2.105-95 ЕСКД « Загальні вимоги до текстових документів».

Технічне завдання до виконання отримав _____ Попов В.Д.

ДОДАТОК Б

Код проекту

Файл «RoleApp»

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TestSystem.AppCode {
  class RoleApp {
    public List<Role> GetRoleList() {
      List<Role> RoleList = new List<Role>();
      RoleList.Add(new Role(1, "Викладач"));
      RoleList.Add(new Role(2, "Студент"));
      return RoleList;
    }
  }
}

public class Role {
  private int _RoleId;
  private string _RoleName;
  public Role() {
    _RoleId = 0;
    _RoleName = String.Empty;
  }
  public Role(int RoleId, string RoleName) {
```

```
    _RoleId = RoleId;
    _RoleName = RoleName;
  }

  public int RoleId {
    set { _RoleId = value; }
    get { return _RoleId; }
  }
  public string RoleName {
    set { _RoleName = value; }
    get { return _RoleName; }
  }
}
```

Файл «StatisticBLL»

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using TestSystem.Providers;

namespace TestSystem.BLL {
  class StatisticBLL {
    private StatisticsProvider _StatisticProvider = new StatisticsProvider();
    private DisciplinesProvider _DisciplinesProvider = new DisciplinesProvider();
    private UsersProvider _UsersProvider = new UsersProvider();

    public List<Statistics> GetAllStatistic() {
```

```
      List<Users> usersList = new List<Users>();
      List<Disciplines> disciplinesList = new List<Disciplines>();
      usersList = _UsersProvider.GetAllUsers();
      disciplinesList = _DisciplinesProvider.GetAllDisciplines();

      List<Statistics> statisticList = new List<Statistics>();
      statisticList = _StatisticProvider.GetAllStatistics();

      for (int i = 0; i < statisticList.Count; i++) {
        statisticList[i].FIO = GetFIO(statisticList[i].UsersId, usersList);
        statisticList[i].DisciplineName = GetDisciplineName(statisticList[i].DisciplineId,
disciplinesList);
        statisticList[i].RunTime = TimeSpan.FromSeconds(statisticList[i].Seconds);
      }
      return statisticList;
    }

    private string GetFIO(int UsersId, List<Users> UsersList) {
      for (int i = 0; i < UsersList.Count; i++) {
        if (UsersId == UsersList[i].UsersId) {
          return UsersList[i].FIO;
        }
      }
      return "";
    }

    private string GetDisciplineName(int DisciplineId, List<Disciplines>
DisciplinesList) {
      for (int i = 0; i < DisciplinesList.Count; i++) {
```

```
        if (DisciplineId == DisciplinesList[i].DisciplinesId) {
          return DisciplinesList[i].DisciplinesName;
        }
      }
      return "";
    }

  }
}
```

Файл «DisciplinesForm»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TestSystem.AppCode;
using TestSystem.Providers;

namespace TestSystem.Forms.Dictionary {
  public partial class DisciplinesForm : Form {
    private int _selectedRowIndex = 0;
    private ValidationMy _validation = new ValidationMy();
    private DisciplinesProvider _DisciplinesProvider = new DisciplinesProvider();
    private List<Disciplines> _DisciplinesList = new List<Disciplines>();
```

```
    public DisciplinesForm() {
      InitializeComponent();
      SetDateTimeDTP.Value = new DateTime(1979, 1, 1, 0, 30, 0);
      DataLoad();
    }

    private void AddBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        _DisciplinesProvider.InsertDisciplines(DisciplinesNameTBox.Text,
DescriptionTBox.Text, SetDateTimeDTP.Value);
        DataLoad();
        ClearAllControls();
      }
    }

    private void ClearBtn_Click(object sender, EventArgs e) {
      ClearAllControls();
    }

    private void ExitBtn_Click(object sender, EventArgs e) {
      this.Close();
    }

    private void DataLoad() {
      int firstRowIndex = 0;
      if (DisciplinesGridView.FirstDisplayedScrollingRowIndex > 0) {
        firstRowIndex = DisciplinesGridView.FirstDisplayedScrollingRowIndex;
      }
      try {
```

