МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЖИТОМИРСЬКИЙ АГРОТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ

ВІДДІЛЕННЯ
«ІНЖЕНЕРНА ІНФРАСТРУКТУРА ТА КОМП’ЮТЕРНІ НАУКИ»

ЦИКЛОВА КОМІСІЯ
«ІНЖЕНЕРНА ІНФРАСТРУКТУРА ТА КОМП'ЮТЕРНІ НАУКИ»

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломної роботи освітнього ступеня «фаховий молодший бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
на тему:

«Програма для автоматизації підготовки до іспитів»

Виконав здобувач освіти групи П-42
Червінський Андрій Михайлович

Керівник роботи:
Устименко Ярослав Іванович

Рецензент:
Устименко Людмила Миколаївна

Житомир – 2025

Зміст

РЕФЕРАТ

Обсяг: 58 стор., 20 рис., 2 таблиці, 1 додаток, 20 джерел

У дипломній роботі розроблено програму для автоматизації підготовки до іспитів на основі Windows Forms і .NET 8, що включає модулі створення та проходження тестів. Метою роботи було створення зручного інструменту для студентів і викладачів із функціями введення питань, збереження тестів у JSON, таймера (1 хвилина на питання) і прогрес-барів. Проведено аналіз аналогів (Quizlet, Kahoot!, Moodle), обґрунтовано вибір технологій за критеріями простоти й продуктивності.

Розроблено класи TestQuestion, Form1 і Form2, реалізовано алгоритми з часовою складністю O(1) для основних операцій і O(n) для роботи з файлами. Програма підтримує локальне зберігання даних і кирилицю (UTF-8). Інтерфейс включає текстові поля, радіокнопки, списки та прогрес-бари, забезпечуючи інтуїтивність. Описано системні вимоги (Windows 10+, 2 ГБ ОЗП) та інструкцію з використання.

Програма є ефективним інструментом для самоперевірки знань, із потенціалом для розширення (кросплатформність, нові типи питань). Робота підтверджує практичну цінність розробки для навчального процесу.

Ключові слова: автоматизація, підготовка до іспитів, Windows Forms, .NET 8, JSON, таймер, прогрес-бар, тестування.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API: Application Programming Interface - інтерфейс програмування додатків, який використовується для отримання погодних даних з відповідного джерела.

UI: User Interface - інтерфейс користувача, який включає в себе всі елементи та взаємодію з користувачем.

GPS: Global Positioning System - глобальна система позиціонування, використовується для визначення місцезнаходження користувача.

JSON: JavaScript Object Notation - формат обміну даними, часто використовується для передачі погодних даних з сервера.

HTTP: Hypertext Transfer Protocol - протокол передачі гіпертексту, використовується для взаємодії з сервером для отримання погодних даних.

UI/UX: User Interface/User Experience - дизайн інтерфейсу та користувацький досвід, важливі аспекти при розробці зручного та привабливого додатку.

SDK: Software Development Kit - комплект розробника програмного забезпечення, який може включати в себе інструменти для розробки додатків для певної платформи або сервісу.

SSL: Secure Sockets Layer - протокол шару захищеного з'єднання, який забезпечує захищеність передачі даних між додатком та сервером.

UX Research: User Experience Research - дослідження користувацького досвіду, що включає в себе аналіз потреб користувачів та їхніх вимог до додатку.

ВСТУП

У сучасному світі, де обсяги інформації стрімко зростають, ефективна підготовка до іспитів стає важливим завданням для студентів, учнів та фахівців, які прагнуть підвищити свою кваліфікацію. Традиційні методи підготовки, такі як конспектування, повторення матеріалу чи використання паперових тестів, часто є трудомісткими та не завжди дозволяють оптимально організувати навчальний процес. У зв'язку з цим актуальність створення програмного забезпечення для автоматизації підготовки до іспитів набуває особливого значення.

Автоматизація підготовки до іспитів за допомогою спеціалізованих програм дає змогу систематизувати навчальний матеріал, створювати інтерактивні тести, відстежувати прогрес і оптимізувати час, витрачений на вивчення. Такі програми дозволяють не лише підвищити ефективність засвоєння знань, але й зробити процес підготовки більш зручним і мотивуючим завдяки інтерактивним елементам, таким як таймери, прогрес-бари чи статистика результатів.

Метою даної дипломної роботи є розробка програми для автоматизації підготовки до іспитів, яка поєднує в собі можливості створення тестів та їх проходження. Програма розробляється з використанням технології Windows Forms на платформі .NET 8, що забезпечує зручний графічний інтерфейс, високу продуктивність і підтримку сучасних стандартів розробки програмного забезпечення.

Основними завданнями роботи є:

- аналіз вимог до програмного забезпечення для підготовки до іспитів;
- проектування зручного інтерфейсу для створення та проходження тестів;
- реалізація функціоналу для збереження та завантаження тестів у форматі JSON;
- впровадження таймера та прогрес-барів для контролю часу та відстеження прогресу;

- тестування програми для забезпечення її стабільності та зручності використання.

Розроблена програма складатиметься з двох основних модулів: модуля створення тестів, який дозволить викладачам або користувачам формувати питання з варіантами відповідей, та модуля проходження тестів, який надасть можливість користувачам перевіряти свої знання з обмеженням часу та відображенням прогресу. Використання Windows Forms забезпечить простоту у використанні, а .NET 8 гарантуватиме сумісність із сучасними операційними системами та можливість подальшого розширення функціоналу.

Очікується, що розроблена програма стане корисним інструментом для студентів, учнів та викладачів, сприяючи підвищенню ефективності підготовки до іспитів і забезпечуючи зручний спосіб самоперевірки знань. У подальших розділах роботи будуть детально розглянуті етапи аналізу, проектування, реалізації та тестування програми, а також оцінено її практичну цінність.

РОЗДІЛ 1. ОГЛЯД ТА АНАЛІЗ ТЕХНОЛОГІЙ РОЗРОБКИ

1.1. Постановка основної задачі розробки

Розробка програми для автоматизації підготовки до іспитів є актуальним завданням, спрямованим на підвищення ефективності навчального процесу. Основна мета полягає у створенні зручного та функціонального програмного забезпечення, яке дозволить користувачам створювати тести, проходити їх із контролем часу та відстежувати прогрес. Програма має відповідати сучасним вимогам до інтерфейсу, бути простою у використанні та забезпечувати надійне зберігання даних.

Основна задача полягає у проектуванні та реалізації програми з двома ключовими модулями:

1. **Модуль створення тестів**:
   - Надання можливості вводити питання, чотири варіанти відповідей та вказувати правильну відповідь.
   - Забезпечення функцій редагування, видалення та збереження тестів у файл формату JSON.
   - Реалізація зручного механізму завантаження раніше створених тестів для подальшого редагування.
2. **Модуль проходження тестів**:
   - Відображення питань по одному з можливістю вибору відповіді.
   - Контроль часу на кожне питання (1 хвилина) з візуалізацією через прогрес-бар.
   - Відстеження прогресу проходження тесту за допомогою додаткового прогрес-бару.
   - Підрахунок правильних відповідей та відображення результатів після завершення тесту.

Вимоги до програми

Для виконання поставленої задачі програма повинна відповідати наступним вимогам:

1. **Функціональні вимоги**:
   - Можливість створення тестів із підтримкою кирилиці та збереження їх у файли через діалогові вікна.
   - Інтерактивне проходження тестів із таймером і відображенням прогресу.
   - Збереження результатів тестування у вигляді статистики (кількість правильних відповідей, відсоток успішності).
   - Кожне питання має містити рівно чотири варіанти відповідей, один із яких є правильним.
   - Таймер для кожного питання становить 1 хвилину, що є компромісом між достатнім часом для обдумування та необхідністю підтримувати темп тестування.

2. **Нефункціональні вимоги**:
   - Зручний і інтуїтивно зрозумілий графічний інтерфейс.
   - Висока продуктивність і стабільність роботи.
   - Підтримка кодування UTF-8 для коректного відображення тексту українською мовою.
   - Сумісність із сучасними версіями операційних систем Windows.

Постановка задачі розробки програми для автоматизації підготовки до іспитів визначає чіткі функціональні та нефункціональні вимоги, які забезпечать створення ефективного інструменту для навчання. Вибір Windows Forms і .NET 8 як основних технологій обґрунтований їхньою зручністю, продуктивністю та підтримкою сучасних стандартів. У наступних підрозділах будуть розглянуті детальний аналіз альтернативних технологій і обґрунтування остаточного вибору інструментів розробки.

1.2. Огляд та порівняння аналогічних систем.

Для визначення оптимального підходу до розробки програми для автоматизації підготовки до іспитів необхідно провести аналіз існуючих аналогічних систем. Це дозволяє виявити їхні сильні та слабкі сторони, а також визначити унікальні особливості, які можна реалізувати у власній програмі. У цьому підрозділі розглядаються популярні системи для створення та проходження тестів, їх функціонал, технології, переваги та недоліки, а також проводиться порівняння з проєктованою програмою.

1.2.1. Огляд аналогічних систем

1. Quizlet

Quizlet — це вебплатформа та мобільний додаток для створення навчальних матеріалів, зокрема флеш-карток, тестів і вправ. Вона широко використовується студентами для підготовки до іспитів.

Технології: Вебтехнології (HTML, CSS, JavaScript), хмарне зберігання даних, мобільні додатки на iOS та Android.

Функціонал:

- Створення тестів із питаннями різних типів (множинний вибір, правда/брехня, відкрита відповідь).
- Можливість ділитися тестами з іншими користувачами.
- Інтерактивні режими навчання (флеш-картки, ігри).
- Відстеження прогресу та статистики.

Переваги:

- Кросплатформність (доступ із браузера та мобільних пристроїв).
- Велика бібліотека готових тестів, створених іншими користувачами.
- Інтеграція з хмарними сервісами для синхронізації даних.

Недоліки:

- Обмежений функціонал у безкоштовній версії (наприклад, відсутність офлайн-доступу).
- Залежність від інтернет-з’єднання.
- Складність створення складних тестів із таймером для кожного питання.

2. Kahoot!

Kahoot! — це платформа для створення інтерактивних вікторин, які використовуються в освітніх закладах для тестування знань у форматі гри.

Технології: Вебтехнології, хмарні сервіси, мобільні додатки.

Функціонал:

- Створення тестів із множинним вибором і таймером.
- Проведення тестів у реальному часі з групою учасників.
- Гейміфікація процесу тестування (рейтинги, бали).
- Звіти про результати для викладачів.

Переваги:

- Висока залученість користувачів завдяки ігровому формату.
- Можливість використання в класі або дистанційно.
- Простий інтерфейс для створення тестів.

Недоліки:

- Обов’язкова наявність інтернет-з’єднання.
- Обмежена підтримка офлайн-режиму.
- Орієнтація на групове тестування, що не завжди підходить для індивідуальної підготовки.

3. Moodle

Moodle — це система управління навчанням (LMS), яка включає модуль для створення та проходження тестів.

Технології: PHP, MySQL, вебсервер (Apache/Nginx), хмарне або локальне розгортання.

Функціонал:

- Створення тестів із різними типами питань (множинний вибір, есе, відповідність).
- Налаштування таймерів для всього тесту або окремих питань.
- Автоматична перевірка відповідей і збереження результатів.
- Інтеграція з іншими навчальними модулями (форуми, журнали оцінок).

Переваги:

- Широкі можливості налаштування тестів.
- Підтримка великих груп користувачів.
- Безкоштовність і відкритий вихідний код.

Недоліки:

- Складність налаштування та адміністрування для невеликих проєктів.
- Вимагає сервера для розгортання.
- Інтерфейс може бути менш інтуїтивним для початківців.

4. Microsoft Forms

Microsoft Forms — це вебсервіс для створення опитувань і тестів, інтегрований із Microsoft 365.

Технології: Хмарні технології Microsoft Azure, вебінтерфейс.

Функціонал:

- Створення тестів із множинним вибором, відкритими питаннями тощо.
- Автоматична оцінка відповідей.
- Експорт результатів у Excel.
- Інтеграція з Teams і OneDrive.

Переваги:

- Простота використання.
- Хмарне зберігання даних.
- Інтеграція з екосистемою Microsoft.

Недоліки:

- Обмежений функціонал для складних тестів (наприклад, відсутність прогрес-барів для питань).
- Залежність від підписки Microsoft 365 для повного доступу.
- Відсутність офлайн-режиму.

