НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
(КПІ ім. Ігоря Сікорського)

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ
кафедра БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

«**До захисту допущено**»

В.о. завідувач кафедри БМК

_____ **Євген НАСТЕНКО**

«__»_____2023 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою
«Комп'ютерні технології в біології та медицині»
спеціальності *122 «Комп'ютерні науки»*

на тему: **Програмний застосунок для організації навчання з надання першої долікарської допомоги**

Виконав: студент **IV** курсу, групи _БС-92_

ФІРТИЧ БОГДАН ВАДИМОВИЧ _____
(прізвище, ім'я, по батькові) (підпис)

Керівник:
професор каф. БМК, д.т.н, професор Тачиніна Олена Миколаївна _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Консультант *доц. каф. охорони праці, промислової та цивільної безпеки (ОППЦБ), доц., к.т.н. Глива Валентин Анатолійович* _____

Рецензент: *доцент кафедри біомедичної інженерії, к.т.н. Рудніцька Олена Володимирівна* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент ___ _____
(підпис)

Київ – 2023 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет біомедичної інженерії

Кафедра біомедичної кібернетики

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні технології в біології та медицині»

ЗАТВЕРДЖУЮ

В.о. завідувач кафедри БМК

_____ Євген НАСТЕНКО

«__»__________2023 р.

ЗАВДАННЯ
на дипломну роботу студенту

ФІРТИЧ БОГДАН ВАДИМОВИЧ

(прізвище, ім'я, по батькові)

1. Тема роботи **Програмний застосунок для організації навчання з надання першої долікарської допомоги**

Керівник роботи

Тачиніна Олена Миколаївна, д.т.н, професор

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «31» *травня* 2023 р. №***2106-с***

2. Термін подання студентом роботи ***06-08 червня 2023р.***

3. Вихідні дані до роботи: *дослідження документообігу та бізнес-процесів при наданні медичної допомоги пацієнту*

4. Зміст роботи: *анотації (на двох мовах), вступ, огляд літературних джерел, теоретична частина, практична частина, висновки, список використаних джерел*

5. Перелік ілюстративного матеріалу: 14 слайдів

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата завдання	
		видав	прийняв
Дипломної роботи	Глива Валентин Анатолійович, д.т.н., професор	30.05.2023	09.06.2023

7. Дата видачі завдання ***30 травня 2023 року.***

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримати завдання за темою ДР на практику	До 15.02.2023р.	*виконано*
2	Переддипломна практика	За графіком	*виконано*
3	Виконання розділів ДР (Вступ, аналітичний огляд літературних джерел, теоретична частина)	До кінця практики	*виконано*
4	Виконання розділів ДР (практична частина, загальні висновки, список джерел)	Не пізніше 1 тижня до засідання каф-ри	*виконано*
5	Перевірка ДР науковим керівником	Не пізніше 1 тижня до засідання каф-ри	*виконано*
6	Подання в електронному вигляді ДР та анотації до неї на перевірку нормоконтролера та плагіат (UNICHECK).	---- « ------	*виконано*
7	Надання документів на засідання кафедри	За день до засідання	*виконано*
8	Предзахист ДР та допуск до захисту дисертації	Згідно плану каф.	*виконано*
9	Подання ДР рецензенту. Отримання рецензії.	До подання пакету документів до ЕК	*виконано*
10	Подання пакету документів по ДР та супровідних до неї документів до захисту в ЕК[1]	За 5 днів до дати захисту ДР за графіком	*виконано*
11	Захист ДР в ЕК		

Студент _____ **Богдан ФІРТИЧ**
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник ДР _____ **Олена ТАЧИНІНА**
(підпис) (ім'я, ПРІЗВИЩЕ)

Нормоконтролер _____ **Галина КОРНІЄНКО**
(підпис) (ім'я, ПРІЗВИЩЕ)

[1] не пізніше ніж за 5 днів до затвердженої дати захисту ДР в ЕК

Анотація

Дипломна робота за темою «Програмний застосунок для організації навчання з надання першої долікарської допомоги» виконана студентом кафедри біомедичної кібернетики ФБМІ *Фіртичем Богданом Вадимовичем зі спеціальності 122 «Комп'ютерні науки» за освітньо-професійною програмою «Комп'ютерні технології в біології та медицині» та* складається зі: вступу; 3 розділів (*аналітичний огляд літературних джерел, теоретична частина, практична частина)*, висновків до кожного з цих розділів; загальних висновків; списку використаних джерел, який налічує 25 джерел. Загальний обсяг роботи 67 сторінок.

Актуальність теми дипломної роботи відображає важливість розвитку інноваційних технологій та підвищення рівня знань населення в сфері надання першої допомоги.

За останні роки спостерігається зростання кількості нещасних випадків та аварій, що вимагають негайного надання допомоги потерпілим. Згідно зі статистичними даними, у 2021 році загальна кількість нещасних випадків на виробництві у світі становила близько 340 мільйонів, з яких 2,78 мільйонів призвели до смерті. Також у 2021 році було зафіксовано більше 1,35 мільйона смертей внаслідок дорожньо-транспортних пригод.

Наслідками таких ситуацій часто стають травми, судоми, розриви судин та інші стани, що вимагають негайного втручання. Часто перші хвилини після нещасного випадку вирішально впливають на розуміння ймовірності порятунку життя та здоров'я потерпілого. Відсутність базових навичок надання першої допомоги може призвести до трагічних наслідків.

Враховуючи актуальність проблеми, розробка програмного застосунку для організації навчання з надання першої долікарської допомоги дозволить забезпечити доступне та ефективне навчання широких верств населення. Така система сприятиме підвищенню рівня загальної культури здоров'я та зменшенню кількості смертей внаслідок неправильного надання першої допомоги.

Мета і завдання роботи.

Метою роботи є підвищення інформованості населення у сфері правил надання долікарської допомоги, а, також, спрощення процесу здобуття знань у цій області.

Її досягнення передбачає вирішення наступних *завдань*:

1. Аналіз наукових джерел на тему надання першої домедичної допомоги;
2. Розробка тестів для перевірки рівня розуміння вивчених правил;
3. Визначення основних вимоги до додатку та вибір середовища розробки;
4. Проектування користувацького інтерфейсу.
5. Реалізація додатку.
6. Обґрунтування прийнятих рішень за темою дипломної роботи на практику.

Використані методи. Мова програмування C#, графічний фреймворк WinForms, СКБД MySQL на сервері Google Cloud SQL, алгоритм SHA для шифрування тестових даних.

Отримані результати. Повноцінний застосунок для навчання правилам надання першої долікарської допомоги.

Публікації. За результатами виконаної роботи публікації не передбачаються.

Ключові слова. фреймворк, реалізація, архітектура, СКБД, додаток, домедична допомога, інтерфейс.

Бібліографічний опис ДР

Фіртич Б.В. Програмний застосунок для організації навчання з надання першої долікарської допомоги : дипломна роб. бакалавра : 122 Комп’ютерні науки / Фіртич Богдан Вадимович. – Київ, 2023. – 67 с.

Abstract

Diploma thesis on the topic "A software application for the organization of training on the provision of first aid " was carried out by a student of the Department of Biomedical Cybernetics, Faculty of Biomedical Engineering, *Firtych Bohdan Vadymovych*, majoring in 122 "Computer Sciences" under the educational and professional program "Computer Technologies in Biology and medicine" and consists of: introduction; 3 sections (analytical review of literary sources, theoretical part, practical part), conclusions to each of these sections; general conclusions; the list of used sources, which includes 25 sources. The total volume of the thesis is 67 pages.

Relevance of the topic reflects the importance of developing innovative technologies and increasing the level of knowledge of the population in the field of first aid.

In recent years, there has been an increase in the number of accidents and accidents that require immediate assistance to the victims. According to statistics, in 2021, the total number of occupational accidents in the world was about 340 million, of which 2.78 million resulted in death. Also, in 2021, more than 1.35 million deaths due to traffic accidents were recorded.

The consequences of such situations are often injuries, convulsions, blood vessel ruptures and other conditions that require immediate intervention. Often, the first minutes after an accident have a decisive influence on the understanding of the probability of saving the victim's life and health. Lack of basic first aid skills can lead to tragic consequences.

Taking into account the urgency of the problem, the development of a software application for the organization of training in the provision of first aid will allow to provide affordable and effective training for broad segments of the population. Such a system will contribute to increasing the level of general health culture and reducing the number of deaths due to incorrect first aid

Purpose and tasks of the thesis.

The purpose of the work is to increase the awareness of the population in the field of rules for providing pre-medical care, as well as to simplify the process of acquiring knowledge in this area.

Its achievement involves solving the following tasks:

Analysis of scientific sources on the topic of providing first aid;

1. Development of tests to check the level of understanding of the studied rules;
2. Determination of the main requirements for the application and selection of the development environment;
3. User interface design.
4. Implementation of the application.
5. Justification of the decisions made on the topic of the thesis for practice.