```
      _DisciplinesList = _DisciplinesProvider.GetAllDisciplines();
      LoadDataInDisciplinesGridView(_DisciplinesList);
      if (_selectedRowIndex == DisciplinesGridView.Rows.Count) {
        _selectedRowIndex = DisciplinesGridView.Rows.Count - 1;
      }
      if (_selectedRowIndex >= 0) {
        DisciplinesGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
        DisciplinesGridView.Rows[_selectedRowIndex].Selected = true;
      }
    } catch { }
  }

  private void LoadDataInDisciplinesGridView(List<Disciplines> DisciplinesList) {
    DisciplinesGridView.DataSource = null;
    DisciplinesGridView.Columns.Clear();
    DisciplinesGridView.AutoGenerateColumns = false;
    DisciplinesGridView.RowHeadersVisible = false;

    DisciplinesGridView.DataSource = DisciplinesList;

    if (DisciplinesList.Count > 0) {
      if (DisciplinesList[0].Message ==
NamesMy.NoDataNames.NoDataInDisciplines) {
        DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
        messageColumn.DataPropertyName = "Message";
        messageColumn.Width = DisciplinesGridView.Width -
NamesMy.SizeOptins.MinusSizePanel;
        DisciplinesGridView.Columns.Add(messageColumn);
      } else {
```

```
        DataGridViewColumn DetailIdColumn = new DataGridViewTextBoxColumn();
```

ДОДАТОК В

```
      DetailIdColumn.DataPropertyName = "DisciplinesId";
        DisciplinesGridView.Columns.Add(DetailIdColumn);
        DisciplinesGridView.Columns[0].Visible = false;

        DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
        numberColumn.HeaderText = "№ ";
        numberColumn.DataPropertyName = "Number";
        numberColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        numberColumn.Width = NamesMy.SizeOptins.NumberSize;
        DisciplinesGridView.Columns.Add(numberColumn);

        DataGridViewColumn DisciplinesNameColumn = new
DataGridViewTextBoxColumn();
        DisciplinesNameColumn.HeaderText = "Дисципліни";
        DisciplinesNameColumn.DataPropertyName = "DisciplinesName";
        DisciplinesNameColumn.Width = NamesMy.SizeOptins.NameSize;
        DisciplinesGridView.Columns.Add(DisciplinesNameColumn);

        DataGridViewColumn EvendDateColumn = new
DataGridViewTextBoxColumn();
        EvendDateColumn.HeaderText = "Час тестування";
        EvendDateColumn.DataPropertyName = "SetDateTime";
        EvendDateColumn.DefaultCellStyle.Format = "HH:mm:ss";
        EvendDateColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        EvendDateColumn.Width = NamesMy.SizeOptins.Date;
```

```
        DisciplinesGridView.Columns.Add(EvendDateColumn);

      }
      for (int i = 0; i < DisciplinesGridView.Columns.Count; i++) {
        DisciplinesGridView.Columns[i].HeaderCell.Style.BackColor =
Color.LightGray;
      }
    }
  }

  private void ClearAllControls() {
    DisciplinesNameTBox.Text = String.Empty;
    DescriptionTBox.Text = String.Empty;
    SetDateTimeDTP.Value = new DateTime(1979, 1, 1, 0, 30, 0);
  }

  private bool IsDataEnteringCorrect() {
    bool isCorrect = true;
    if (_validation.IsDataEntering(DisciplinesNameTBox.Text)) {
      DisciplinesNameValiadtionLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
    } else {
      DisciplinesNameValiadtionLbl.Text =
NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
    }
    return isCorrect;
  }
```