1.2.2. Порівняння аналогів із проєктованою програмою

Таблиця 1.2.2.1. Порівняння аналогів

Критерій	Quizlet	Kahoot!	Moodle	Microsoft Forms
Кросплатформність	Так	Так	Так	Так
Офлайн-режим	Ні	Ні	Так	Ні
Налаштування таймера	Ні	Так	Так	Так (на тест)
Прогрес тестування	Так	Так	Так	Ні
Локальне зберігання	Ні	Ні	Так	Ні
Простота інтерфейсу	Висока	Висока	Середня	Висока

Аналіз аналогів показав, що більшість популярних систем (Quizlet, Kahoot!, Microsoft Forms) орієнтовані на хмарні технології та групове використання, що робить їх залежними від інтернет-з'єднання та обмежує офлайн-функціонал. Moodle пропонує ширші можливості, але її складність налаштування робить її менш придатною для індивідуальних користувачів.

Огляд аналогічних систем підтвердив актуальність розробки програми з локальним зберіганням даних і таймером на кожне питання. Проєктована програма поєднує простоту використання, офлайн-функціонал і специфічні можливості (прогрес-бари, таймер на питання), які вигідно відрізняють її від аналогів. У наступних підрозділах буде детально розглянуто вибір технологій і проектування програми для реалізації поставлених завдань.

1.3. Вибір та обґрунтування технологій розробки

Для успішної реалізації програми для автоматизації підготовки до іспитів необхідно обрати технології, які забезпечать зручність розробки, високу продуктивність, стабільність і відповідність вимогам проєкту. У цьому підрозділі розглядаються можливі варіанти технологій, проводиться їх порівняння та обґрунтовується вибір інструментів для створення програми, а саме технології Windows Forms на платформі .NET 8.

1.3.1. Аналіз вимог до технологій

На основі поставлених завдань (підрозділ 1.1) сформовано такі вимоги до технологій розробки:

- Програма повинна мати зручний і інтуїтивно зрозумілий інтерфейс для створення та проходження тестів.

- Підтримка збереження та завантаження тестів у файли без залежності від інтернет-з’єднання.
- Можливість серіалізації та десеріалізації даних у форматі JSON із підтримкою кирилиці (кодування UTF-8).
- Швидка обробка операцій із невеликою кількістю даних (до кількох сотень питань у тесті).
- Реалізація таймерів для обмеження часу на питання та прогрес-барів для відображення стану.
- Робота на сучасних операційних системах Windows, які є основною платформою цільової аудиторії.
- Технологія має дозволяти швидке створення прототипу та підтримувати подальше розширення функціоналу.

1.3.2. Огляд можливих технологій

Розглянемо кілька технологій для розробки графічних додатків, які можуть бути використані для реалізації програми:

1. Windows Forms (.NET)

Windows Forms — це бібліотека для створення графічних інтерфейсів у .NET, яка дозволяє швидко розробляти додатки з підтримкою елементів керування, таких як кнопки, текстові поля та прогрес-бари.

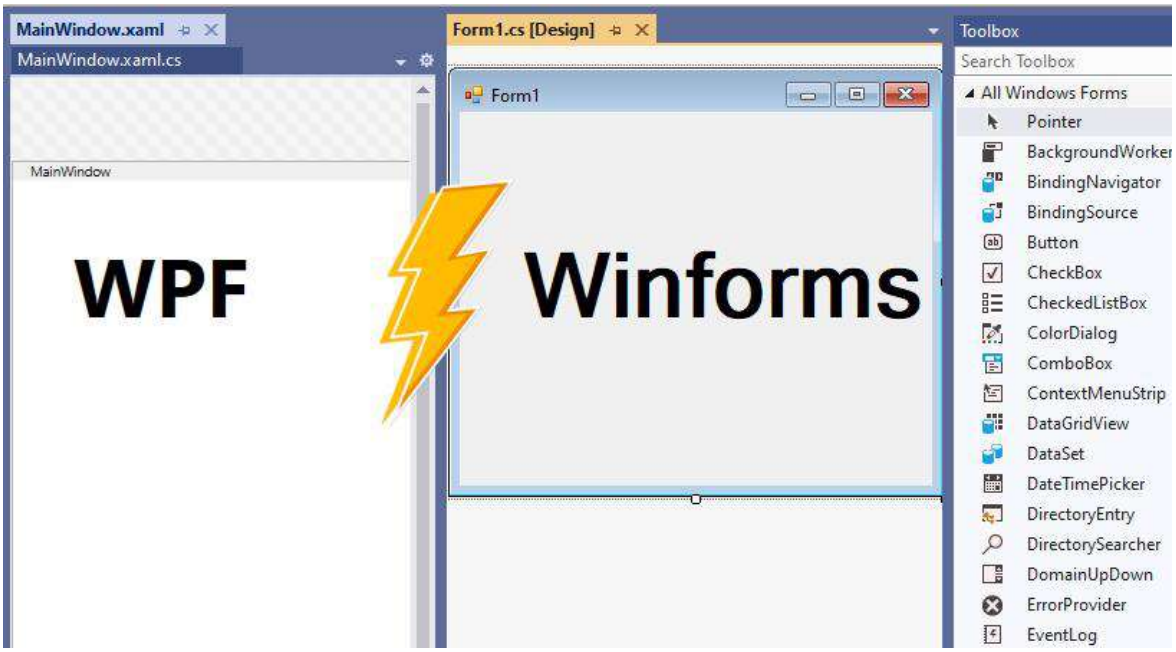

Рисунок 1.3.1 – Windows Forms

Переваги:

- Простота створення інтерфейсу завдяки вбудованому дизайнеру у Visual Studio.
- Нативна інтеграція з .NET для роботи з файлами, JSON і таймерами.
- Підтримка локального зберігання даних без додаткових залежностей.
- Висока продуктивність для невеликих додатків.
- Сумісність із Windows.

Недоліки:

- Обмежена кросплатформність (лише Windows).
- Менш сучасний вигляд інтерфейсу порівняно з новими технологіями (наприклад, WPF).

2. WPF (.NET)

Windows Presentation Foundation — це сучасніша альтернатива Windows Forms, яка використовує XAML для створення гнучких і візуально привабливих інтерфейсів.

Рисунок 1.3.2 – WPF

Переваги:

- Підтримка сучасного дизайну з анімаціями та кастомізацією.
- Інтеграція з .NET для роботи з JSON і таймерами.
- Можливість створення адаптивних інтерфейсів.

Недоліки:

- Вища складність розробки порівняно з Windows Forms.
- Більше споживання ресурсів, що може бути надмірним для простого додатку.
- Обмежена кросплатформність (основна підтримка Windows).

3. WinUI 3

WinUI 3 — це нова бібліотека для створення сучасних Windows-додатків із підтримкою Windows 11 і Fluent Design.

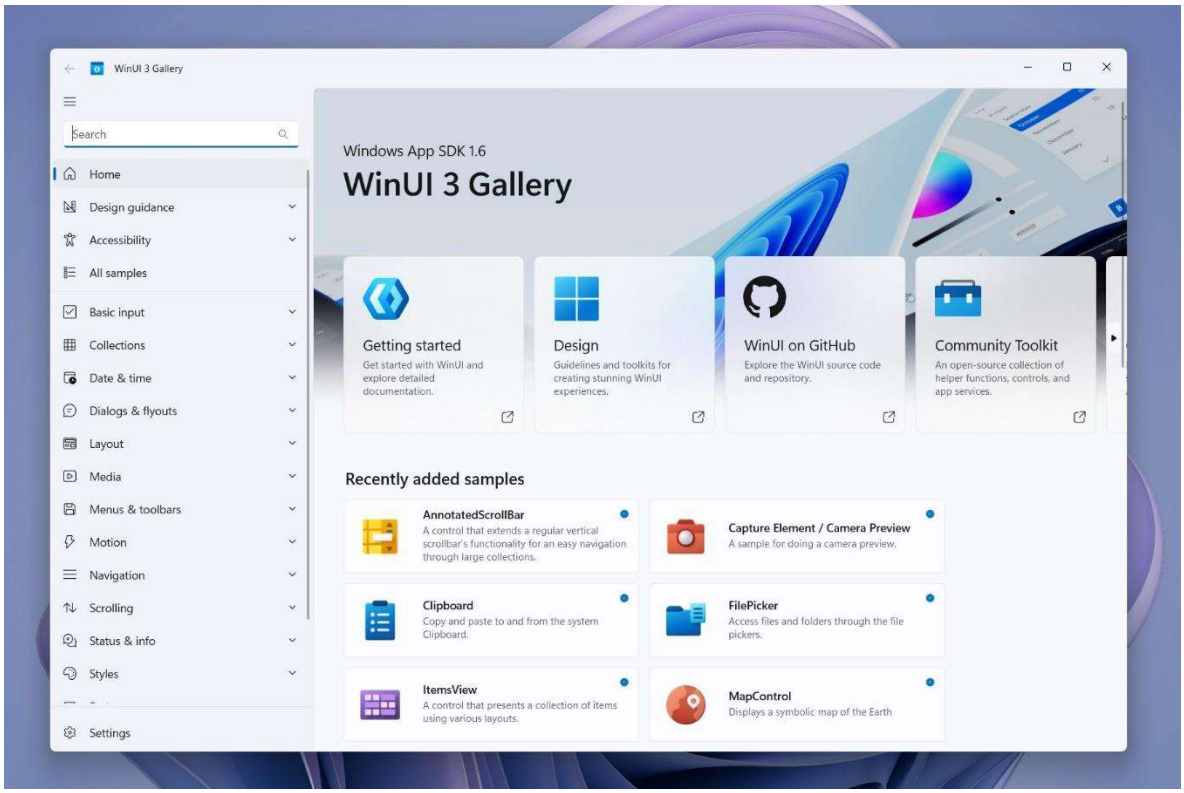

Рисунок 1.3.3 – WinUI 3

Переваги:

- Сучасний вигляд інтерфейсу, адаптований до Windows 11.
- Підтримка .NET для обробки даних.
- Потенційна кросплатформність у майбутніх версіях.

Недоліки:

- Відносно нова технологія з меншою кількістю документації.
- Складніша розробка порівняно з Windows Forms.
- Обмежена підтримка старіших версій Windows.

4. Qt (C++)

Qt — це кросплатформна бібліотека для створення графічних додатків із використанням C++.

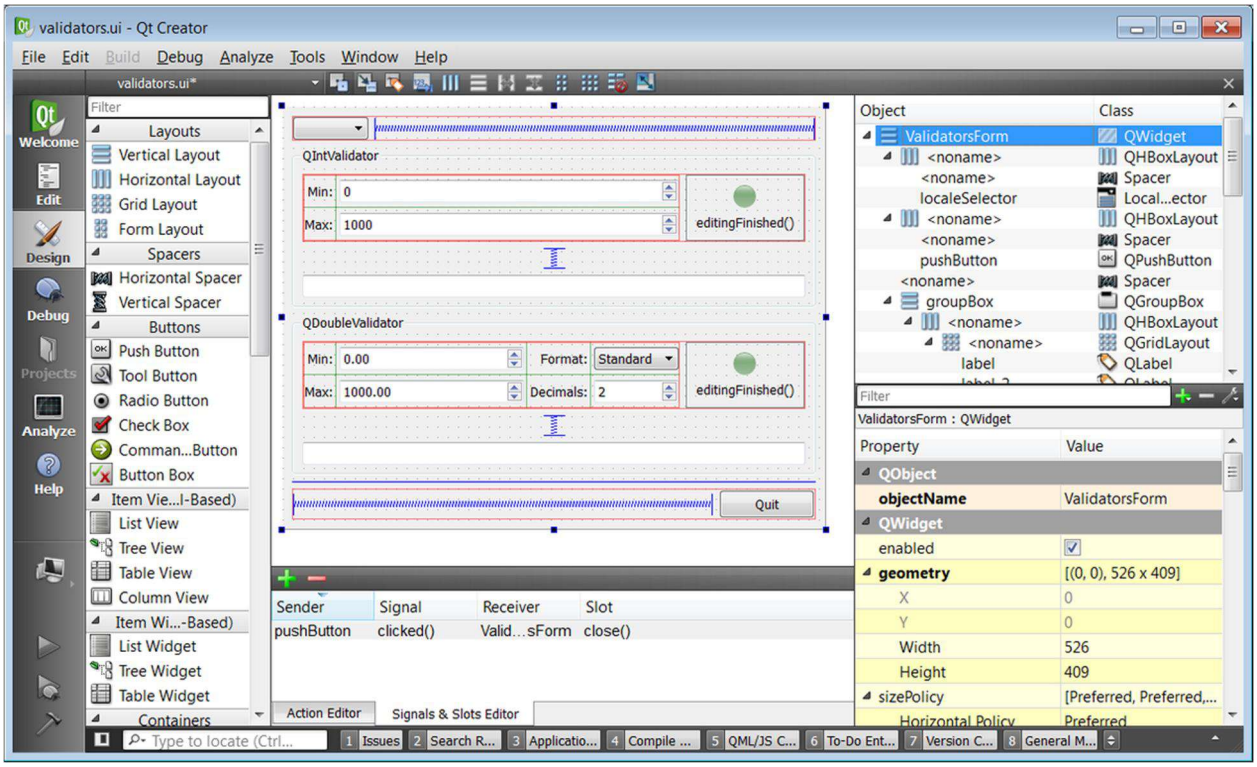

Рисунок 1.3.4 – Qt

Переваги:

- Кросплатформність (Windows, macOS, Linux).
- Висока продуктивність завдяки C++.
- Гнучкість у дизайні інтерфейсу.