Used methods. C# programming language, WinForms graphic framework, MySQL database on Google Cloud SQL server, SHA algorithm, used for encryption of test data.
Obtained results. A full-fledged tool for teaching the rules of providing first aid.
Publications. Based on the results of the work performed, publications are not expected.
Keywords. framework, implementation, architecture, DBMS, application, pre-medical assistance, interface.

Bibliographic description of the diploma thesis:

Firtych B.V. [A software application for the organization of training on the provision of first aid]: bachelor's diploma thesis: 122 Computer Science / Firtych Bohdan Vadymovych. – Kyiv, 2023. – 67 pages.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Рис – Рисунок.

Див – Дивитись.

GUI – Graphical User Interface

СК(У)БД – Система Керування(Управління) Базами Даних

MVVM – Model View ViewModel

MVM – Model View Model

MVC – Model View Controller

ВСТУП

Актуальність теми дипломної роботи відображає важливість розвитку інноваційних технологій та підвищення рівня знань населення в сфері надання першої допомоги.

За останні роки спостерігається зростання кількості нещасних випадків та аварій, що вимагають негайного надання допомоги потерпілим. Згідно зі статистичними даними, у 2021 році загальна кількість нещасних випадків на виробництві у світі становила близько 340 мільйонів, з яких 2,78 мільйонів призвели до смерті. Також у 2021 році було зафіксовано більше 1,35 мільйона смертей внаслідок дорожньо-транспортних пригод.

Наслідками таких ситуацій часто стають травми, судоми, розриви судин та інші стани, що вимагають негайного втручання. Часто перші хвилини після нещасного випадку вирішально впливають на розуміння ймовірності порятунку життя та здоров'я потерпілого. Відсутність базових навичок надання першої допомоги може призвести до трагічних наслідків.

Враховуючи актуальність проблеми, розробка програмної системи для організації навчання з надання першої долікарської допомоги дозволить забезпечити доступне та ефективне навчання широких верств населення. Така система сприятиме підвищенню рівня загальної культури здоров'я та зменшенню кількості смертей внаслідок неправильного надання першої допомоги.

Мета і завдання роботи

1)Аналіз наукових джерел на тему надання першої домедичної допомоги;

2) Розробка тестів для перевірки рівня розуміння вивчених правил;

3) Визначення основних вимоги до додатку та вибір середовища розробки;

4) Проектування користувацького інтерфейсу.

5) Реалізація додатку.

Методи дослідження

Мова програмування C#, графічний фреймворк WinForms, СКБД MySQL на сервері Google Cloud SQL, алгоритм SHA для шифрування тестових даних.

Практичне значення одержаних результатів

Розроблений у ході роботи додаток можна використовувати для підвищення рівня обізнаності населення у сфері надання першої допомоги. Застосовувати додаток можна як у спеціалізованих установах/на курсах/тренінгах, так і для персонального навчання.

Апробація результатів роботи

За результатами виконаної роботи апробація не передбачається.

Публікації

За результатами виконаної роботи публікації не передбачаються.

Структура роботи

Загальний обсяг тексту складає 67 сторінок. Структура роботи включає: вступ; три розділи (включаючи висновки до кожного з них), що включають огляд тематичної літератури, розкриття використовуваних матеріалів і методів, а також презентацію результатів та обговорення проведеного дослідження; загальні висновки; список використаних джерел, що містить 25 пунктів (1 – на кирилиці). Робота включає 35 рисунків та 0 таблиць.

РОЗДІЛ 1
АНАЛІТИЧНИЙ ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1.1 Аналіз можливих засобів розробки

1.1.1 Мова програмування

C# є об'єктно-орієнтованою мовою програмування, розробленою компанією Microsoft для платформи .NET. Однією з основних цілей розробки C# було забезпечення високої продуктивності програміста, забезпечуючи простоту та зручність використання [1]. C# поєднує переваги таких мов програмування, як Java та C++, але має додаткові засоби для полегшення роботи розробника, наприклад, вбудовані типи даних та синтаксичний цукор [2]. Версія 7.0 C# включає в себе ряд нових властивостей, які спрямовані на підвищення продуктивності розробника та підтримку сучасних концепцій програмування, таких як шаблони, кортежі та локальні функції [3].

Однією з ключових особливостей C# є підтримка об'єктно-орієнтованого програмування, яке дозволяє розробникам створювати модульні, гнучкі та масштабовані програми. ООП-парадигма забезпечує засоби для створення класів, об'єктів, спадковості, поліморфізму та інкапсуляції, які сприяють підтримці коду та легкій адаптації до змін вимог [4]. Крім того, C# включає підтримку подій, делегатів та лямбда-виразів, що робить роботу з асинхронним програмуванням простішою [1].

Альтернативами для розробки додатків на платформі .NET є мови програмування, такі як Visual Basic .NET (VB.NET) та F#.

VB.NET також є мовою програмування на платформі .NET, яка спрощує синтаксис та робить код більш читабельним. Однак, VB.NET може мати меншу продуктивність та менше можливостей порівняно з C# [2]. Це може призвести до більшої складності коду та меншої гнучкості при розробці додатків.

F# є функціональною мовою програмування на платформі .NET, яка надає високий рівень абстракції та можливість розробки математично обґрунтованих

програм. Однак, порівняно з C#, F# може мати меншу підтримку та менший набір бібліотек, що може ускладнити розробку додатків [3].

1.1.2 Фреймворк для розробки графічного інтерфейсу

WinForms (або Windows Forms) є бібліотекою для створення графічного інтерфейсу користувача (GUI) для Windows-додатків на платформі .NET. WinForms надає засоби для створення вікон, діалогів, кнопок, текстових полів, списків та інших елементів GUI [12]. Бібліотека дозволяє розробникам створювати інтуїтивно зрозумілі додатки для кінцевого користувача, оскільки вона надає великий набір вбудованих компонентів інтерфейсу та забезпечує можливість розширення за допомогою сторонніх компонентів [12]. Microsoft Documentation for WinForms містить докладні відомості про використання бібліотеки для створення GUI додатків [7].

Однією з переваг WinForms є швидкість розробки, так як вона надає візуальний дизайнер форм, що дозволяє розробникам легко створювати інтерфейси за допомогою перетягування елементів управління на форму [12]. Також WinForms має багато прикладів коду, наявних онлайн, що сприяє швидкому вивченню та вирішенню проблем.

Альтернативами для WinForms є Windows Presentation Foundation (WPF) та Universal Windows Platform (UWP).

WPF є модернізованою бібліотекою для створення GUI, яка надає потужніші засоби для створення налаштовуваних інтерфейсів з використанням XAML [14]. Однак, WPF може мати вищі вимоги до ресурсів системи та складніший процес розробки порівняно з WinForms [12].

UWP є універсальною платформою для розробки додатків, які працюють на всіх пристроях Windows 10, включаючи настільні комп'ютери, планшети та смартфони [10]. UWP дозволяє створювати інтуїтивні та адаптивні інтерфейси, але обмежує доступ до деяких можливостей системи та може мати вищі вимоги до ресурсів порівняно з WinForms [12].

1.1.3 Система керування базами даних

MySQL є однією з найпопулярніших систем керування базами даних (СКБД) і має декілька аналогів, з якими можна порівняти його функціональність та переваги. Порівняємо його з доступними альтернативами та розглянемо переваги та недоліки:

Microsoft SQL Server: MySQL і Microsoft SQL Server є двома з найпопулярніших СКБД. Обидва вони мають розширену функціональність, але вони відрізняються в деяких аспектах. MySQL частіше використовується у веб-розробці, особливо в поєднанні зі стеком LAMP (Linux, Apache, MySQL, PHP/Python/Perl), тоді як Microsoft SQL Server частіше використовується в корпоративному середовищі, особливо з Microsoft .NET-технологіями [12].

Oracle Database: Oracle Database є однією з найпоширеніших СКБД у підприємництві і має багато спеціалізованих функцій та можливостей для масштабування. Однак, в порівнянні з Oracle Database, MySQL зазвичай є більш доступним і простим у використанні. MySQL широко використовується у стартапах, веб-проектах та середовищах з великим обсягом даних [13].

PostgreSQL: Як і MySQL, PostgreSQL є безкоштовною та відкритою СКБД, але вони відрізняються за своїм підходом до дизайну та функціональності. PostgreSQL вважається більш розширеною та масштабованою системою, з великою кількістю додаткових функцій, таких як повнотекстовий пошук, географічні запити, підтримка JSON і багато іншого. Однак, MySQL має простіший та більш інтуїтивний синтаксис, що робить його більш популярним для початківців і веб-розробників [14].

Переваги MySQL порівняно з аналогами:

- Простота використання: MySQL має досить простий синтаксис запитів і зрозумілу структуру бази даних, що робить його легким у використанні, особливо для початківців [5].
- Широке застосування: MySQL широко використовується в веб-розробці і має сильну підтримку у великому співтоваристві розробників. Він

інтегрується з багатьма веб-фреймворками та інструментами, що спрощує розробку і підтримку веб-додатків [12].