```
    private void DisciplinesGridView_CellClick(object sender,
DataGridViewCellEventArgs e) {
      if (e.RowIndex >= 0 && DisciplinesGridView[0, e.RowIndex].Value.ToString() !=
_DisciplinesList[0].Message) {
        _selectedRowIndex = e.RowIndex;
        UpdateDisciplinesForm updateDisciplinesForm = new
UpdateDisciplinesForm(Convert.ToInt32(DisciplinesGridView[0,
e.RowIndex].Value.ToString()));
        updateDisciplinesForm.ShowDialog();
        DataLoad();
      }
    }
  }
}
```

Файл «TestForm»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TestSystem.AppCode;
using TestSystem.Providers;

namespace TestSystem.Forms.Dictionary {
```

```
public partial class TestForm : Form {
  private int _selectedRowIndex = 0;
  private ValidationMy _validation = new ValidationMy();
  private TestProvider _TestProvider = new TestProvider();
  private List<Test> _TestList = new List<Test>();
  private TestTypeProvider _TestTypeProvider = new TestTypeProvider();
  private List<TestType> _TestTypeList = new List<TestType>();
  private List<TestType> _TestTypeSrchList = new List<TestType>();
  private DisciplinesProvider _DisciplinesProvider = new DisciplinesProvider();
  private List<Disciplines> _DisciplinesList = new List<Disciplines>();
  private List<Disciplines> _DisciplinesSrchList = new List<Disciplines>();
  private bool _IsTestType = false;
  private bool _IsDisciplines = false;

  private string _filter = "Image Files(*.jpg; *.jpeg; *.gif; *.bmp; *.png)|*.jpg; *.jpeg;
*.gif; *.bmp; *.png";
  private byte[] _QuestionImages;
  private byte[] _AnswerAImages;
  private byte[] _AnswerBImages;
  private byte[] _AnswerCImages;
  private byte[] _AnswerDImages;

  public TestForm() {
    InitializeComponent();
    LoadAllDate();
    DataLoad();
  }

  private void AddBtn_Click(object sender, EventArgs e) {
```

```
    if (Convert.ToInt32(TestTypeCBox.SelectedValue) == 1) {
      if (IsDataEnteringCorrect()) {
        _TestProvider.InsertTest(QuestionNameTBox.Text, AnswerATBox.Text,
AnswerABoolCBox.Checked, AnswerBTBox.Text, AnswerBBoolCBox.Checked,
          AnswerCTBox.Text, AnswerCBoolCBox.Checked, AnswerDTBox.Text,
AnswerDBoolCBox.Checked,
          Convert.ToInt32(TestTypeCBox.SelectedValue),
Convert.ToInt32(DisciplineCBox.SelectedValue),
         _QuestionImages, _AnswerAImages, _AnswerBImages, _AnswerCImages,
_AnswerDImages);
        DataLoad();
        ClearAllControls();
      }
    } else {
      if (IsDataEnteringCorrectImages()) {
        _TestProvider.InsertTest(QuestionNameTBox.Text, AnswerATBox.Text,
AnswerABoolCBox.Checked, AnswerBTBox.Text, AnswerBBoolCBox.Checked,
          AnswerCTBox.Text, AnswerCBoolCBox.Checked, AnswerDTBox.Text,
AnswerDBoolCBox.Checked,
          Convert.ToInt32(TestTypeCBox.SelectedValue),
Convert.ToInt32(DisciplineCBox.SelectedValue),
         _QuestionImages, _AnswerAImages, _AnswerBImages, _AnswerCImages,
_AnswerDImages);
        DataLoad();
        ClearAllControls();
      }
    }
  }
```

```
    private void ClearBtn_Click(object sender, EventArgs e) {
      ClearAllControls();
    }

    private void ExitBtn_Click(object sender, EventArgs e) {
      this.Close();
    }

    private void DataLoad() {
      int firstRowIndex = 0;
      if (TestGridView.FirstDisplayedScrollingRowIndex > 0) {
        firstRowIndex = TestGridView.FirstDisplayedScrollingRowIndex;
      }
      try {
        _TestList =
_TestProvider.GetAllTest(Convert.ToInt32(DisciplineCBox.SelectedValue));
        LoadDataInTestGridView(_TestList);
        if (_selectedRowIndex == TestGridView.Rows.Count) {
          _selectedRowIndex = TestGridView.Rows.Count - 1;
        }
        if (_selectedRowIndex >= 0) {
          TestGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
          TestGridView.Rows[_selectedRowIndex].Selected = true;
        }
      } catch { }
    }

    private void LoadDataInTestGridView(List<Test> TestList) {
      TestGridView.DataSource = null;
```

```
        TestGridView.Columns.Clear();
        TestGridView.AutoGenerateColumns = false;
        TestGridView.RowHeadersVisible = false;

        TestGridView.DataSource = TestList;

        if (TestList.Count > 0) {
          if (TestList[0].Message == NamesMy.NoDataNames.NoDataInTest) {
            DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
            messageColumn.DataPropertyName = "Message";
            messageColumn.Width = TestGridView.Width -
NamesMy.SizeOptins.MinusSizePanel;
            TestGridView.Columns.Add(messageColumn);
          } else {
            DataGridViewColumn DetailIdColumn = new DataGridViewTextBoxColumn();
            DetailIdColumn.DataPropertyName = "TestId";
            TestGridView.Columns.Add(DetailIdColumn);
            TestGridView.Columns[0].Visible = false;

            DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
            numberColumn.HeaderText = "№ ";
            numberColumn.DataPropertyName = "Number";
            numberColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
            numberColumn.Width = NamesMy.SizeOptins.NumberSize;
            TestGridView.Columns.Add(numberColumn);

            DataGridViewColumn TestNameColumn = new
DataGridViewTextBoxColumn();
```

```
        TestNameColumn.HeaderText = "Запитання";
        TestNameColumn.DataPropertyName = "QuestionName";
        TestNameColumn.Width = NamesMy.SizeOptins.NameSize;
        TestGridView.Columns.Add(TestNameColumn);

        DataGridViewButtonColumn deleteBtn = new DataGridViewButtonColumn();
        deleteBtn.HeaderText = NamesMy.ProgramButtons.DeleteBtn;