Недоліки:

- Складність розробки через використання C++.
- Потреба в додаткових бібліотеках для роботи з JSON.
- Менш інтуїтивний процес створення інтерфейсу порівняно з .NET.

5. Electron (JavaScript)

Electron дозволяє створювати кросплатформні десктопні додатки з використанням вебтехнологій (HTML, CSS, JavaScript).

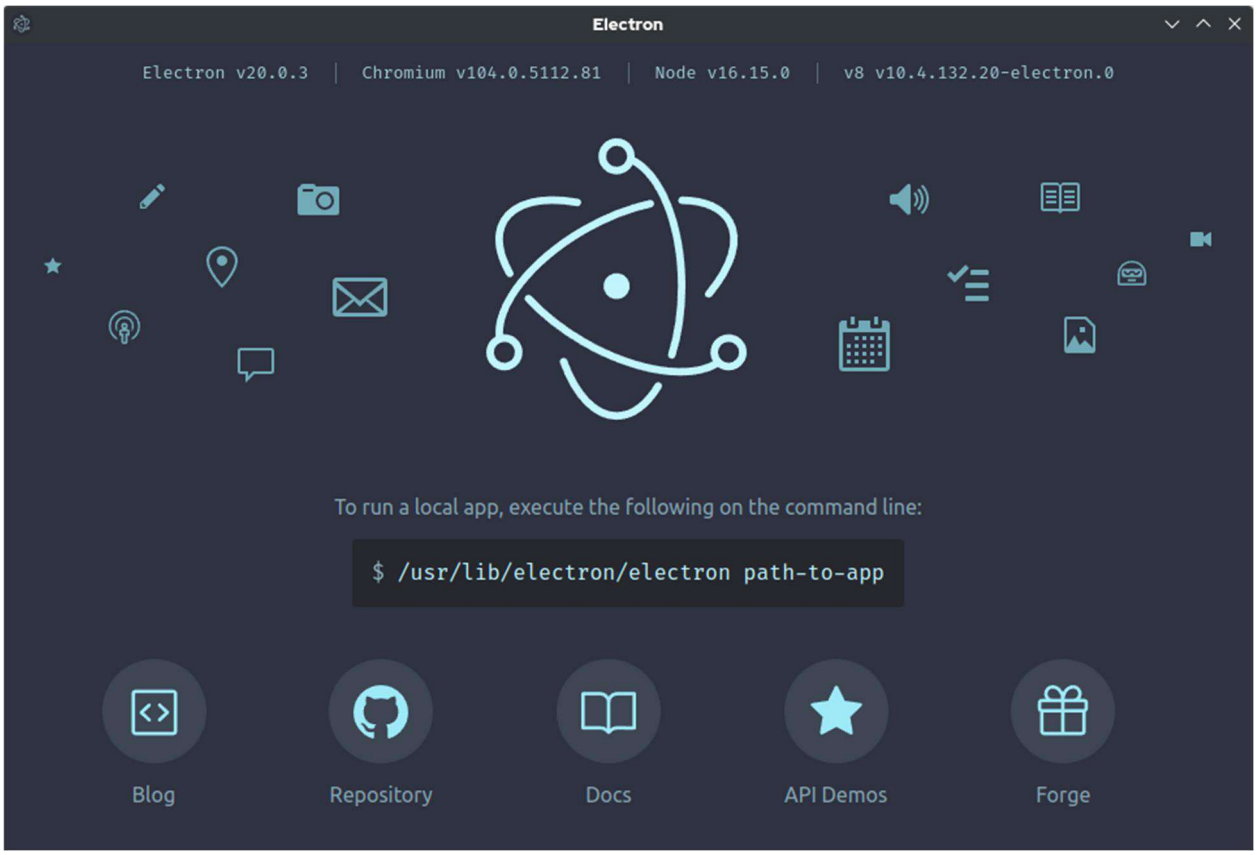

Рисунок 1.3.5 – Electron

Переваги:

- Кросплатформність.
- Простота роботи з JSON.
- Гнучкість у дизайні завдяки вебтехнологіям.

Недоліки:

- Високе споживання пам’яті.
- Залежність від браузерного рушія (Chromium).
- Менш нативний вигляд на Windows.

1.3.3. Порівняння технологій

Для вибору оптимальної технології порівняємо їх за ключовими критеріями:

Таблиця 1.3.3.1. Порівняння технологій

Критерій	Windows Forms	WPF	WinUI 3	Qt (C++)	Electron
Простота розробки	Висока	Середня	Низька	Низька	Середня
Продуктивність	Висока	Середня	Висока	Висока	Низька
Кросплатформність	Ні	Ні	Обмежена	Так	Так
Підтримка JSON	Так	Так	Так	З бібліотеками	Так
Нативний вигляд (Windows)	Так	Так	Так	Обмежено	Обмежено
Підтримка таймерів	Так	Так	Так	Так	Так
Сумісність із Windows	Висока	Висока	Висока	Висока	Висока

1.3.4. Обґрунтування вибору Windows Forms і .NET 8

На основі аналізу вимог і порівняння технологій обрано Windows Forms на платформі .NET 8 з таких причин:

- Windows Forms має вбудований дизайнер у Visual Studio, який дозволяє швидко створювати графічні елементи (кнопки, текстові поля, прогрес-бари).
- Логіка програми легко реалізується завдяки інтеграції з C# і .NET.
- Підтримка локального зберігання тестів у форматі JSON через вбудовані бібліотеки System.Text.Json.
- Наявність компонентів для реалізації таймерів (Timer) і прогрес-барів (ProgressBar).
- Коректна робота з кирилицею завдяки кодуванню UTF-8.
- Windows Forms забезпечує високу швидкість роботи для невеликих додатків, таких як програма для створення та проходження тестів.
- .NET 8 оптимізує продуктивність і підтримує сучасні стандарти розробки.
- Оскільки цільова аудиторія (студенти та викладачі) переважно використовує Windows, обмеження кросплатформності не є критичним.
- Windows Forms гарантує нативний вигляд і стабільну роботу на всіх сучасних версіях Windows.

- .NET 8 є актуальною платформою з довготривалою підтримкою (LTS), що забезпечує стабільність і можливість оновлення програми в майбутньому.
- Вбудовані бібліотеки для серіалізації, роботи з файлами та обробки винятків спрощують реалізацію.

Хоча WPF і WinUI 3 пропонують сучасніший вигляд, їхня складність і більші вимоги до ресурсів не виправдані для програми з відносно простим функціоналом. Qt і Electron забезпечують кросплатформність, але їхня розробка займає більше часу, а продуктивність (особливо в Electron) нижча. Windows Forms є оптимальним вибором, враховуючи баланс між швидкістю розробки, продуктивністю та відповідністю вимогам.

Вибір Windows Forms на платформі .NET 8 обґрунтований простотою розробки, високою продуктивністю, підтримкою локального зберігання даних і сумісністю з Windows. Ця технологія дозволяє ефективно реалізувати модулі створення та проходження тестів із таймерами та прогрес-барами. Використання JSON і UTF-8 забезпечує універсальність і коректну роботу з текстом. У наступних розділах буде розглянуто детальне проектування програми та реалізацію її функціоналу.

Висновки до розділу 1

Аналіз і постановка задачі розробки програми для автоматизації підготовки до іспитів визначили необхідність створення двох модулів: для створення тестів і їх проходження з таймером та прогрес-барами. Огляд аналогічних систем (Quizlet, Kahoot!, Moodle, Microsoft Forms) показав, що більшість рішень орієнтовані на хмарні технології, тоді як проєктована програма вирізняється офлайн-доступністю та локальним зберіганням даних у форматі JSON. Вибір Windows Forms на платформі .NET 8 обґрунтовано простотою розробки, високою продуктивністю, підтримкою кирилиці та сумісністю з Windows, що відповідає вимогам цільової аудиторії. Ці технології забезпечують ефективну реалізацію функціоналу програми з можливістю подальшого розширення.

РОЗДІЛ 2. ПРОЕКТУВАННЯ І РОЗРОБКА

2.1. Вибір алгоритмів та їх ефективність

Проектування програми для автоматизації підготовки до іспитів вимагає ретельного вибору алгоритмів, які забезпечать швидкодію, простоту реалізації та відповідність функціональним вимогам. У цьому підрозділі розглядаються основні алгоритми, використані для реалізації модулів створення та проходження тестів, а також аналізується їхня ефективність із точки зору часової та просторової складності.

Для модуля створення тестів основними операціями є додавання, редагування, видалення питань, а також збереження та завантаження тестів у форматі JSON. Додавання питання передбачає створення об’єкта, що містить текст питання, чотири варіанти відповідей і правильну відповідь, з подальшим збереженням у динамічний масив. Ця операція виконується шляхом додавання елемента до списку, що в .NET реалізується через клас List<TestQuestion>. Часова складність додавання становить O(1) у середньому, оскільки List забезпечує амортизовану константу для операцій додавання в кінець. Просторова складність залежить від кількості питань і становить O(n), де n — кількість питань у тесті.

Редагування питання полягає у виборі елемента зі списку за індексом і оновленні його полів. Оскільки доступ до елемента за індексом у List має часову складність O(1), а оновлення даних є константною операцією, загальна складність редагування також становить O(1). Просторова складність залишається O(1), оскільки не створюються додаткові структури даних.

Видалення питання реалізується через видалення елемента зі списку за індексом. У List операція видалення має часову складність O(n) у найгіршому випадку через необхідність зсуву елементів після видаленого. Однак, враховуючи невелику кількість питань (десятки або сотні), ця операція залишається прийнятною за швидкодією. Просторова складність видалення становить O(1), оскільки не потребує додаткової пам’яті.

Збереження тесту у файл JSON передбачає серіалізацію списку питань. Використання бібліотеки System.Text.Json забезпечує перетворення об’єктів у текстовий формат із подальшим записом у файл. Часова складність серіалізації пропорційна кількості питань і розміру даних, тобто O(n), де n — кількість питань. Запис у файл також має складність O(n) через послідовну обробку даних. Просторова складність залежить від розміру JSON-рядка і становить O(n). Завантаження тесту виконується зворотним процесом — десеріалізацією JSON у список об’єктів, що має аналогічну часову та просторову складність O(n).

Для модуля проходження тестів ключовими операціями є відображення питань, перевірка відповідей і підрахунок результатів. Відображення питання полягає у виборі об’єкта зі списку за поточним індексом і заповненні елементів інтерфейсу (текст питання, варіанти відповідей). Оскільки доступ до елемента за індексом у List має складність O(1), а оновлення інтерфейсу є константною операцією, загальна часова складність становить O(1). Просторова складність також O(1), оскільки не створюються додаткові структури.

Перевірка відповіді користувача виконується шляхом порівняння обраного індексу з правильним індексом відповіді, що є операцією з часовою складністю O(1). Збільшення лічильника правильних відповідей також займає константний час. Просторова складність перевірки становить O(1), оскільки використовується лише одна змінна для підрахунку.

Обмеження часу на кожне питання реалізується через таймер із періодичністю 1 секунда. Алгоритм таймера зменшує змінну часу та оновлює прогрес-бар, що виконується з часовою складністю O(1) для кожної ітерації. Якщо час закінчується, програма автоматично переходить до наступного питання, що еквівалентно натисканню кнопки "Наступне" і має складність O(1). Просторова складність таймера становить O(1).

Прогрес проходження тесту відображається через додатковий прогрес-бар, який розраховує відсоток виконаних питань як відношення поточного індексу до загальної кількості питань. Ця операція включає ділення та

оновлення значення прогрес-бару, що виконується за O(1). Просторова складність також становить O(1).