- Висока продуктивність: MySQL є швидким і ефективним у роботі з великими обсягами даних. Він має оптимізований двигун бази даних, який дозволяє виконувати запити швидко і ефективно [7].
- Розширені можливості: MySQL підтримує широкий спектр функцій, включаючи реплікацію, кластеризацію, транзакції, розподілені бази даних та багато іншого. Ці можливості дозволяють створювати складні системи забезпечення надійності та масштабованості [8].

1.1.4 Підсумок аналізу можливих засобів розробки

На основі аналізу джерел та аналогів, можна зробити висновок, що використання C#, WinForms та MySQL для створення програмної системи для організації навчання з надання першої долікарської допомоги є виправданим. Ці технології дозволять розробити ефективний, зручний та доступний продукт, який зможе задовольнити потреби користувачів та конкурувати з існуючими аналогами на ринку.

1.2 Аналіз існуючих аналогів

Розглянемо декілька популярних програмних систем, які фокусуються на навчанні першої долікарської допомоги. Серед них:

- First Aid for the USMLE Step 1 є навчальним посібником, який спеціально розроблений для студентів медичних вишів, які готуються до складання екзамену USMLE Step 1. Цей ресурс включає детальні описи процедур надання першої допомоги та медичні алгоритми, але не містить інтерактивних елементів та може бути складним для осіб без медичної освіти.[16]
- First Aid Training by British Red Cross є навчальним додатком, який включає відео, анімації та інтерактивні тести для навчання користувачів першої допомоги. Додаток містить розділи про

реанімацію, задуху, серцеві напади та інші невідкладні ситуації. Однак, цей додаток фокусується на мобільних пристроях та може мати обмеження у функціональності на настільних комп'ютерах. [17]

- CPR & First Aid Certification Course by American Health Care Academy є онлайн-курсом, який пропонує сертифікацію з першої допомоги та серцево-легеневої реанімації. Курс включає відео, текстові матеріали та тести для навчання та перевірки знань користувачів. Проте, цей курс може вимагати оплати та не має вбудованої системи бази даних для відстеження прогресу користувачів. [18]
- First Aid by Australian Red Cross є ще одним мобільним додатком, який навчає користувачів першої допомоги. Додаток містить інформацію про поширені ситуації, коли потрібна перша допомога, та демонструє, як надавати допомогу крок за кроком. Однак, також як і First Aid Training by British Red Cross, цей додаток може мати обмеження у функціональності на настільних комп'ютерах. [19]

Висновок до розділу 1

Проведений аналіз популярних програмних систем для навчання з надання першої долікарської допомоги дозволив виявити ключові особливості та обмеження цих продуктів. Більшість з них фокусуються на мобільних пристроях, що може створювати обмеження у функціональності на настільних комп'ютерах. Також, деякі з розглянутих продуктів можуть бути складними для осіб без медичної освіти або вимагати оплати за навчання та сертифікацію.

Враховуючи зазначені особливості, можна припустити, що існує потреба в розробці програмної системи, яка буде спрямована на широку аудиторію, проста у використанні, адаптована для настільних комп'ютерів та безкоштовна. Такий продукт повинен поєднувати навчальні матеріали, інтерактивні елементи та вбудовану систему бази даних для відстеження прогресу користувачів.

РОЗДІЛ 2
ТЕОРЕТИЧНА ЧАСТИНА

У даній частині дипломної роботи розглядаються основні аспекти розробки додатків для платформи Windows. В рамках цього дослідження аналізуються переваги та недоліки розробки додатків на Windows, розглядаються можливі архітектури додатків з графічним інтерфейсом, надається огляд середовища розробки Microsoft Visual Studio та розглядаються різні варіанти роботи з базами даних, включаючи ORM (Object-Relational Mapping) та звичайні запити.

2.1 Переваги та недоліки розробки додатків на Windows

Почнемо з переваг розробки додатків на платформі Windows. Широке розповсюдження платформи Windows серед користувачів створює велику базу потенційних користувачів для додатків, що може бути важливим фактором при комерційній розробці. Крім того, на платформі Windows існує єдине середовище розробки Microsoft Visual Studio, яке надає широкий набір інструментів для розробки додатків на Windows. Також слід відзначити багато можливостей для створення графічного інтерфейсу, таких як WinForms, WPF (Windows Presentation Foundation) та UWP (Universal Windows Platform), що дозволяють розробникам створювати інтуїтивно зрозумілі та гнучкі інтерфейси користувача. Нарешті, важливим аспектом є інтеграція з пакетом Microsoft Office, що спрощує обмін даними з програмами, такими як Word, Excel та PowerPoint.

З іншого боку, розробка додатків на Windows також має свої недоліки. Один з них - обмежений вибір мов програмування порівняно з іншими платформами. Хоча Microsoft Visual Studio підтримує кілька мов програмування, вибір можливостей порівняно з альтернативними платформами може бути обмеженим. Також варто відзначити, що порівняно з деякими альтернативними операційними системами, Windows має нижчий рівень відкритості, що може створювати обмеження доступу до системних ресурсів або функціональності для розробників.

2.2 Архітектури додатків з графічним інтерфейсом

Архітектура Model-View-Controller (MVC) є однією з найпоширеніших та добре відомих архітектур для розробки додатків з графічним інтерфейсом. Вона дозволяє ефективно розділити логіку додатку на три основні компоненти: Модель (Model), Представлення (View) та Контролер (Controller).

1. Модель (Model): Модель відповідає за управління даними та бізнес-логікою додатку. Вона представляє структуру та поведінку даних, забезпечуючи їх доступ та маніпулювання. Модель може включати класи, які взаємодіють з базою даних, веб-сервісами або іншими джерелами даних.
2. Представлення (View): Представлення відповідає за візуалізацію даних та взаємодію з користувачем. Воно відображає дані з моделі користувачеві та приймає його взаємодію. Представлення може бути графічним інтерфейсом, веб-сторінкою або будь-яким іншим способом відображення даних.
3. Контролер (Controller): Контролер відповідає за обробку дій користувача та керування потоком додатку. Він приймає вхідні дані від користувача через представлення, обробляє їх та взаємодіє з моделлю для оновлення даних. Контролер також може оновлювати представлення залежно від змін у моделі.

Один з ключових принципів архітектури MVC (див. рис. 2.1) - це розділення відповідальності між компонентами. Це дозволяє зберігати код чистим, розширювати та тестувати окремі компоненти незалежно один від одного. Крім того, ця архітектура підтримує принцип розширюваності та повторного використання коду.

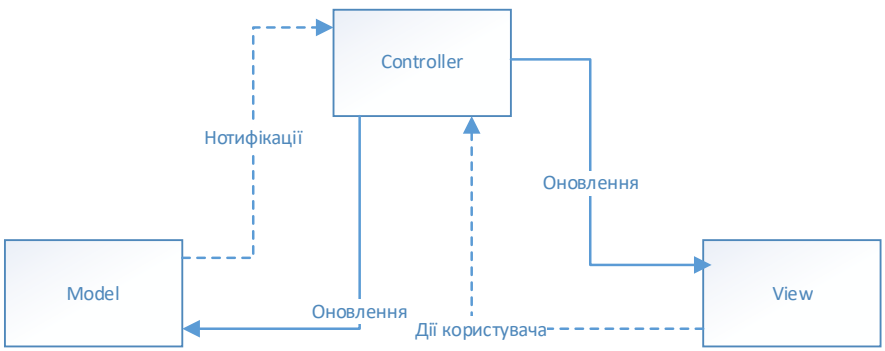

Рисунок 2.1 - Архітектура MVC

Архітектура Model-View-ViewModel (MVVM) є однією з найпоширеніших та ефективних архітектур для розробки додатків з графічним інтерфейсом на платформі Windows. Вона базується на розділенні логіки додатку на три основні компоненти: Модель (Model), Представлення (View) та ViewModel.

1. Модель (Model): Модель відповідає за управління даними та бізнес-логікою додатку. Вона представляє структуру та поведінку даних, забезпечуючи їх доступ та маніпулювання. Модель може включати класи, які взаємодіють з базою даних, веб-сервісами або іншими джерелами даних.
2. Представлення (View): Представлення відповідає за візуалізацію даних та взаємодію з користувачем. Воно відображає дані з ViewModel та приймає користувацькі дії. Представлення може бути графічним інтерфейсом, веб-сторінкою або будь-яким іншим способом відображення даних.
3. ViewModel: ViewModel служить як посередник між Моделлю та Представленням. Вона містить логіку, яка підготовлює дані з Моделі для відображення в Представленні та обробляє взаємодію користувача з додатком. ViewModel також може містити команди, які відповідають на користувацькі дії та взаємодіють з Моделлю для оновлення даних.

Одним з ключових принципів архітектури MVVM (див. рис. 2.2) є розділення логіки від представлення. ViewModel виконує цю роль, надаючи абстракцію між Представленням та Моделлю. Це дозволяє створювати більш гнучкі та вкриті тестами додатки.