        deleteBtn.Text = NamesMy.ProgramButtons.DeleteBtn;
        deleteBtn.UseColumnTextForButtonValue = true;
        deleteBtn.ToolTipText = NamesMy.ProgramButtons.DeleteBtn;
        deleteBtn.Width = NamesMy.SizeOptins.DeleteBtnSize;
        TestGridView.Columns.Add(deleteBtn);
      }
      for (int i = 0; i < TestGridView.Columns.Count; i++) {
        TestGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
      }
    }
  }

  private void ClearAllControls() {
    QuestionNameTBox.Text = "";
    AnswerATBox.Text = "";
    AnswerBTBox.Text = "";
    AnswerCTBox.Text = "";
    AnswerDTBox.Text = "";
    AnswerABoolCBox.Checked = false;
    AnswerBBoolCBox.Checked = false;
    AnswerCBoolCBox.Checked = false;
    AnswerDBoolCBox.Checked = false;
```

```
    _QuestionImages = null;
    _AnswerAImages = null;
    _AnswerBImages = null;
    _AnswerCImages = null;
    _AnswerDImages = null;
    QuestionImagesPBox.Image = null;
    AnswerAImagesPBox.Image = null;
    AnswerBImagesPBox.Image = null;
    AnswerCImagesPBox.Image = null;
    AnswerDImagesPBox.Image = null;
  }

  private bool IsDataEnteringCorrect() {
    bool isCorrect = true;
    if (_validation.IsDataEntering(QuestionNameTBox.Text)) {
      QuestionNameValiadtionLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
    } else {
      QuestionNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
    }
    if (_validation.IsDataEntering(AnswerATBox.Text)) {
      AnswerAValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
    } else {
      AnswerAValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
    }
    if (_validation.IsDataEntering(AnswerBTBox.Text)) {
      AnswerBValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
```

```
      } else {
        AnswerBValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(AnswerCTBox.Text)) {
        AnswerCValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        AnswerCValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(AnswerDTBox.Text)) {
        AnswerDValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        AnswerDValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }

    public bool IsDataEnteringCorrectImages() {
      bool isCorrect = true;
      if (QuestionImagesPBox.Image !=null) {
        QuestionImagesValidationLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
      } else {
        QuestionImagesValidationLbl.Text =
NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
```

```
    if (AnswerAImagesPBox.Image != null) {
      AnswerAImagesValidationLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
    } else {
      AnswerAImagesValidationLbl.Text =
NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
    }
    if (AnswerBImagesPBox.Image != null) {
      AnswerBImagesValidationLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
    } else {
      AnswerBImagesValidationLbl.Text =
NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
    }
    if (AnswerCImagesPBox.Image != null) {
      AnswerCImagesValidationLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
    } else {
      AnswerCImagesValidationLbl.Text =
NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
    }
    if (AnswerDImagesPBox.Image != null) {
      AnswerDImagesValidationLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
    } else {
```

```
        AnswerDImagesValidationLbl.Text =
NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }


    private void LoadAllDate() {
      GraphicsGBox.Location = new System.Drawing.Point(9, 92);
      _TestTypeList = _TestTypeProvider.GetAllTestType();
      TestTypeCBox.DataSource = _TestTypeList;
      TestTypeCBox.ValueMember = "TestTypeId";
      TestTypeCBox.DisplayMember = "TestTypeName";
      _IsTestType = true;

      _DisciplinesList = _DisciplinesProvider.GetAllDisciplines();
      DisciplineCBox.DataSource = _DisciplinesList;
      DisciplineCBox.ValueMember = "DisciplinesId";
      DisciplineCBox.DisplayMember = "DisciplinesName";
      _IsDisciplines = true;
    }

    private void SearchBtn_Click(object sender, EventArgs e) {
      DataLoad();
    }

    private void QuestionImagesPBox_Click(object sender, EventArgs e) {
      OpenFileDialog openFileDialog1 = new OpenFileDialog();
```

```
  openFileDialog1.Filter = _filter;
  if (openFileDialog1.ShowDialog() == DialogResult.OK) {
    QuestionImagesPBox.SizeMode = PictureBoxSizeMode.StretchImage;
    QuestionImagesPBox.Image = new Bitmap(openFileDialog1.FileName);
    _QuestionImages = System.IO.File.ReadAllBytes(openFileDialog1.FileName);
  }
}

private void AnswerAImagesPBox_Click(object sender, EventArgs e) {
  OpenFileDialog openFileDialog1 = new OpenFileDialog();
  openFileDialog1.Filter = _filter;
  if (openFileDialog1.ShowDialog() == DialogResult.OK) {
    AnswerAImagesPBox.SizeMode = PictureBoxSizeMode.StretchImage;
    AnswerAImagesPBox.Image = new Bitmap(openFileDialog1.FileName);
    _AnswerAImages = System.IO.File.ReadAllBytes(openFileDialog1.FileName);
  }
}

private void AnswerBImagesPBox_Click(object sender, EventArgs e) {
  OpenFileDialog openFileDialog1 = new OpenFileDialog();
  openFileDialog1.Filter = _filter;
  if (openFileDialog1.ShowDialog() == DialogResult.OK) {
    AnswerBImagesPBox.SizeMode = PictureBoxSizeMode.StretchImage;
    AnswerBImagesPBox.Image = new Bitmap(openFileDialog1.FileName);
    _AnswerBImages = System.IO.File.ReadAllBytes(openFileDialog1.FileName);
  }
}

private void AnswerCImagesPBox_Click(object sender, EventArgs e) {
```