Підрахунок результатів після завершення тесту передбачає виведення кількості правильних відповідей і відсотка успішності. Оскільки лічильник правильних відповідей ведеться під час проходження, підсумковий розрахунок зводиться до ділення та форматування тексту, що має часову складність O(1). Просторова складність становить O(1), оскільки результат зберігається у кількох змінних.

Ефективність обраних алгоритмів є високою, оскільки більшість операцій мають константну часову складність O(1), а операції з файлами та списком питань мають лінійну складність O(n), що прийнятно для невеликих обсягів даних (до кількох сотень питань). Використання List<TestQuestion> забезпечує гнучкість і швидкий доступ до даних, а бібліотека System.Text.Json оптимізує роботу з файлами JSON. Просторова складність усіх операцій залишається мінімальною, що дозволяє програмі ефективно працювати на сучасних комп’ютерах із обмеженими ресурсами.

Обрані алгоритми забезпечують баланс між простотою реалізації, швидкодією та відповідністю вимогам програми. Вони дозволяють реалізувати всі необхідні функції — від створення тестів до їх проходження з таймером і відображенням прогресу — без надмірного ускладнення коду чи використання ресурсів. У подальших підрозділах буде детально розглянуто проектування архітектури програми та реалізацію її компонентів.

2.2. Спроєктовані класи програми

Розробка програми для автоматизації підготовки до іспитів потребує чіткої структуризації коду для забезпечення модульності, зрозумілості та можливості подальшого розширення. У цьому підрозділі описуються спроєктовані класи програми, їх призначення, структура та взаємозв'язки. Програма складається з двох основних модулів: створення тестів і проходження тестів, що реалізуються через окремі форми Windows Forms та допоміжний клас для представлення даних.

Основний клас даних: TestQuestion

Для представлення одного питання тесту спроєктовано клас TestQuestion. Цей клас є центральною моделлю даних, яка використовується як у модулі створення, так і в модулі проходження тестів. Він містить усі необхідні атрибути питання та забезпечує серіалізацію в JSON для збереження у файли.

```
4 references
public class TestQuestion
{
    4 references
    public string? Question { get; set; }
    6 references
    public List<string>? Options { get; set; }
    3 references
    public int CorrectAnswerIndex { get; set; }
}
```

Рисунок 2.2.1 – Клас TestQuestion

Структура класу TestQuestion:

- Поле Question типу string зберігає текст питання.
- Поле Options типу List<string> містить список із чотирьох варіантів відповідей.
- Поле CorrectAnswerIndex типу int вказує індекс правильної відповіді (0–3).

Клас TestQuestion не містить методів, оскільки його основна роль — зберігання даних. Завдяки простій структурі він легко серіалізується бібліотекою System.Text.Json, що забезпечує збереження та завантаження тестів у форматі JSON. Використання List<string> для Options дозволяє гнучко працювати з варіантами відповідей, а int для CorrectAnswerIndex гарантує компактність і однозначність визначення правильної відповіді.

Модуль створення тестів: Form1

Модуль створення тестів реалізовано через клас Form1, який успадковується від System.Windows.Forms.Form. Цей клас відповідає за графічний інтерфейс і логіку створення, редагування, видалення, збереження та завантаження тестів.

```
3 references
partial class Form1
{
    /// <summary> Required designer variable.
    private System.ComponentModel.IContainer components = null;
    /// <summary> Clean up any resources being used.
    0 references
    protected override void Dispose(bool disposing)...
    Windows Form Designer generated code
    private TextBox txtQuestion;
    private TextBox txtOption1;
    private TextBox txtOption2;
    private TextBox txtOption3;
    private TextBox txtOption4;
    private RadioButton rbCorrect1;
    private RadioButton rbCorrect2;
    private RadioButton rbCorrect3;
    private RadioButton rbCorrect4;
    private TableLayoutPanel tableLayoutPanel1;
    private Label label2;
    private ListBox lstQuestions;
    private Label label1;
    private Button btnSaveTest;
    private Button btnAddQuestion;
    private Button btnDeleteQuestion;
    private Button btnEditQuestion;
    private Button btnLoadTest;
}
```

Рисунок 2.2.2 – Клас Form1

Структура класу Form1:

- Поле questions типу List<TestQuestion> зберігає список усіх питань тесту.
- Поле selectedQuestionIndex типу int вказує індекс питання, яке редагується, або -1, якщо редагування не активне.
- Елементи інтерфейсу включають текстові поля для введення питання та варіантів відповідей (txtQuestion, txtOption1–txtOption4), радіокнопки для вибору правильної відповіді (rbCorrect1–rbCorrect4), список для відображення питань (lstQuestions) і кнопки для дій (btnAddQuestion, btnSaveTest, btnLoadTest, btnDeleteQuestion, btnEditQuestion).
- Метод btnAddQuestion_Click обробляє додавання нового питання або збереження змін під час редагування. Він перевіряє вибір правильної відповіді, створює або оновлює об’єкт TestQuestion і додає його до списку.
- Метод btnSaveTest_Click відкриває діалогове вікно SaveFileDialog і серіалізує список questions у JSON-файл із кодуванням UTF-8.
- Метод btnLoadTest_Click використовує OpenFileDialog для завантаження JSON-файлу, десеріалізує його в questions і оновлює lstQuestions.
- Метод btnDeleteQuestion_Click видаляє вибране питання зі списку questions і lstQuestions.
- Метод btnEditQuestion_Click заповнює поля інтерфейсу даними вибраного питання для редагування.
- Допоміжні методи GetCorrectAnswerIndex, ClearInputs і ClearRadioButtons забезпечують отримання індексу правильної відповіді, очищення полів і скидання радіокнопок.

Клас Form1 інкапсулює всю логіку модуля створення тестів, використовуючи TestQuestion як основну модель даних. Він взаємодіє з користувачем через графічний інтерфейс і забезпечує збереження даних у файли.

Модуль проходження тестів: Form2

Модуль проходження тестів реалізовано через клас Form2, який також успадковується від System.Windows.Forms.Form. Цей клас відповідає за відображення питань, обробку відповідей, контроль часу та підрахунок результатів.

```
3 references
partial class Form2
{
    /// <summary> Required designer variable.
    private System.ComponentModel.IContainer components = null;

    /// <summary> Clean up any resources being used.
    0 references
    protected override void Dispose(bool disposing)...

    Windows Form Designer generated code

    private TableLayoutPanel tableLayoutPanel1;
    private Button btnNext;
    private Button btnLoadTest;
    private System.Windows.Forms.Timer questionTimer;
    private Label lblProgress;
    private ProgressBar progressBarTest;
    private Label lblQuestion;
    private Label label2;
    private RadioButton rbOption4;
    private RadioButton rbOption3;
    private Label label1;
    private RadioButton rbOption2;
    private RadioButton rbOption1;
    private ProgressBar progressBarTime;
}
```

Рисунок 2.2.3 – клас Form2

Структура класу Form2:

- Поле questions типу List<TestQuestion> зберігає список питань, завантажених із файлу.
- Поле currentQuestionIndex типу int вказує на поточне питання.

- Поле correctAnswers типу int підраховує кількість правильних відповідей.
- Поле timeLeft типу int відстежує залишок часу на поточне питання (у секундах).
- Елементи інтерфейсу включають мітку для питання (lblQuestion), радіокнопки для варіантів відповідей (rbOption1–rbOption4), кнопку для переходу до наступного питання (btnNext), мітку для прогресу (lblProgress), прогрес-бар для часу (progressBarTime), прогрес-бар для тесту (progressBarTest), кнопку для завантаження тесту (btnLoadTest) і таймер (questionTimer).
- Метод btnLoadTest_Click відкриває OpenFileDialog, завантажує JSON-файл, десеріалізує його в questions і запускає відображення першого питання.
- Метод DisplayQuestion відображає поточне питання, оновлює радіокнопки, прогрес-бари та запускає таймер.
- Метод btnNext_Click перевіряє вибір відповіді, зупиняє таймер, викликає CheckAnswer і переходить до наступного питання.
- Метод QuestionTimer_Tick зменшує timeLeft, оновлює progressBarTime і, якщо час закінчується, автоматично переходить до наступного питання.
- Метод CheckAnswer порівнює вибрану відповідь із правильною і збільшує correctAnswers за потреби.
- Метод UpdateTestProgress розраховує відсоток виконаних питань і оновлює progressBarTest.
- Метод ShowResults виводить підсумки тесту (правильні відповіді, відсоток).
- Допоміжний метод ClearRadioButtons скидає вибір радіокнопок.

Клас Form2 повністю інкапсулює логіку проходження тестів, використовуючи TestQuestion для доступу до даних питань. Він забезпечує

інтерактивність через таймер і прогрес-бари, що підвищує зручність для користувача.

Клас TestQuestion є спільною моделлю даних для Form1 і Form2, що забезпечує єдність формату даних. Form1 створює та змінює об’єкти TestQuestion, записуючи їх у JSON-файл. Form2 зчитує ці файли, використовуючи TestQuestion для відображення та обробки питань. Обидва класи Form1 і Form2 є незалежними формами Windows Forms, які не взаємодіють напряму, але обмінюються даними через JSON-файли. Така архітектура забезпечує модульність і можливість окремого тестування кожного модуля.

Спроєктовані класи програми — TestQuestion, Form1 і Form2 — чітко розподіляють функціонал між моделлю даних і модулями створення та проходження тестів. TestQuestion забезпечує просту та універсальну структуру для зберігання питань, тоді як Form1 і Form2 реалізують логіку користувацького інтерфейсу та обробки даних. Використання Windows Forms і .NET 8 дозволяє ефективно інтегрувати графічні елементи, таймери та роботу з файлами, що відповідає вимогам програми. У подальших підрозділах буде описано детальну реалізацію та тестування цих класів.

2.3. Програмна реалізація

У цьому підрозділі описується процес програмної реалізації програми для автоматизації підготовки до іспитів на основі спроєктованих класів (підрозділ 2.2). Програма розроблена з використанням Windows Forms на платформі .NET 8 у середовищі Visual Studio 2022. Реалізація охоплює два основних модулі: створення тестів (Form1) і проходження тестів (Form2), а також модель даних TestQuestion.

2.3.1. Налаштування проєкту

Розробка виконувалася у Visual Studio 2022 шляхом створення двох окремих проєктів Windows Forms (.NET 8): TestCreator для модуля створення тестів і TestTaker для модуля проходження тестів. Обидва проєкти

використовують спільний клас TestQuestion, який визначено в кожному проєкті для уникнення залежностей між ними. У файлі проєкту (.csproj) указано підтримку .NET 8.

```
<PropertyGroup>
  <OutputType>WinExe</OutputType>
  <TargetFramework>net8.0-windows</TargetFramework>
  <UseWindowsForms>true</UseWindowsForms>
</PropertyGroup>
```

Рисунок 2.3.1 – Налаштування проєкту

Бібліотека System.Text.Json використовується для серіалізації та десеріалізації даних у JSON, а System.Text — для роботи з кодуванням UTF-8.

2.3.2. Реалізація модуля створення тестів (Form1)

Модуль створення тестів реалізовано через клас Form1 у проєкті TestCreator. Інтерфейс створено за допомогою дизайнера Visual Studio і включає текстові поля (txtQuestion, txtOption1–txtOption4), радіокнопки (rbCorrect1–rbCorrect4), список (lstQuestions) і кнопки (btnAddQuestion, btnSaveTest, btnLoadTest, btnDeleteQuestion, btnEditQuestion).