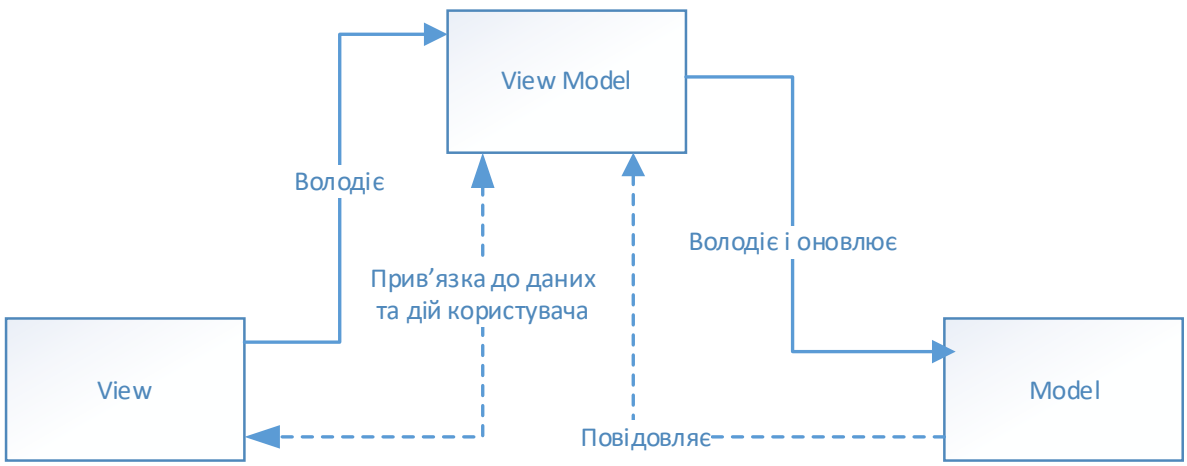

Рисунок 2.2 - Архітектура MVVM

Поряд з архітектурою MVVM, ще одна популярна архітектура для розробки додатків з графічним інтерфейсом на платформі Windows - це архітектура Model-View-Presenter (MVP). Архітектура MVP використовується для розділення логіки додатку від його представлення та взаємодії з користувачем.

У архітектурі MVP присутні три основні компоненти:

1. Модель (Model): Модель відповідає за обробку даних та бізнес-логіку додатку. Вона має незалежний від представлення стан та методи, які дозволяють доступ і маніпуляцію з даними. Модель може включати різні компоненти, такі як класи для роботи з базами даних, веб-сервісами або зовнішніми джерелами даних.
2. Представлення (View): Представлення відображає дані користувачу та обробляє його взаємодію з додатком. Це може бути графічний інтерфейс, вікно, сторінка або будь-який інший компонент, який візуалізує дані користувачеві. Представлення відправляє дії користувача до презентера та отримує від нього дані для відображення.
3. Презентер (Presenter): Презентер виступає як посередник між моделлю та представленням. Він приймає дії користувача від представлення, обробляє їх та взаємодіє з моделлю для отримання необхідних даних або виконання певних дій. Після цього презентер оновлює представлення з отриманими даними або повідомляє про зміни стану.

Один з ключових принципів архітектури MVP (див. рис. 2.3) - це розділення відповідальності між компонентами. Модель відповідає за обробку даних, презентер займається логікою та управлінням, а представлення забезпечує візуалізацію та взаємодію з користувачем. Це дозволяє зберігати код чистим, розширювати та тестувати окремі компоненти незалежно один від одного.

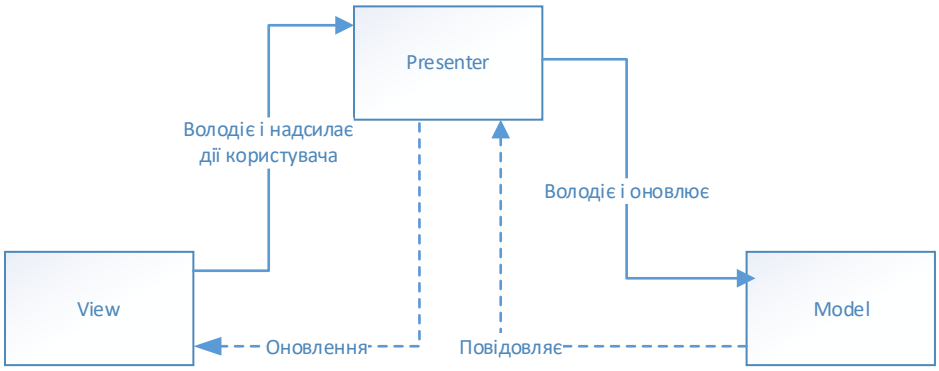

Рисунок 2.3 - Архітектура MVP

2.3 Середовища розробки для платформи Windows

Microsoft Visual Studio: Microsoft Visual Studio є інтегрованим середовищем розробки (IDE), яке надає розробникам широкий спектр інструментів для розробки додатків на платформі Windows. Воно підтримує різні мови програмування, такі як C#, VB.NET, C++ та F#, і надає потужні засоби для редагування коду, відлагодження, компіляції та пакування додатків. Visual Studio також має інтуїтивний інтерфейс та багатофункціональну інтеграцію з іншими інструментами Microsoft, такими як Azure та Office.

JetBrains Rider: JetBrains Rider є іншим популярним середовищем розробки, яке підтримує розробку додатків для платформи Windows. Воно забезпечує повну підтримку мови C# та інших мов, таких як VB.NET, F#, JavaScript та ін. JetBrains Rider має розширені можливості відлагодження, автоматичне завершення коду, систему контролю версій та інші корисні функції для розробників.

Visual Studio Code: Visual Studio Code є легким та швидким редактором коду, який також підтримує розробку додатків для платформи Windows. Він має широкий набір розширень та можливостей для налаштування, що дозволяє розробникам працювати з різними мовами програмування та платформами. Visual Studio Code також інтегрується з Git для контролю версій та має багато інших корисних функцій для розробників.

2.4 Робота з базами даних

При розробці додатків для платформи Windows розробники мають кілька варіантів для роботи з базами даних. Один з них - використання звичайних SQL-запитів, де розробник самостійно пише SQL-запити для отримання, оновлення, видалення або вставки даних у базу даних. Цей підхід дозволяє розробнику більш гнучко управляти запитами та оптимізувати взаємодію з базою даних.

Ще одним варіантом є використання об'єктно-реляційних відображень (ORM) (див. рис. 2.4), таких як Entity Framework. ORM-інструменти забезпечують спрощену роботу з базами даних шляхом відображення таблиць у класи та автоматичного виконання операцій з базою даних на основі об'єктів. Використання ORM спрощує кодування та забезпечує більшу абстракцію взаємодії з базою даних.

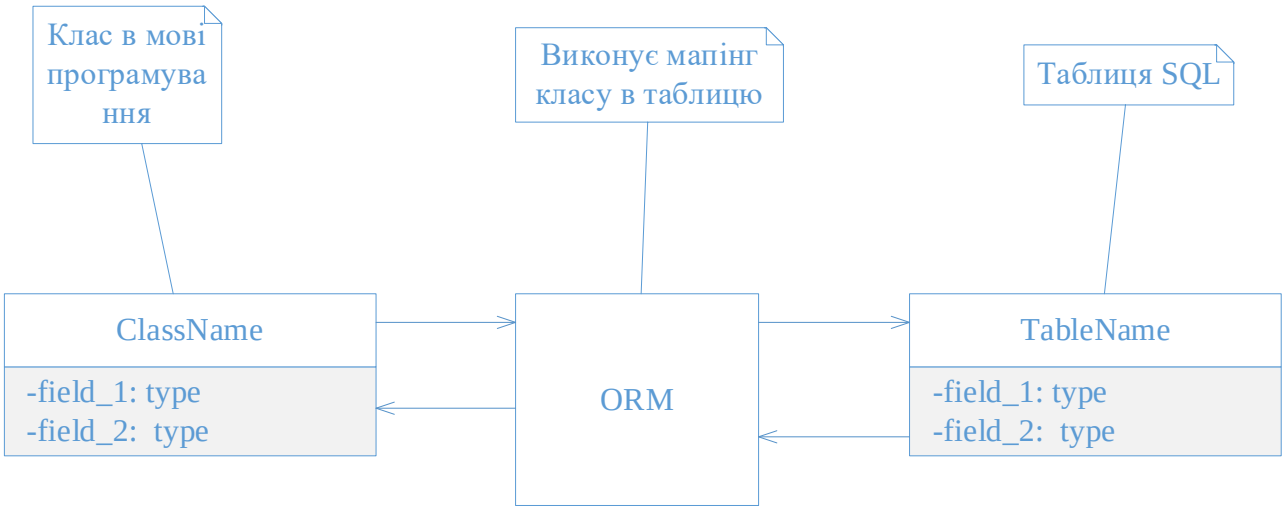

Рисунок 2.4 - Графічне представлення ORM-архітектури

2.5 Засоби тестування додатків

NUnit: NUnit є одним з найпопулярніших фреймворків для тестування додатків на платформі .NET. Він забезпечує можливості для написання, виконання та аналізу автоматизованих тестів. NUnit дозволяє створювати тести з використанням анотацій, асертів та інших методів для перевірки правильності

роботи додатків. Він підтримує різні типи тестів, такі як юніт-тести, інтеграційні тести та тести прийняття.