ДОДАТОК Г

```
  OpenFileDialog openFileDialog1 = new OpenFileDialog();
  openFileDialog1.Filter = _filter;
  if (openFileDialog1.ShowDialog() == DialogResult.OK) {
    AnswerCImagesPBox.SizeMode = PictureBoxSizeMode.StretchImage;
    AnswerCImagesPBox.Image = new Bitmap(openFileDialog1.FileName);
    _AnswerCImages = System.IO.File.ReadAllBytes(openFileDialog1.FileName);
  }
}

private void AnswerDImagesPBox_Click(object sender, EventArgs e) {
  OpenFileDialog openFileDialog1 = new OpenFileDialog();
  openFileDialog1.Filter = _filter;
  if (openFileDialog1.ShowDialog() == DialogResult.OK) {
    AnswerDImagesPBox.SizeMode = PictureBoxSizeMode.StretchImage;
    AnswerDImagesPBox.Image = new Bitmap(openFileDialog1.FileName);
    _AnswerDImages = System.IO.File.ReadAllBytes(openFileDialog1.FileName);
  }
}

private void TestTypeCBox_SelectedValueChanged(object sender, EventArgs e) {
  if (_IsTestType) {
    if (Convert.ToInt32(TestTypeCBox.SelectedValue) == 1) {
      AddTextGBox.Visible = true;
      GraphicsGBox.Visible = false;
    } else {
      AddTextGBox.Visible = false;
      GraphicsGBox.Visible = true;
    }
```

```
    }
  }

  private void TestGridView_CellClick(object sender, DataGridViewCellEventArgs
e) {
    if (e.ColumnIndex == 3) {
      if (MessageBox.Show("Ви дійсно бажаєте видалити це замовлення зі
списку?", "Видалити", MessageBoxButtons.YesNo) == DialogResult.Yes) {
        int selectedTestId = Convert.ToInt32(TestGridView[0,
e.RowIndex].Value.ToString());
        _TestProvider.DeleteTestByTestId(selectedTestId);
        DataLoad();
      }
    }
  }

  private void DisciplineCBox_SelectedValueChanged(object sender, EventArgs e) {
    if (_IsDisciplines) {
      DataLoad();
    }
  }
 }
}
```