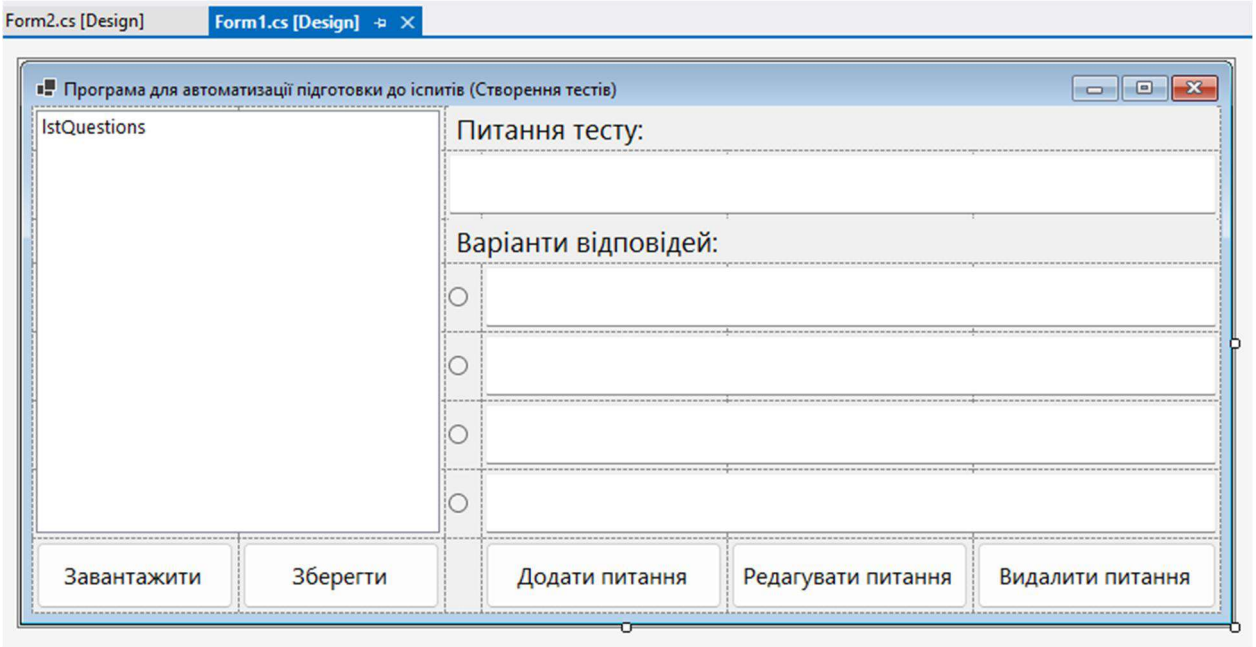

Рисунок 2.3.2 – Дизайнер Form1

Особливості реалізації:

- Використання SaveFileDialog і OpenFileDialog забезпечує вибір файлів для збереження та завантаження.
- Кодування UTF-8 і параметр UnsafeRelaxedJsonEscaping у JsonSerializerOptions гарантують коректне збереження кирилиці.
- Перевірка умов (наприклад, вибір правильної відповіді чи наявність питань) запобігає помилкам.
- Зміна тексту кнопки btnAddQuestion між "Додати питання" і "Зберегти зміни" полегшує розуміння режиму редагування.

2.3.4. Реалізація модуля проходження тестів (Form2)

Модуль проходження тестів реалізовано через клас Form2 у проєкті TestTaker. Інтерфейс включає мітку (lblQuestion), радіокнопки (rbOption1–rbOption4), кнопку (btnNext), мітку прогресу (lblProgress), прогрес-бари (progressBarTime, progressBarTest), кнопку завантаження (btnLoadTest) і таймер (questionTimer).

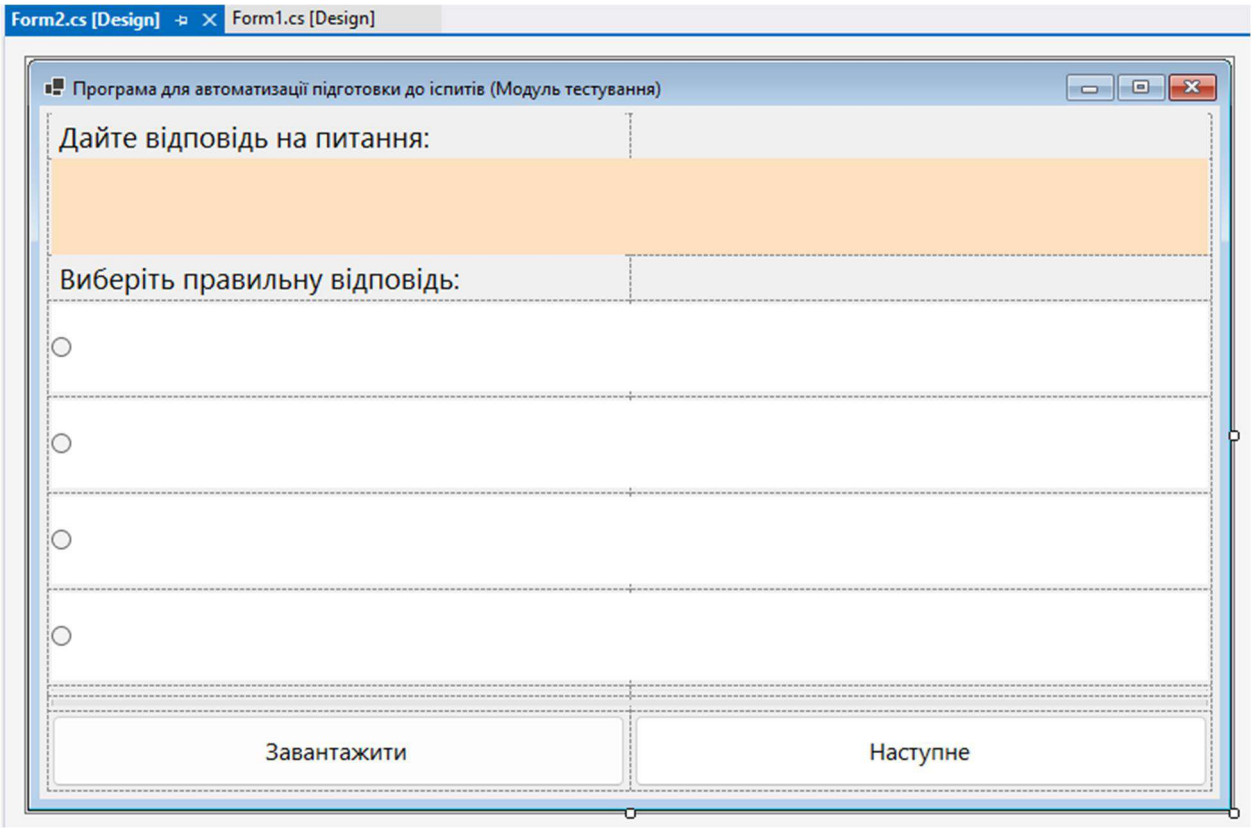

Рисунок 2.3.3 – Дизайнер Form2

Особливості реалізації:

- Таймер (questionTimer) оновлюється кожну секунду, зменшуючи timeLeft і синхронізуючи progressBarTime.
- progressBarTest відображає прогрес тесту у відсотках, розрахованих як (currentQuestionIndex + 1) / questions.Count * 100.
- Перевірка вибору відповіді перед переходом до наступного питання запобігає пропуску.
- Після закінчення часу автоматичний перехід до наступного питання імітує поведінку реального тесту.

2.3.4. Особливості роботи з JSON

Обидва модулі використовують System.Text.Json для роботи з файлами. У Form1 параметр UnsafeRelaxedJsonEscaping дозволяє зберігати кирилицю без екранування (наприклад, "Питання" замість \u041f\u0438\u0442\u0430\u043d\u043d\u044f). Кодування UTF-8 забезпечує коректне читання та запис файлів. Обробка винятків у btnLoadTest_Click захищає від пошкоджених або некоректних файлів.

2.3.5. Організація коду

Код організовано модульно: кожен клас (Form1, Form2, TestQuestion) відповідає за свою частину функціоналу. Методи поділено на логічні блоки (обробка подій, допоміжні операції), що полегшує підтримку та розширення. Використання using для діалогових вікон і потоків забезпечує належне звільнення ресурсів.

Програмна реалізація виконана відповідно до спроєктованих класів, забезпечуючи повний функціонал створення та проходження тестів. Модуль створення тестів (Form1) дозволяє гнучко керувати питаннями, а модуль проходження (Form2) забезпечує інтерактивність із таймером і прогрес-барами. Використання Windows Forms, .NET 8 і JSON дозволило створити стабільний і зручний додаток. У наступних підрозділах буде розглянуто тестування програми та аналіз її ефективності.

Висновки до розділу 2

Розділ 2 присвячений проектуванню та розробці програми для автоматизації підготовки до іспитів. Обрані алгоритми для створення, редагування, видалення та проходження тестів характеризуються високою ефективністю: більшість операцій мають часову складність O(1), а обробка файлів — O(n), що є прийнятним для невеликих обсягів даних.

Спроєктовані класи — TestQuestion як модель даних, Form1 для створення тестів і Form2 для їх проходження — забезпечують чіткий розподіл функціоналу та модульність. Програмна реалізація на основі Windows Forms і .NET 8 успішно інтегрує графічний інтерфейс, таймери, прогрес-бари та роботу з JSON-файлами, відповідаючи всім поставленим вимогам. Використання UTF-8 і System.Text.Json гарантує коректну обробку кирилиці та стабільність роботи з даними.

Розроблена програма є зручним інструментом для створення та проходження тестів, готова до тестування та подальшого вдосконалення.

РОЗДІЛ 3. КЕРІВНИЦТВО КОРИСТУВАННЯ ПРОГРАМОЮ

3.1. Мінімальні вимоги до системи

Для коректної роботи програми для автоматизації підготовки до іспитів, розробленої на основі Windows Forms і .NET 8, необхідно, щоб комп'ютер користувача відповідав певним технічним характеристикам. У цьому підрозділі описано мінімальні системні вимоги, які забезпечують стабільну роботу програми, а також рекомендовані параметри для оптимальної продуктивності.

Розроблена програма є легким додатком із низьким споживанням ресурсів, що робить її доступною для використання на більшості сучасних комп'ютерів під керуванням операційної системи Windows. Оскільки програма залежить від платформи .NET 8, її сумісність обмежується операційними системами, які підтримують цю версію фреймворку.

Операційна система є ключовим фактором для запуску програми. Програма сумісна з Windows 10 (версія 1809 або новіша) та Windows 11. Старіші версії Windows (наприклад, Windows 7 або 8.1) не підтримуються .NET 8, тому їх використання неможливе без встановлення додаткових оновлень або альтернативних фреймворків, що не входить у рамки цього проєкту.

Процесор із тактовою частотою від 1 ГГц забезпечить базову продуктивність для виконання програми. Оскільки додаток не виконує складних обчислень, однопотокової продуктивності сучасних процесорів достатньо для швидкої обробки операцій створення та проходження тестів.

Оперативна пам'ять об'ємом 2 ГБ є мінімальною вимогою для запуску програми разом із операційною системою. Програма використовує незначну кількість пам'яті для зберігання списку питань (List<TestQuestion>) і графічного інтерфейсу, тому навіть на системах із мінімальними ресурсами вона працюватиме стабільно.

Вільне місце на диску має становити щонайменше 200 МБ. Цей обсяг включає встановлення .NET 8 Runtime (приблизно 150 МБ) і місце для самої

програми та її файлів (близько 50 МБ). JSON-файли з тестами займають мінімальний обсяг (кілька кілобайт на тест), тому вимоги до дискового простору залишаються низькими.

Відеокарта не має специфічних вимог, оскільки програма використовує стандартний графічний інтерфейс Windows Forms, який підтримується інтегрованими графічними адаптерами. Роздільна здатність екрана від 1024x768 пікселів є достатньою для комфортного відображення всіх елементів інтерфейсу.

Додаткове програмне забезпечення включає .NET 8 Runtime, який необхідно встановити, якщо він відсутній на комп'ютері користувача. Цей компонент доступний для завантаження з офіційного сайту Microsoft і є обов'язковим для запуску програми.

Для оптимальної роботи програми рекомендується використовувати Windows 11 або Windows 10 (версія 22H2), що забезпечує повну сумісність із .NET 8 і сучасними оновленнями безпеки. Процесор із частотою 2 ГГц або вище (наприклад, Intel Core i3 або еквівалент) гарантує швидке завантаження програми та плавну роботу інтерфейсу. Оперативна пам'ять об'ємом 4 ГБ і більше дозволяє комфортно працювати з кількома програмами одночасно, уникаючи затримок. Вільне місце на диску від 500 МБ забезпечить достатній простір для зберігання кількох тестів і оновлень .NET Runtime.

Програма не потребує підключення до інтернету, оскільки всі дані зберігаються локально у JSON-файлах. Вона не залежить від зовнішніх бібліотек, окрім стандартних компонентів .NET 8, що спрощує її встановлення та використання. Відсутність кросплатформності є свідомим вибором, оскільки цільова аудиторія (студенти та викладачі) переважно використовує Windows як основну платформу.