MSTest: MSTest є офіційним фреймворком для тестування, який постачається з Microsoft Visual Studio. Він надає можливості для написання, виконання та аналізу тестів на платформі .NET. MSTest має інтеграцію з Visual Studio та надає спеціальні атрибути для визначення тестових методів та асертів. Він також підтримує створення тестових наборів та налаштування середовища виконання тестів.

xUnit.net: xUnit.net є ще одним популярним фреймворком для тестування на платформі .NET. Він надає простий та елегантний синтаксис для написання тестів та перевірки результатів. xUnit.net підтримує різні види тестів, включаючи параметризовані тести, тести з використанням фікстур, тести залежності та багато іншого. Він також має гнучкі можливості налаштування та розширення для зручного тестування.

Висновок до розділу 2

В цьому розділі було розглянуто декілька важливих аспектів розробки додатків для платформи Windows. Перш за все, були проаналізовані переваги та недоліки розробки додатків на Windows порівняно з альтернативними платформами. Виявилося, що Windows надає широкий функціонал, потужні інструменти розробки та гарну підтримку програмних продуктів, але може мати обмеження у відкритості та платіжній політиці. Далі, були розглянуті можливі архітектури додатків з графічним інтерфейсом на платформі Windows, такі як модель Model-View-Controller (MVC) та Model-View-ViewModel (MVVM). Ці архітектури дозволяють розділити логіку, представлення та взаємодію з даними в додатках, спрощуючи розширення та підтримку коду. Потім було охарактеризоване середовище розробки Microsoft Visual Studio, яке є потужним інтегрованим середовищем розробки з багатим функціоналом і інструментами для розробки додатків на платформі Windows. Також було згадано про інші популярні

середовища розробки, такі як JetBrains Rider та Visual Studio Code, які також надають зручні інструменти для розробки на платформі Windows.

Крім того, були представлені різні варіанти роботи з базами даних, такі як використання звичайних SQL-запитів та об'єктно-реляційних відображень (ORM). Це дає розробникам можливість ефективно працювати з даними та забезпечувати їх цілісність у додатках.

Загалом, розгорнутий аналіз та огляд теоретичних аспектів розробки додатків для платформи Windows дозволив отримати глибше розуміння процесу розробки, вибір інструментів та архітектур для розробки ефективних та якісних додатків. Враховуючи цю інформацію, розробники можуть приймати обґрунтовані рішення щодо розробки програмного забезпечення для платформи Windows, враховуючи їхні потреби та вимоги.

РОЗДІЛ 3
ПРАКТИЧНА ЧАСТИНА

3.1 Вибір стеку технологій для розробки застосунку

3.1.1 Вибір мови програмування

Для розробки було обрано стек технологій, що включає C#, WinForms, MySQL на базі Google Cloud SQL та архітектуру MVC. Вибір цих конкретних технологій і архітектурного стилю був обумовлений низкою причин.

C# відомий як мова програмування, що допомагає забезпечити високий рівень абстракції, включаючи деталізовані класи та об'єкти. Це спрощує процес програмування, оскільки програмісти можуть фокусуватися на бізнес-логіці програми, не вдаючись до низькорівневих деталей. Одночасно C# забезпечує високу продуктивність та надійність, що робить цю мову вибором для багатьох великих проектів.

Для створення графічного інтерфейсу додатка було вибрано WinForms. Це рішення практично за будь-які стандарти, але є простим і надійним інструментом для створення віконних додатків. WinForms дозволяє легко створювати інтуїтивно зрозумілі інтерфейси, що мають важливе значення для додатка, призначеного для навчання.

3.1.2 Вибір графічного фреймворку

WinForms пропонує значний набір стандартних елементів управління, що зазвичай зустрічаються в додатках Windows, таких як кнопки, текстові поля, чекбокси та ін. Однак, однією з великих переваг WinForms є його гнучкість та можливість розширення, яка дозволяє створювати кастомні елементи управління.

Кастомні елементи управління можуть бути створені шляхом наслідування від існуючих стандартних елементів управління і додавання або зміни функціональності, або шляхом створення цілком нових елементів управління від нуля. Це дозволяє розробникам створювати унікальний, зручний та ефективний інтерфейс користувача, що відповідає конкретним потребам додатка.

Наприклад, в контексті цієї роботи, можна розробити кастомні елементи управління, які покращать якість інтерфейсу, та зроблять його більш приємним для користувача. Наприклад, створення кастомної кнопки зі заокругленими краями та градієнтним фоном при наведенні на неї курсору на WinForms полягає у наслідуванні від класу Button і перевизначенні деяких його методів.

Спочатку ми створюємо новий клас, який наслідує від класу Button. Далі ми відключаємо стандартне малювання кнопки, встановлюючи властивість FlatStyle в значення FlatStyle.Flat і властивість FlatAppearance.BorderSize в 0.

Після цього нам потрібно перевизначити метод OnPaint, який викликається при перерисовці кнопки. У цьому методі ми малюємо заокруглену кнопку за допомогою методів DrawEllipse або DrawArc та DrawLine.

Також ми можемо перевизначити методи OnMouseEnter і OnMouseLeave, щоб змінювати фон кнопки при наведенні на неї курсору. У методі OnMouseEnter ми встановлюємо фон кнопки на градієнт, а у методі OnMouseLeave - повертаємо до стандартного вигляду.

Нарешті, при використанні цього класу в інтерфейсі програми, ми використовуємо наш новий клас кнопки замість стандартного Button.

Таким чином, створюючи власний клас кнопки і перевизначаючи деякі методи, ми можемо додати таку функціональність, як заокруглення країв і градієнтний фон при наведенні курсору.

Таким чином, можливість створення кастомних елементів управління в WinForms є ще одним важливим чинником, що обґрунтовує вибір цієї технології для розробки нашого додатку.

3.1.3 Вибір системи керування базами даних

Щодо вибору бази даних, то в даному випадку це MySQL на базі Google Cloud SQL. MySQL є однією з найбільш популярних систем управління базами даних, яка відрізняється високою продуктивністю та надійністю. Використання Google Cloud SQL дозволяє зосередитися на використанні бази даних, замість її адміністрування. Крім того, забезпечується висока доступність та масштабованість, що може виявитися корисним при великій кількості користувачів.

Використання серверної бази даних, такої як Google Cloud SQL, над локальною базою даних пропонує низку переваг, особливо для програми, що вимагає зберігання та аналізу результатів тестування користувачів.

Перш за все, серверна база даних дозволяє забезпечити централізований доступ до даних. Це означає, що дані можуть бути доступні з будь-якого місця та пристрою, що має підключення до Інтернету. Це особливо важливо для викладачів або інших осіб, що контролюють результати тестування, оскільки вони можуть переглядати ці результати незалежно від свого розташування.

Другою важливою перевагою є масштабованість. Серверні бази даних, особливо ті, що базуються на хмарних сервісах, таких як Google Cloud SQL, легко масштабуються для підтримки збільшення кількості даних. Це означає, що якщо кількість користувачів або обсяг тестових результатів зростає, база даних зможе ефективно обробляти це збільшення без необхідності в значних апгрейдах або переналаштуваннях.

Ще однією перевагою є надійність та безпека. Провайдери хмарних баз даних, такі як Google, забезпечують високий рівень надійності та безпеки, включаючи автоматичне резервне копіювання та відновлення даних, а також захист від зловмисних атак.

3.1.4 Вибір архітектури додатку

Вибір архітектурного стилю Model-View-Controller (MVC) для розробки додатку пропонує ряд переваг, які полегшують розробку, тестування та підтримку системи.

Основним принципом стилю MVC є розділення відповідальності між моделлю даних, виглядом (або представленням даних) та контролером. Це розділення приводить до високої ступені модульності та гнучкості, що є основними перевагами цього підходу.

Модель в MVC відповідає за бізнес-логіку та управління даними. Вона відокремлена від користувацького інтерфейсу, що дозволяє розробникам змінювати бізнес-логіку без зміни представлення. Це може виявитися корисним у

контексті нашої системи, якщо, наприклад, потрібно внести зміни у спосіб обробки результатів тестування, але не хочеться змінювати інтерфейс для користувача.

Представлення у MVC відповідає за відображення даних користувачу. Воно відокремлене від моделі, що дозволяє розробникам вносити зміни у користувацький інтерфейс без впливу на бізнес-логіку. Це може бути корисно, якщо виникає потреба оновити дизайн системи або додати нові візуальні елементи.

Контролер у MVC є посередником між моделлю та представленням. Він обробляє вхідні події (наприклад, кліки миші) і оновлює модель та представлення відповідно. Відокремлення контролера дозволяє розробникам змінювати спосіб взаємодії користувача з системою без зміни бізнес-логіки або відображення даних.

Використання MVC також полегшує тестування, оскільки кожна частина системи (модель, представлення та контролер) може бути тестована окремо.

Отже, архітектурний стиль MVC дозволяє розробляти більш гнучкі, модульні та легко тестовані системи, що робить його хорошим вибором для розробки нашої програмної системи.