Файл «UpdateDisciplinesForm»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using TestSystem.AppCode;

namespace TestSystem.Providers {
  class SettingsProvider {
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];

    public void InsertSettings(int DisciplineId, int Seconds) {
      string SqlString = "INSERT INTO Settings (DisciplineId, Seconds) Values(?, ?)";
      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          cmd.CommandType = CommandType.Text;
          cmd.Parameters.AddWithValue("DisciplineId", DisciplineId);
          cmd.Parameters.AddWithValue("Seconds", Seconds);
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }

    public List<Settings> GetAllSettings() {
      int i = 0;
      string SqlString = "SELECT * " +
      "FROM Settings ORDER BY DisciplineId ASC";
      List<Settings> listAllSettings = new List<Settings>();
```

```
    public void UpdateTest(string QuestionName, string AnswerA, bool AnswerABool,
string AnswerB, bool AnswerBBool,
      string AnswerC, bool AnswerCBool, string AnswerD, bool AnswerDBool, int
TestTypeId, int DisciplineId,
      byte[] QuestionImages, byte[] AnswerAImages, byte[] AnswerBImages, byte[]
AnswerCImages, byte[] AnswerDImages, int TestId) {
      string SqlString = "UPDATE Tests SET QuestionName=?, AnswerA=?,
AnswerABool=?, AnswerB=?, AnswerBBool=?, AnswerC=?,  " +
    "AnswerCBool=?, AnswerD=?, AnswerDBool=?, TestTypeId=?, DisciplineId=?," +
    "QuestionImages=?, AnswerAImages=?, AnswerBImages=?, AnswerCImages=?,
AnswerDImages=? WHERE TestId=?";

      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          cmd.CommandType = CommandType.Text;
          cmd.Parameters.AddWithValue("TestName", QuestionName);
          cmd.Parameters.AddWithValue("AnswerA", AnswerA);
          cmd.Parameters.AddWithValue("AnswerABool", AnswerABool);
          cmd.Parameters.AddWithValue("AnswerB", AnswerB);
          cmd.Parameters.AddWithValue("AnswerBBool", AnswerBBool);
          cmd.Parameters.AddWithValue("AnswerC", AnswerC);
          cmd.Parameters.AddWithValue("AnswerCBool", AnswerCBool);
          cmd.Parameters.AddWithValue("AnswerD", AnswerD);
          cmd.Parameters.AddWithValue("AnswerDBool", AnswerDBool);
          cmd.Parameters.AddWithValue("TestTypeId", TestTypeId);
          cmd.Parameters.AddWithValue("DisciplineId", DisciplineId);
          if (QuestionImages == null) {
```

ДОДАТОК Д