3.2. Інтерфейс користувача

У цьому підрозділі описано інтерфейс користувача програми для автоматизації підготовки до іспитів, розробленої на основі Windows Forms і .NET 8. Програма складається з двох основних модулів: створення тестів (Form1) і проходження тестів (Form2). Кожен модуль має власний графічний інтерфейс, спроєктований для зручності використання студентами, викладачами чи іншими користувачами. Інтерфейс є інтуїтивно зрозумілим, із чітким розподілом функціональних елементів, що забезпечує легкість у створенні тестів і проходженні їх із контролем часу та прогресу.

Інтерфейс модуля створення тестів (Form1)

Модуль створення тестів реалізовано у вигляді вікна Windows Forms із назвою "Створення тестів". Інтерфейс поділено на кілька логічних зон для введення даних, перегляду списку питань і керування тестами.

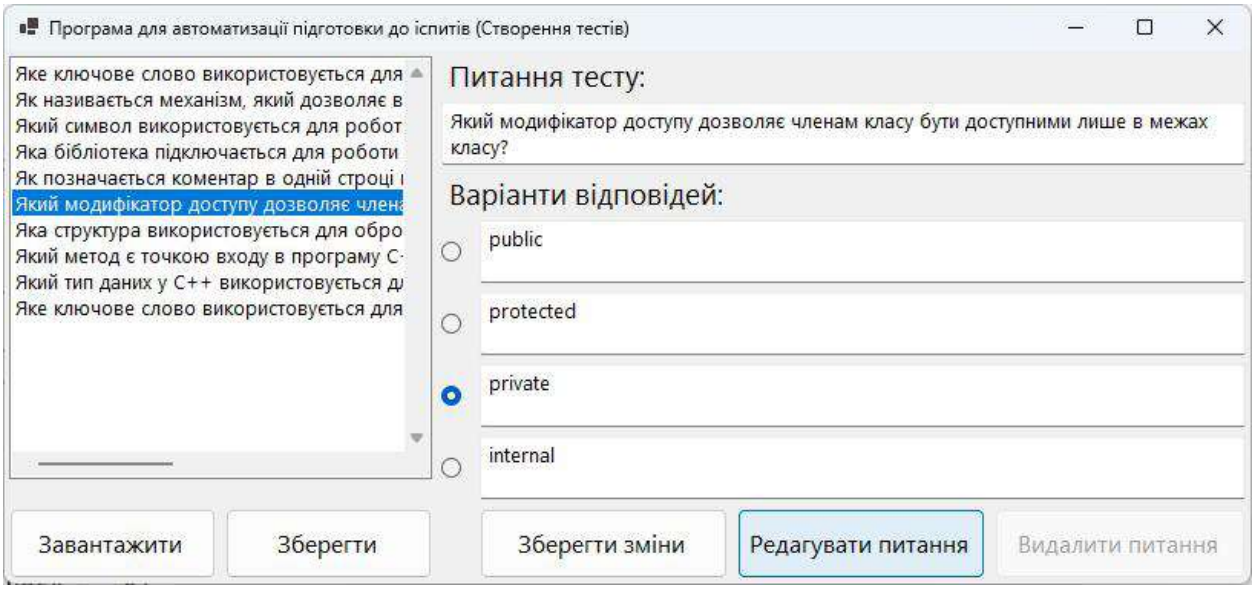

Рисунок 3.2.1 – Інтерфейс модуля створення тестів

У верхній частині вікна розташовано текстове поле txtQuestion для введення тексту питання. Це поле займає ширину приблизно 80% вікна і дозволяє вводити питання довжиною до 255 символів. Під ним розміщено чотири текстові поля txtOption1, txtOption2, txtOption3 і txtOption4, кожне з яких призначене для введення одного варіанту відповіді. Ці поля розташовані вертикально зліва, поруч із відповідними радіокнопками rbCorrect1,

rbCorrect2, rbCorrect3 і rbCorrect4, які дозволяють обрати правильну відповідь. Радіокнопки згруповано так, що одночасно може бути обрана лише одна з них, що відповідає логіці тесту з єдиною правильною відповіддю.

Ліворуч від полів введення розміщено список lstQuestions у вигляді ListBox, який відображає додані питання за їх текстом. Список має вертикальну прокрутку і займає приблизно половину висоти вікна, дозволяючи переглядати всі питання тесту. Користувач може виділити питання в списку для редагування чи видалення.

У нижній частині вікна розташовано п’ять кнопок: btnAddQuestion, btnSaveTest, btnLoadTest, btnDeleteQuestion і btnEditQuestion. Кнопка btnAddQuestion має текст "Додати питання" (змінюється на "Зберегти зміни" у режимі редагування) і розміщена ліворуч. Вона додає нове питання до списку або зберігає зміни в існуючому. Кнопка btnSaveTest із текстом "Зберегти тест" відкриває діалогове вікно для збереження тесту у файл JSON. Кнопка btnLoadTest ("Завантажити тест") відкриває діалогове вікно для вибору файлу тесту. Кнопка btnDeleteQuestion ("Видалити питання") видаляє виділене питання зі списку, а btnEditQuestion ("Редагувати питання") заповнює поля для редагування вибраного питання. Усі кнопки мають однаковий розмір і розташовані горизонтально для зручного доступу.

Інтерфейс модуля проходження тестів (Form2)

Модуль проходження тестів реалізовано у вікні з назвою "Проходження тестів". Інтерфейс спроєктовано для інтерактивного проходження тестів із відображенням часу та прогресу.

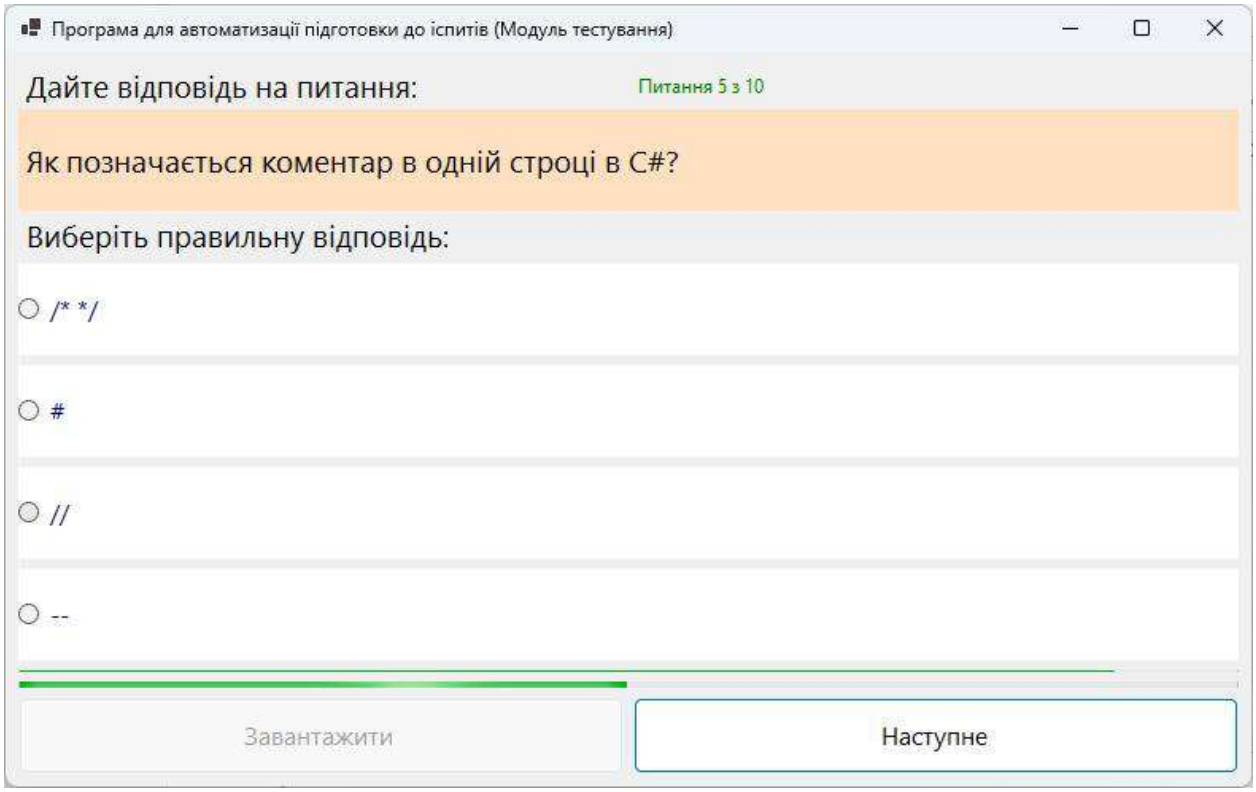

Рисунок 3.2.2 – Інтерфейс модуля проходження тестів

У верхній частині вікна розміщено мітку lblQuestion, яка відображає текст поточного питання. Мітка займає ширину вікна і підтримує перенесення тексту, щоб довгі питання відображалися повністю. Під міткою розташовано чотири радіокнопки rbOption1, rbOption2, rbOption3 і rbOption4, розміщені вертикально. Кожна радіокнопка показує один варіант відповіді, і користувач може обрати лише одну з них. Радіокнопки згруповано для забезпечення взаємовиключного вибору.

Нижче радіокнопок розміщено прогрес-бар progressBarTime, який відображає залишок часу на поточне питання (1 хвилина). Він має ширину 80% вікна і змінює значення від 60 до 0, синхронізуючись із таймером questionTimer. Поруч із ним розташована мітка lblProgress, яка показує текст у форматі "Питання X з Y", де X — номер поточного питання, а Y — загальна кількість питань.

Під цими елементами розміщено другий прогрес-бар progressBarTest, який відображає загальний прогрес проходження тесту у відсотках (0–100%). Він має таку ж ширину, як progressBarTime, і оновлюється після кожного

питання. Прогрес-бари візуально відокремлені один від одного для чіткості сприйняття.

У нижній частині вікна розташовано дві кнопки: btnLoadTest і btnNext. Кнопка btnLoadTest із текстом "Завантажити тест" розміщена ліворуч і відкриває діалогове вікно для вибору JSON-файлу з тестом. Кнопка btnNext ("Наступне") розміщена праворуч і активується після завантаження тесту. Її текст змінюється на "Завершити" для останнього питання. Вона дозволяє перейти до наступного питання або завершити тест.

Загальні особливості інтерфейсу

Обидва модулі мають мінімалістичний дизайн із чітким розташуванням елементів. Використання стандартних компонентів Windows Forms (TextBox, RadioButton, ListBox, Button, Label, ProgressBar) забезпечує знайомий вигляд для користувачів Windows. Текст у полях і мітках підтримує кирилицю завдяки кодуванню UTF-8. Розмір вікон за замовчуванням становить 800x600 пікселів, але користувач може змінювати його, а елементи інтерфейсу адаптуються завдяки властивостям закріплення (Anchor).

Інтерфейс Form1 зосереджений на введенні та управлінні даними, тоді як Form2 — на інтерактивності та візуальному контролі часу й прогресу. Повідомлення у вигляді MessageBox інформують користувача про помилки (наприклад, відсутність вибору відповіді) або успішне виконання дій (збереження, завантаження).

3.3. Інструкція для роботи з програмою

У цьому підрозділі наведено детальну інструкцію для роботи з програмою для автоматизації підготовки до іспитів. Програма складається з двох модулів: створення тестів (Form1) і проходження тестів (Form2). Інструкція описує кроки для запуску програми, створення тестів, їх збереження, завантаження та проходження, а також пояснює, як інтерпретувати результати. Інтерфейс є інтуїтивно зрозумілим, але дотримання інструкції забезпечить максимальну ефективність використання.

Для початку роботи з програмою необхідно переконатися, що на комп’ютері встановлено .NET 8 Runtime (доступний для завантаження з офіційного сайту Microsoft). Програма постачається у вигляді двох виконуваних файлів: TestCreator.exe для створення тестів і TestTaker.exe для їх проходження. Щоб запустити потрібний модуль, достатньо двічі клацнути на відповідному файлі у провіднику Windows. Вікно кожного модуля відкриється з початковим інтерфейсом, описаним у підрозділі 3.2.

Робота з модулем створення тестів (TestCreator)

Створення нового тесту

Після запуску TestCreator.exe відкриється вікно "Створення тестів".