Таким чином, ці технології і підходи були обрані для забезпечення надійності, продуктивності, легкості розробки і супроводу додатка.

3.1.5 Вибір середовища розробки

Для розробки додатку було обрано середовище Microsoft Visual Studio 2022. Вибір Visual Studio 2022 було обґрунтовано кількома ключовими аспектами. Перш за все, Visual Studio 2022 є найновішою версією середовища розробки Microsoft і відрізняється від попередніх версій низкою поліпшень. Вона має покращену продуктивність, підтримує 64-бітну архітектуру, що дозволяє використовувати більше ресурсів комп'ютера та оптимізує процес розробки, зокрема для великих проектів.

В порівнянні з іншими середовищами розробки, Visual Studio 2022 має ряд переваг. Вона підтримує багато мов програмування, включаючи C#, і має багатий набір інструментів, що полегшують процес розробки. Зокрема, вона має вбудовані інструменти для роботи з WinForms, що є особливо важливим для цього проекту.

Інші середовища, такі як Eclipse або IntelliJ IDEA, хоча й мають багато функцій, але вони менш зручні для роботи з C# і WinForms. Visual Studio 2022, в свою чергу, вбудована безпосередньо в екосистему Microsoft і ідеально підходить для розробки на базі фреймворку .NET, до якого належить C# і WinForms.

Окрім того, Visual Studio 2022 має вбудований дебагер, що є одним з найпотужніших на ринку. Це допомагає істотно спростити процес знаходження і виправлення помилок в коді.

Також варто відзначити, що Visual Studio 2022 має вбудовану підтримку систем контролю версій, таких як Git, що допомагає забезпечити ефективну командну роботу та контроль за змінами коду.

Використання Visual Studio 2022 дозволяє максимально використати переваги розробки на C# і WinForms, забезпечуючи ефективність і зручність процесу розробки. Таким чином, вибір цього середовища розробки для цього проекту є обґрунтованим і цілком виправданим.

3.2 Діаграми класів додатку

3.2.1 Діаграми класів вікон додатку

На рис. 3.1 представлено діаграму класу базової форми, котра є батьківською для всіх інших вікон додатку. Форма реалізує основу логіку взаємодії з тайтлбаром вікна та іконками на ньому, тож має відповідні поля для її регулювання, а саме поля, що зберігають посилання на попереднє вікно додатку, для надання можливості повернення до нього в один клік, поля іконок тайтлбару та поля поточного статусу вікна.

Також базова форма підписується на обробники подій вікна, котрі є необхідними для імплементації логіки перетягування, правильного закриття, повернення до попередньої сторінки та обробки статусу вікон додатку.

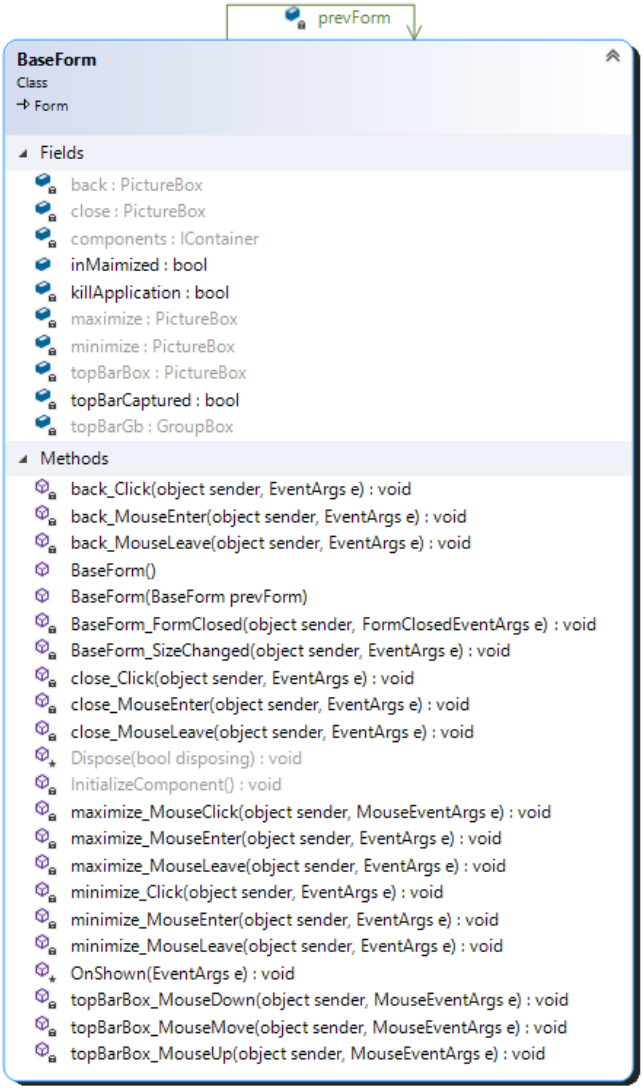

Рисунок 3.1 – Діаграма класу базової форми

На рис. 3.2 представлено діаграму класу форми автентифікації, котра є першою формую при вході в додаток. Клас містить поля наступних типів: DatabaseManager – забезпечує зв’язок з базою даних, необхідний для перевірки наявності облікового запису користувача в базі та додавання нового, у разі запиту такої дії користувачем; PasswordRichTB – замінює символи паролю користувача на ‘*’, запобігаючи випадковому розсекречуванню пароля; RoundedBorderButton – використовується для представлення кнопки «Увійти». Також клас підписується на обробник події натискання на клавішу «Увійти».

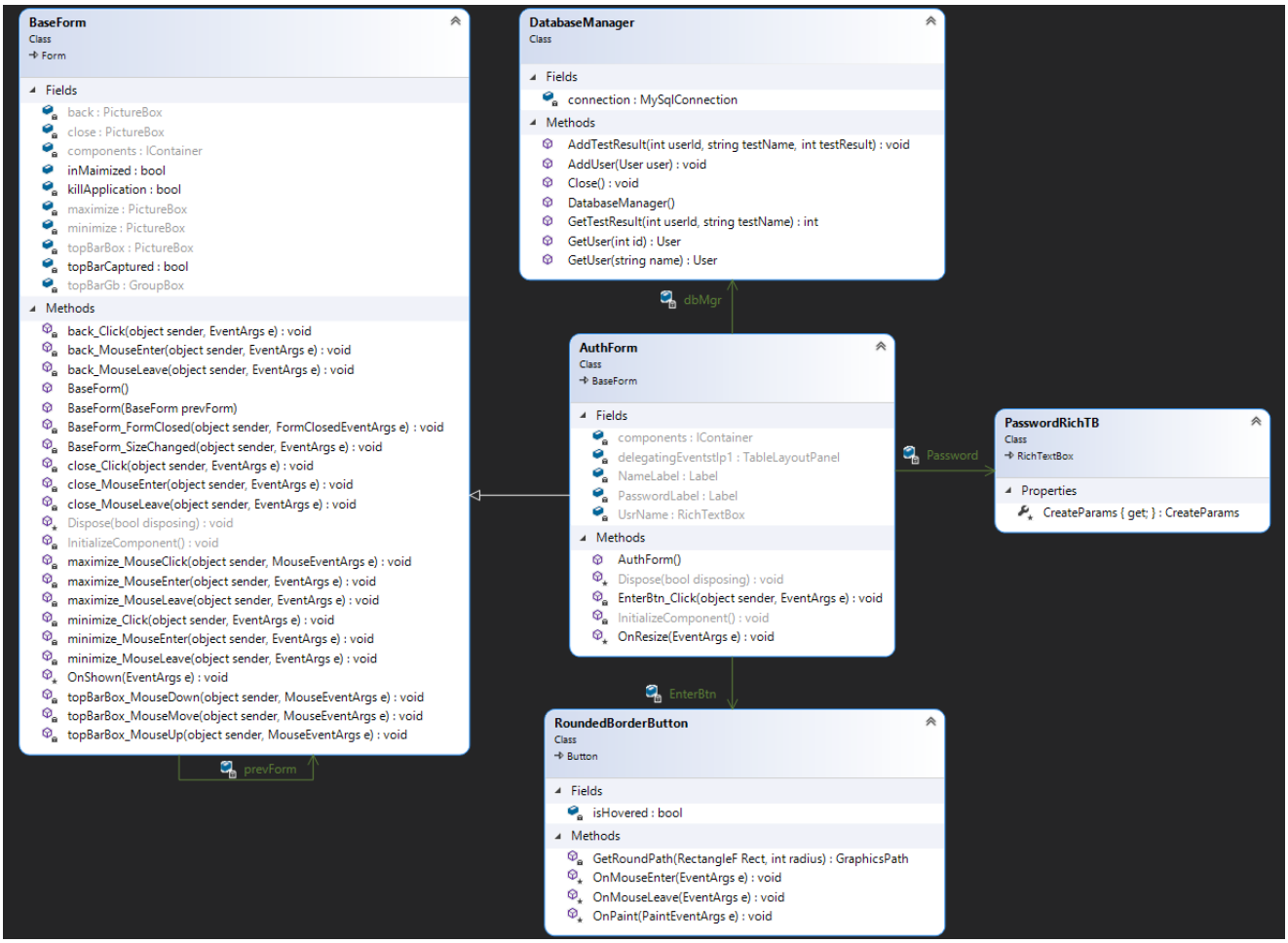

Рисунок 3.2 – Діаграма класу форми автентифікації

На рис. 3.3 представлено діаграму класу форми головного меню, котра відкривається після входу користувача в додаток. Форма містить поля типу RoundedBorderButton, котрі використовуються для представлення кнопок переходу до відповідної навчальної активності, також форма підписує відповідні методи на подію кліку на кнопку.