```
            cmd.Parameters.AddWithValue("QuestionImages",
Encoding.Default.GetBytes(""));
        } else {
            cmd.Parameters.AddWithValue("QuestionImages", QuestionImages);
        }
        if (AnswerAImages == null) {
            cmd.Parameters.AddWithValue("AnswerAImages",
Encoding.Default.GetBytes(""));
        } else {
            cmd.Parameters.AddWithValue("AnswerAImages", AnswerAImages);
        }
        if (AnswerBImages == null) {
            cmd.Parameters.AddWithValue("AnswerBImages",
Encoding.Default.GetBytes(""));
        } else {
            cmd.Parameters.AddWithValue("AnswerBImages", AnswerBImages);
        }
        if (AnswerCImages == null) {
            cmd.Parameters.AddWithValue("AnswerCImages",
Encoding.Default.GetBytes(""));
        } else {
            cmd.Parameters.AddWithValue("AnswerCImages", AnswerCImages);
        }
        if (AnswerDImages == null) {
            cmd.Parameters.AddWithValue("AnswerDImages",
Encoding.Default.GetBytes(""));
        } else {
            cmd.Parameters.AddWithValue("AnswerDImages", AnswerDImages);
```

```
        }

        cmd.Parameters.AddWithValue("FirstName", FirstName);
        cmd.Parameters.AddWithValue("LastName", LastName);
        cmd.Parameters.AddWithValue("UsersName", UsersName);
        cmd.Parameters.AddWithValue("UsersPassword",
_encryptData.Encrypt(UsersPassword));
        cmd.Parameters.AddWithValue("RoleId", RoleId);
        cmd.Parameters.AddWithValue("Description", Description);
        cmd.Parameters.AddWithValue("UsersId", UsersId);
        conn.Open();
        cmd.ExecuteNonQuery();
        conn.Close();
      }
    }
  }

  public void DeleteUsersByUsersId(int UsersId) {
    string SqlString = "DELETE FROM Users WHERE UsersId=" +
UsersId.ToString();
    using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
      using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
        conn.Open();
        cmd.ExecuteNonQuery();
        conn.Close();
      }
    }
  }
```

```
  }
}
            StartTestTimer.Start();
          }
          _RunProgramsList.Add(currentProgramsList[i]);
        }
      }
    }

    for (int i = 0; i < _RunProgramsList.Count; i++) {
      if (IsThisProgramsOnStart(_RunProgramsList[i].ProgramsName,
currentProgramsList)) {
        _RunProgramsList[i].RunTimes = DateTime.Now -
_RunProgramsList[i].StartProgram;
      }
    }
  }

  private bool IsThisProgramsOnStart(string ProgramName, List<Programs>
ProgramsList) {
    for (int i = 0; i < ProgramsList.Count; i++) {
      if (ProgramName == ProgramsList[i].ProgramsName) {
        return true;
      }
    }
    return false;
  }

  private bool IsAllow(string ProgramsName) {
```

```
        if (ProgramsName == "devenv" || ProgramsName == "WINWORD" ||
ProgramsName == "EXCEL") {
          return true;
        }
      return false;
    }


    private List<Programs> GetProgramsList() {
      List<Programs> runPrograms = new List<Programs>();
      foreach (var item in System.Diagnostics.Process.GetProcesses()) {
        Programs oneProgram = new Programs();
        if (item.MainWindowTitle != "") {
          oneProgram.ProgramsName = item.ProcessName;
          oneProgram.MainWindowTitle = item.MainWindowTitle;
          oneProgram.StartProgram = DateTime.Now;
          runPrograms.Add(oneProgram);
        }
      }
      return runPrograms;
    }
```

ДОДАТОК Ж

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ
ЗАПОЗИЧЕНЬ

Назва роботи: _____

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 80,5% Схожість 19,5%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ _____
(підпис)

Керівник роботи _____ _____
(підпис)