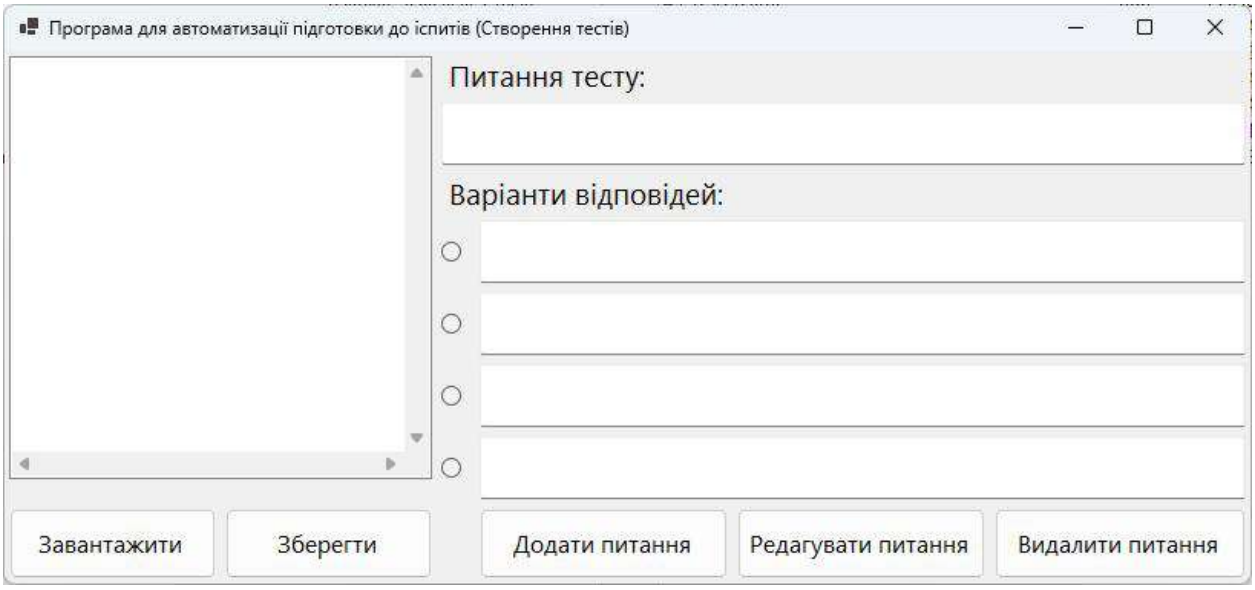

Рисунок 3.3.1 – Інтерфейс модуля створення тестів

Щоб створити нове питання у текстове поле у верхній частині введіть текст питання (наприклад, "Яка столиця України?"). У чотири поля нижче введіть варіанти відповідей (наприклад, "Київ", "Львів", "Одеса", "Харків").

Оберіть правильну відповідь, поставивши позначку на одній із радіокнопок праворуч від варіантів (наприклад, для "Київ" оберіть першу радіокнопку).

Натисніть кнопку "Додати питання". Питання з’явиться у списку праворуч.

Повторіть ці кроки для додавання потрібної кількості питань. Список питань відображає текст кожного доданого питання.

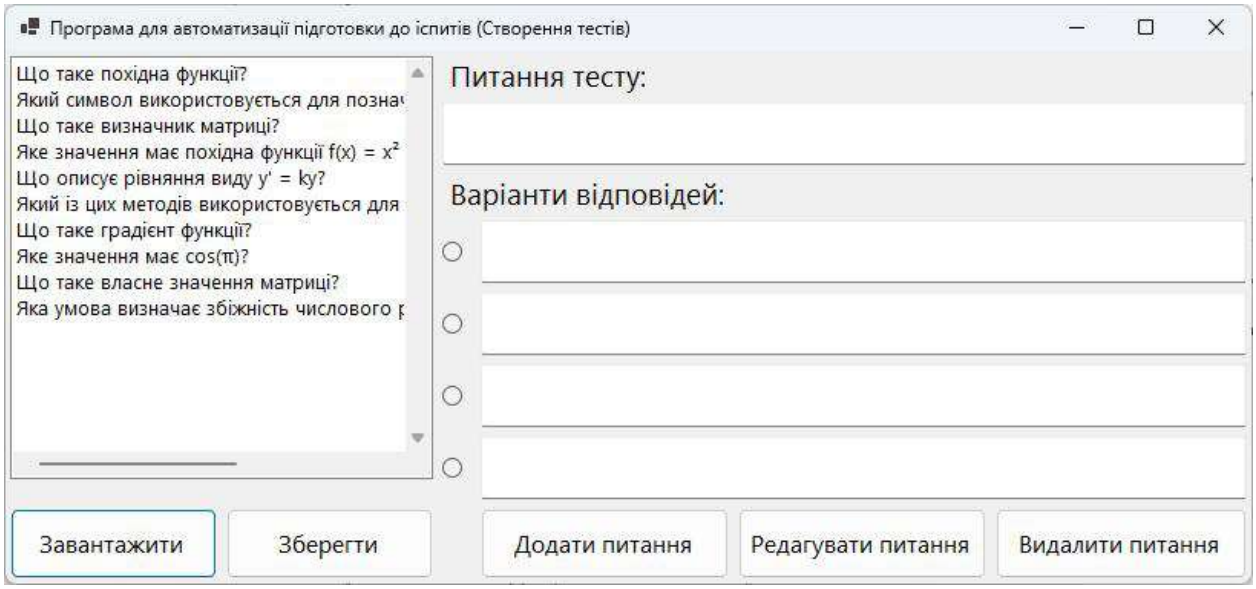

Рисунок 3.3.2 – Додано 10 питань

Щоб відредагувати існуюче питання потрібно виділіть питання у списку праворуч, клацнувши на нього один раз. Натисніть кнопку "Редагувати питання". Поля введення заповняться даними цього питання.

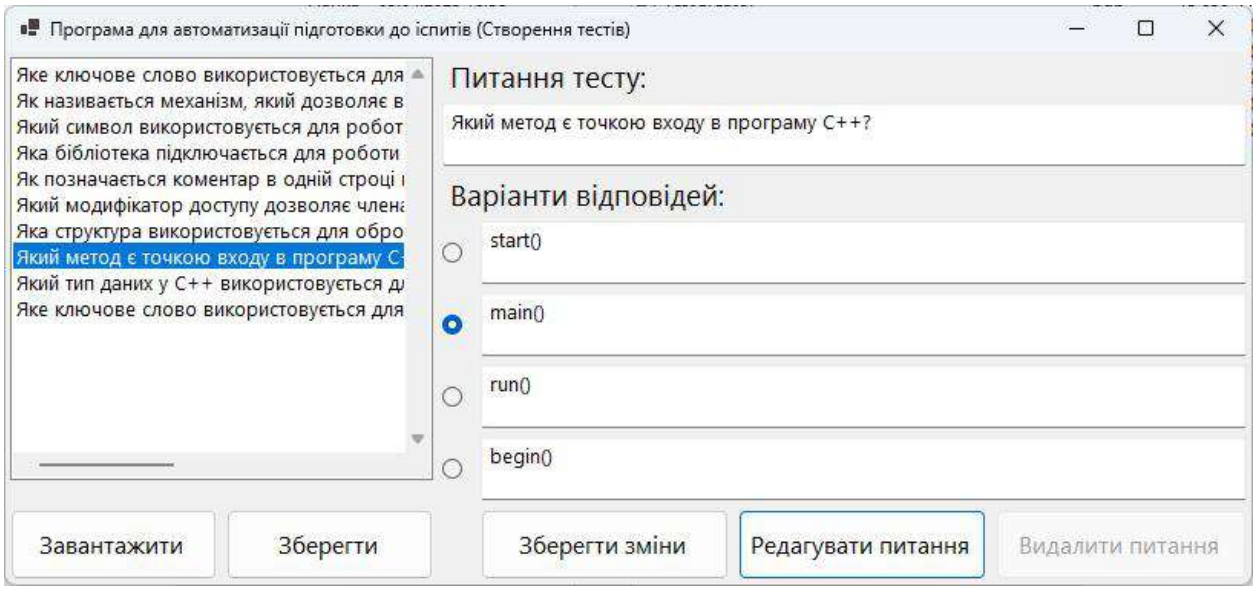

Рисунок 3.3.3 – Редагування питання

Внесіть зміни до тексту питання, варіантів відповідей чи правильної відповіді. Натисніть кнопку "Зберегти зміни" (раніше "Додати питання"). Оновлене питання замінить попередню версію у списку.

Для видалення питання потрібно виділити питання у списку. Натисніть кнопку "Видалити питання". Питання зникне зі списку, а поля введення очистяться.

Після створення всіх питань збережіть тест натиснувши кнопку "Зберегти тест". У діалоговому вікні виберіть місце збереження, введіть назву файлу (наприклад, "MathTest.json") і натисніть "Зберегти". З’явиться повідомлення про успішне збереження із зазначенням шляху до файлу.

Завантаження тесту

Щоб відкрити раніше створений тест для редагування натискайте кнопку "Завантажити тест". У діалоговому вікні виберіть файл JSON із тестом і натисніть "Відкрити". Список питань оновиться вмістом файлу.

Робота з модулем проходження тестів (TestTaker)

Завантаження тесту

Для початку проходження тесту запустіть TestTaker.exe.

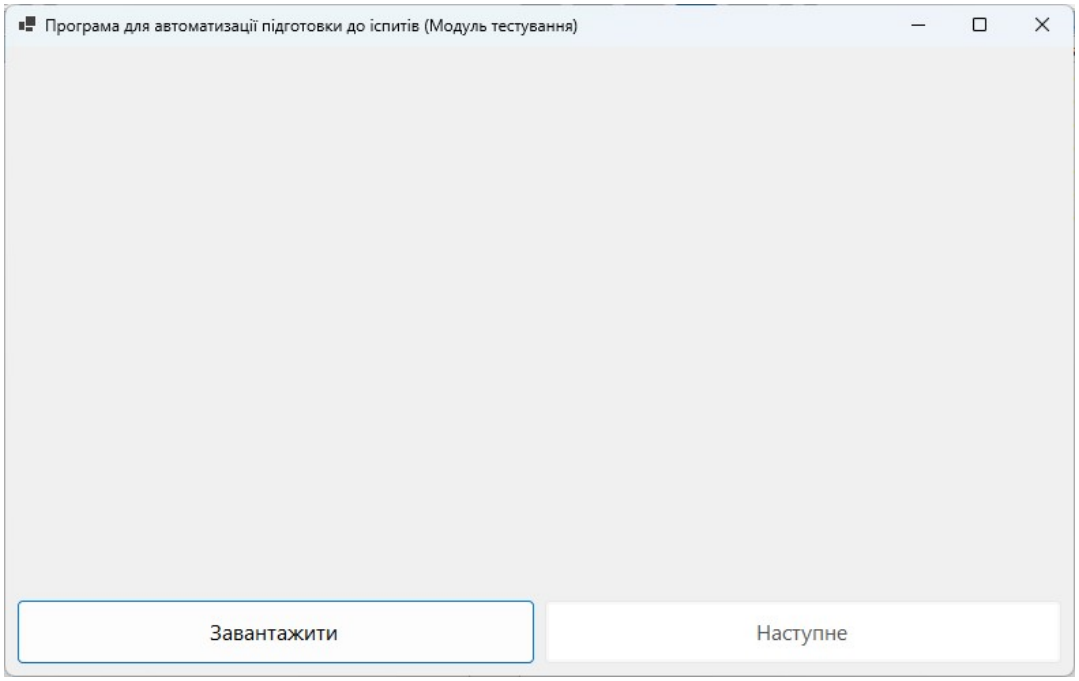

Рисунок 3.3.4 – Інтерфейс модуля проходження тестів

У вікні "Проходження тестів" натисніть кнопку "Завантажити тест". У діалоговому вікні виберіть файл JSON із тестом (наприклад, "MathTest.json") і натисніть "Відкрити". Якщо файл коректний, з’явиться перше питання, а кнопка "Наступне" стане активною.

Після завантаження тесту прочитайте питання у верхній частині вікна (наприклад, "Яка столиця України?"). Ознайомтеся з варіантами відповідей на радіокнопках (наприклад, "Київ", "Львів", "Одеса", "Харків").

Оберіть один варіант, клацнувши на відповідну радіокнопку.