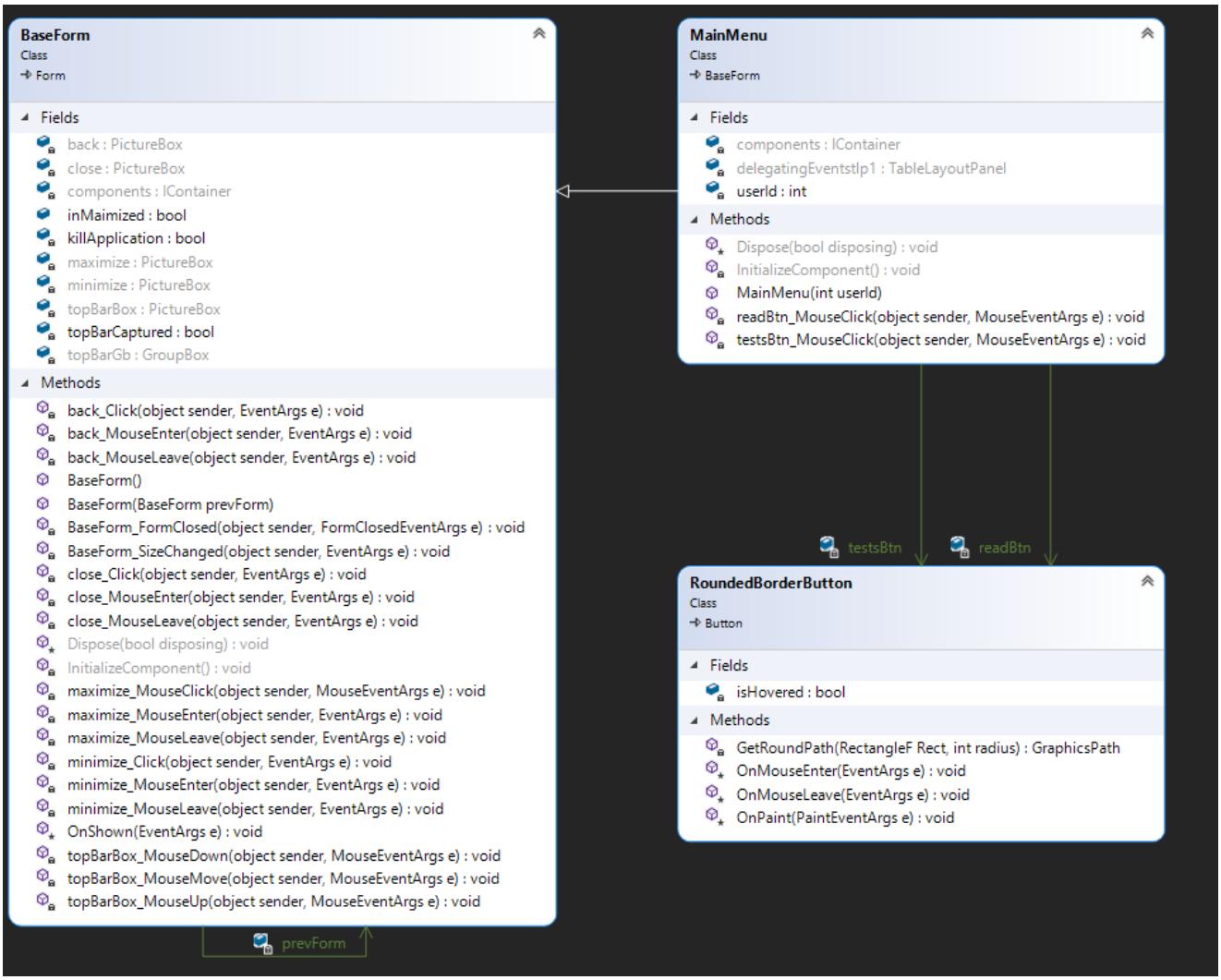

Рисунок 3.3 – Діаграма класу форми головного меню

На рис. 3.4 представлено діаграму класу форми вибору тематичної статті, котра з’являється на екрані після вибору відповідної навчальної активності. Форма містить поля типу RoundedBorderButton, котрі використовуються для представлення кнопок з назвами доступних навчальних статей, також форма підписує відповідні методи на подію кліку на кнопку, що забезпечує можливість перейти до вікна читача.

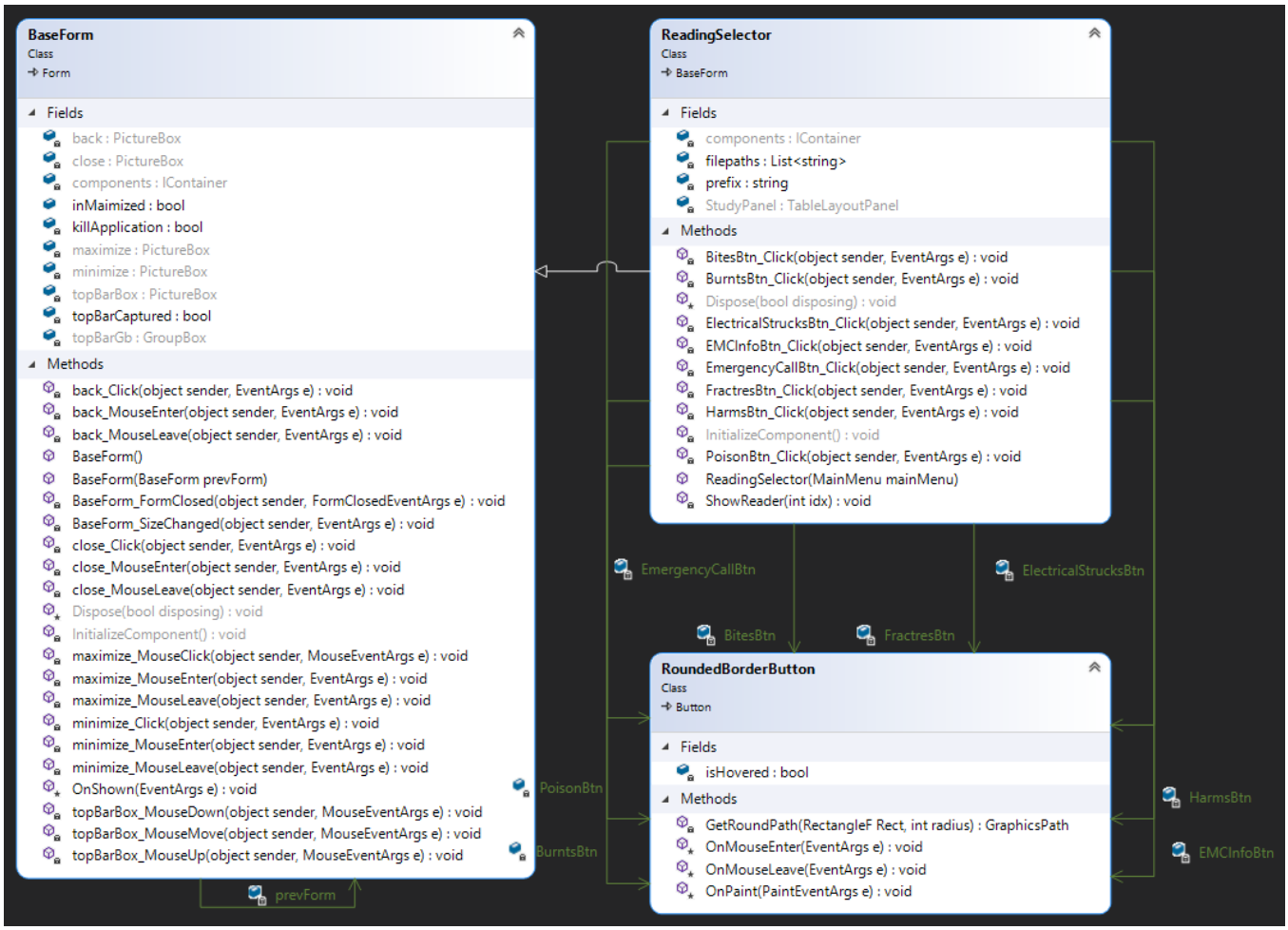

Рисунок 3.4 – Діаграма класу форми вибору навчальної статті

На рис. 3.5 представлено діаграму класу форми читача, котра з’являється після натискання на кнопку статті. Форма містить поля типу RoundedBorderButton, котрі використовуються для представлення кнопок «Попередня сторінка» та «Наступна сторінка», а, також методи обробки кліків на відповідні кнопки, метод, що тригерить завантаження тексту та зображення наступної сторінки і метод зчитування даних з відповідних файлів, та їх розміщення на сторінці.