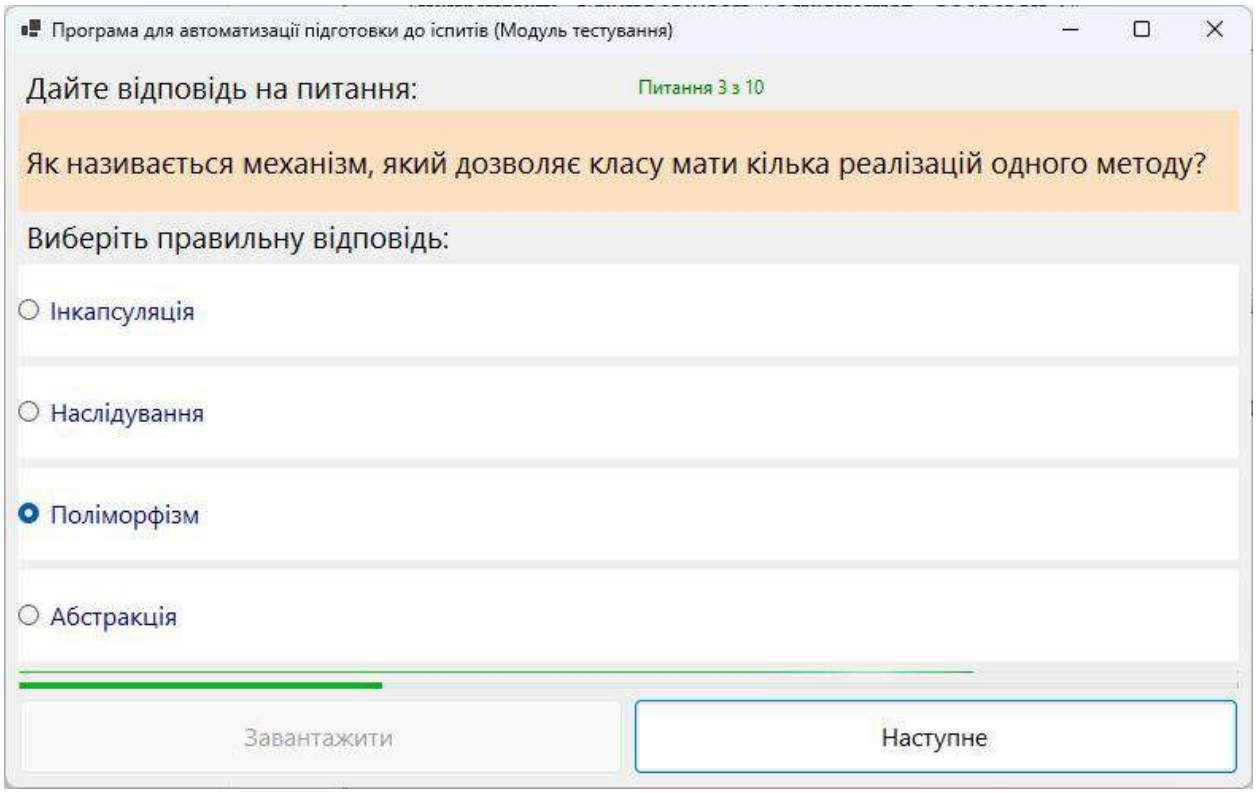

Рисунок 3.3.5 – Проходження тесту

Слідкуйте за прогрес-баром часу (зверху), який показує залишок 1 хвилини на питання. Якщо час закінчиться, програма автоматично перейде до наступного питання, а поточна відповідь не врахується.

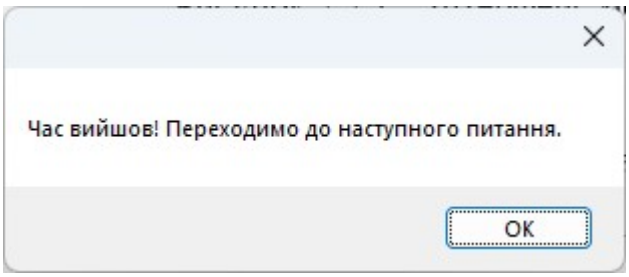

Рисунок 3.3.6 – Час обдування завершився

Натисніть кнопку "Наступне", щоб перейти до наступного питання. Прогрес-бар тесту (нижче) оновиться, показуючи відсоток виконаних питань.

Повторюйте кроки для кожного питання. На останньому питанні кнопка "Наступне" зміниться на "Завершити". Після натискання "Завершити" або завершення всіх питань з’явиться вікно з результатами.

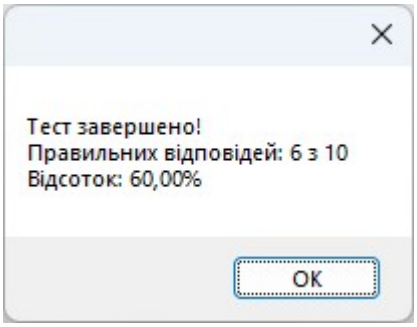

Рисунок 3.3.7 – Вікно з результатами

Після перегляду результатів вікно тесту повернеться до початкового стану, дозволяючи завантажити новий тест.

Переконайтеся, що вводите коректні дані: питання та всі чотири варіанти відповідей мають бути заповнені, а правильна відповідь обрана перед додаванням чи збереженням. Уникайте використання спеціальних символів, які можуть некоректно відображатися у JSON без додаткових налаштувань. Зберігайте тести в доступному місці, щоб легко знайти їх для проходження. Якщо виникає помилка (наприклад, "Помилка при завантаженні тесту"), перевірте цілісність JSON-файлу та його відповідність формату (питання, 4 варіанти, індекс правильної відповіді).

Висновки до розділу 3

Розділ 3 містить керівництво користувача програми для автоматизації підготовки до іспитів, яке охоплює системні вимоги, опис інтерфейсу та інструкцію з використання.

Мінімальні системні вимоги (Windows 10, 1 ГГц процесор, 2 ГБ ОЗП, 200 МБ на диску з .NET 8 Runtime) забезпечують доступність програми для

більшості сучасних комп’ютерів, а рекомендовані параметри підвищують комфортність роботи.

Інтерфейс модулів створення (Form1) і проходження тестів (Form2) є інтуїтивно зрозумілим, із чітким розташуванням текстових полів, радіокнопок, списків, кнопок і прогрес-барів, що полегшує взаємодію користувача з програмою.

Інструкція детально описує процес створення, редагування, збереження тестів, а також їх проходження з контролем часу та переглядом результатів, роблячи програму зручною для студентів і викладачів. Таким чином, розділ підтверджує готовність програми до практичного використання та її відповідність цілям підготовки до іспитів.

ВИСНОВКИ

Дипломна робота присвячена розробці програми для автоматизації підготовки до іспитів із використанням технології Windows Forms на платформі .NET 8. У процесі роботи було досягнуто поставленої мети — створено зручний і функціональний інструмент, який поєднує два основних модулі: створення тестів і їх проходження. Програма дозволяє користувачам формувати тести з питаннями, варіантами відповідей і правильними відповідями, зберігати їх у форматі JSON, а також проходити тести з обмеженням часу на кожне питання (1 хвилина) і відображенням прогресу через прогрес-бари.

У першому розділі проведено аналіз задачі розробки, розглянуто аналоги (Quizlet, Kahoot!, Moodle, Microsoft Forms), що підтвердило унікальність програми завдяки офлайн-доступності та специфічним функціям, таким як таймер на питання. Обґрунтовано вибір Windows Forms і .NET 8 як оптимальних технологій за критеріями простоти, продуктивності та сумісності з Windows, що відповідає потребам цільової аудиторії — студентів і викладачів.

Другий розділ охопив проектування та реалізацію програми. Обрані алгоритми для обробки даних (додавання, редагування, видалення питань, перевірка відповідей) мають високу ефективність із часовою складністю O(1) для більшості операцій і O(n) для роботи з файлами. Спроєктовані класи TestQuestion, Form1 і Form2 забезпечили модульність і чіткий розподіл функцій. Програмна реалізація підтвердила працездатність модулів із підтримкою кирилиці, локального зберігання даних і інтерактивного інтерфейсу.

Третій розділ описав практичне використання програми. Мінімальні системні вимоги (Windows 10, 2 ГБ ОЗП, 200 МБ на диску) роблять її доступною для широкого кола користувачів, а інтерфейс із текстовими полями, радіокнопками, списками та прогрес-барами є зручним і зрозумілим.

Інструкція детально пояснює процес створення, збереження та проходження тестів, що сприяє ефективному застосуванню програми в навчанні.

Розроблена програма успішно виконує поставлені завдання: систематизує підготовку до іспитів, економить час і підвищує зручність самоперевірки знань. Вона може бути корисною для студентів, учнів і викладачів як інструмент для самостійного навчання або контролю знань. У майбутньому програму можна вдосконалити, додавши кросплатформність через .NET MAUI, підтримку інших типів питань (наприклад, відкритих відповідей) або інтеграцію із системами управління навчанням.

Дипломна робота продемонструвала успішну реалізацію програмного продукту, який відповідає сучасним вимогам до автоматизації навчального процесу, і заклала основу для його подальшого розвитку.

ПЕРЕЛІК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Глушков В. М. Основи безпаперової інформатики. Київ: Наукова думка, 1982. 552 с.
2. Дейт К. Дж. Вступ до систем баз даних. 8-е вид. Київ: Вільямс, 2005. 1328 с.
3. Кнут Д. Е. Мистецтво програмування. Т. 1. Основні алгоритми. 3-тє вид. Київ: Вільямс, 2007. 832 с.
4. Кормен Т. Г., Лейзерсон Ч. І., Рівест Р. Л., Стайн К. Алгоритми: побудова й аналіз. 3-тє вид. Харків: Клуб сімейного дозвілля, 2019. 1296 с.
5. Таненбаум Е. Сучасні операційні системи. 4-е вид. Київ: Вільямс, 2018. 1120 с.
6. Фаулер М. Рефакторинг: удосконалення існуючого коду. 2-е вид. Київ: Наш Формат, 2020. 448 с.
7. Саттер Г. Винятковий C++: 47 задач із програмування, стилю та проектування. Київ: DiaSoft, 2003. 240 с.
8. Скієна С. Алгоритми: настільна книга розробника. 2-е вид. Київ: Фабула, 2021. 720 с.
9. Макконнелл С. Досконалий код. 2-е вид. Київ: Наш Формат, 2018. 896 с.
10. Шилдт Г. C#: повне керівництво. 4-е вид. Київ: Вільямс, 2019. 1056 с.
11. Ріхтер Дж. CLR via C#. Програмування на платформі .NET Framework 4.5 мовою C#. 4-е вид. Київ: DiaSoft, 2014. 896 с.
12. Альбахарі Дж., Альбахарі Б. C# 8.0 у скороченому викладі. Київ: Вільямс, 2020. 1088 с.
13. Троелсен Е., Джепікс Ф. Мова програмування C# 8 і платформа .NET Core 3. Київ: DiaSoft, 2021. 1344 с.
14. Петзольд Ч. Програмування для Windows Forms. Київ: Вільямс, 2007. 832 с.
15. Ліберті Дж. Вивчаємо C#. 5-е вид. Київ: Наш Формат, 2022. 928 с.
16. Буч Г., Максимчук Р., Енгл М. Об’єктно-орієнтований аналіз і проектування з прикладами додатків. 3-тє вид. Київ: Вільямс, 2010. 624 с.

17. Гамма Е., Хелм Р., Джонсон Р., Влісідес Дж. Прийоми об’єктно-орієнтованого проєктування. Патерни проєктування. Київ: Наш Формат, 2019. 432 с.
18. Соммервілл І. Інженерія програмного забезпечення. 10-е вид. Харків: Клуб сімейного дозвілля, 2018. 1088 с.
19. Пресман Р. Інженерія програмного забезпечення: практичний підхід. 7-е вид. Київ: Вільямс, 2012. 928 с.
20. Microsoft .NET 8 Documentation [Електронний ресурс] / Microsoft Corporation. — Режим доступу: https://docs.microsoft.com/en-us/dotnet/core/whats-new/dotnet-8. — Дата звернення: 11.05.2025.

ДОДАТКИ

ДОДАТОК А.

Програмний код модулів

```
namespace TestCreator
{
   partial class Form1
   {
      /// <summary>
      ///  Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      ///  Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }

      #region Windows Form Designer generated code

      /// <summary>
      ///  Required method for Designer support - do not modify
      ///  the contents of this method with the code editor.
      /// </summary>
      private void InitializeComponent()
      {
         txtQuestion = new TextBox();
         txtOption1 = new TextBox();
         txtOption2 = new TextBox();
         txtOption3 = new TextBox();
         txtOption4 = new TextBox();
         rbCorrect1 = new RadioButton();
         rbCorrect2 = new RadioButton();
         rbCorrect3 = new RadioButton();
         rbCorrect4 = new RadioButton();
         tableLayoutPanel1 = new TableLayoutPanel();
         label2 = new Label();
         lstQuestions = new ListBox();
         label1 = new Label();
         btnSaveTest = new Button();
         btnAddQuestion = new Button();
         btnDeleteQuestion = new Button();
         btnEditQuestion = new Button();
```