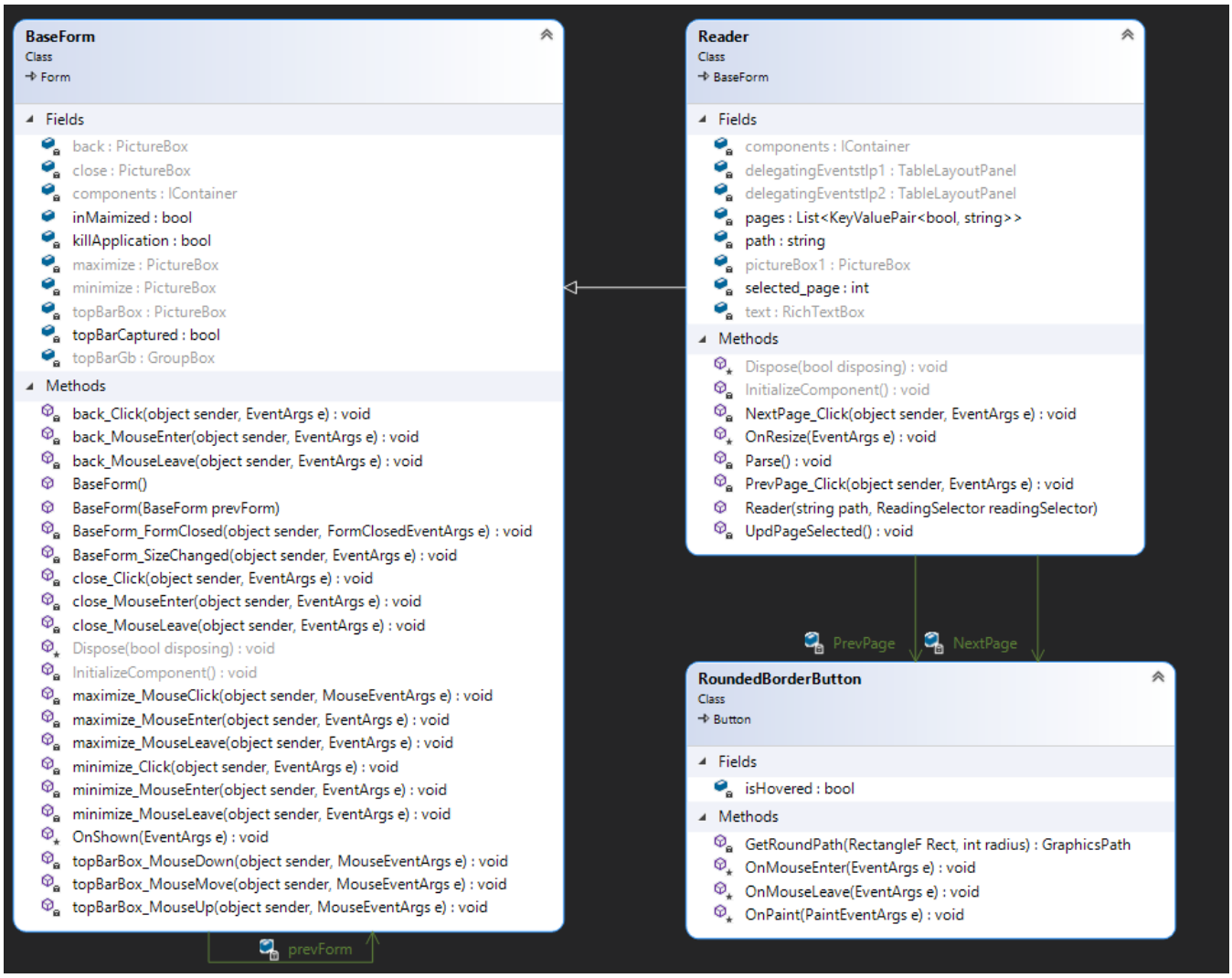

Рисунок 3.5 – Діаграма класу форми читача

На рис. 3.6 представлено діаграму класу форми вибору тематичного тесту, котра з’являється на екрані після вибору відповідної навчальної активності. Форма містить поля типу RoundedBorderButton, котрі використовуються для представлення кнопок з назвами доступних тестів і кнопки переходу до форми перегляду результатів тестування, також форма підписує відповідні методи на подію кліку на кнопку, що забезпечує можливість перейти до вікон тестування та вікна перегляду результатів тестування.

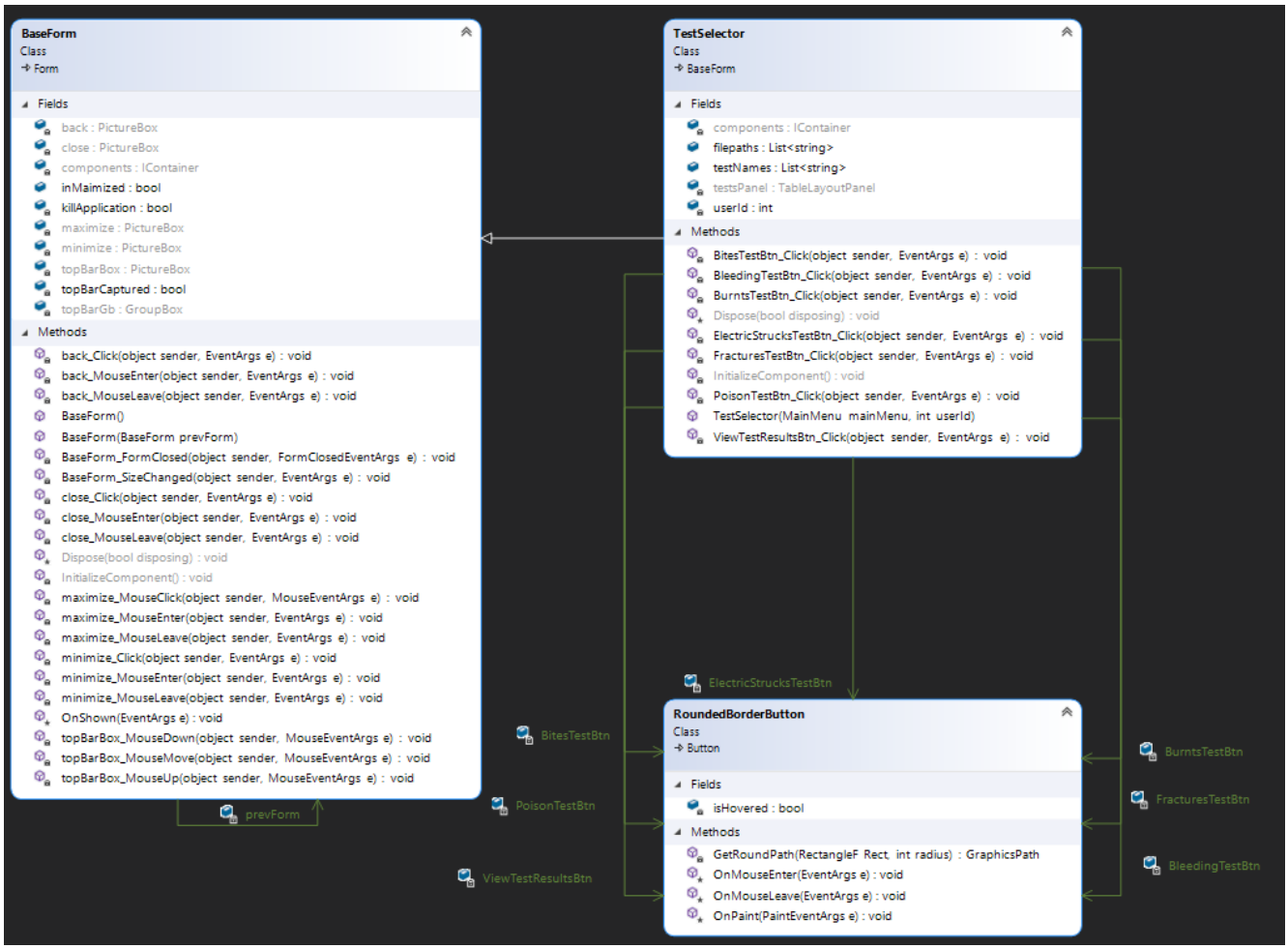

Рисунок 3.6 – Діаграма класу форми вибору тесту

На рис. 3.7 представлено діаграму класу тестової форми, котра з’являється після натискання на кнопку переходу до проходження тесту. Форма містить поле типу RoundedBorderButton, що використовуються для представлення кнопки підтвердження завершення проходження тесту та поле типу DatabaseManager, через яке забезпечується зв’язок з базою даних: зчитування результатів тестування та даних користувачі і запис нового результату проходження тесту.

Також клас містить метод обробки кліку на неї та метод, обробки закриття кнопки, що надає можливість оброблювати випадкове закриття форми та запобігати таким чином втраті результатів тестування.

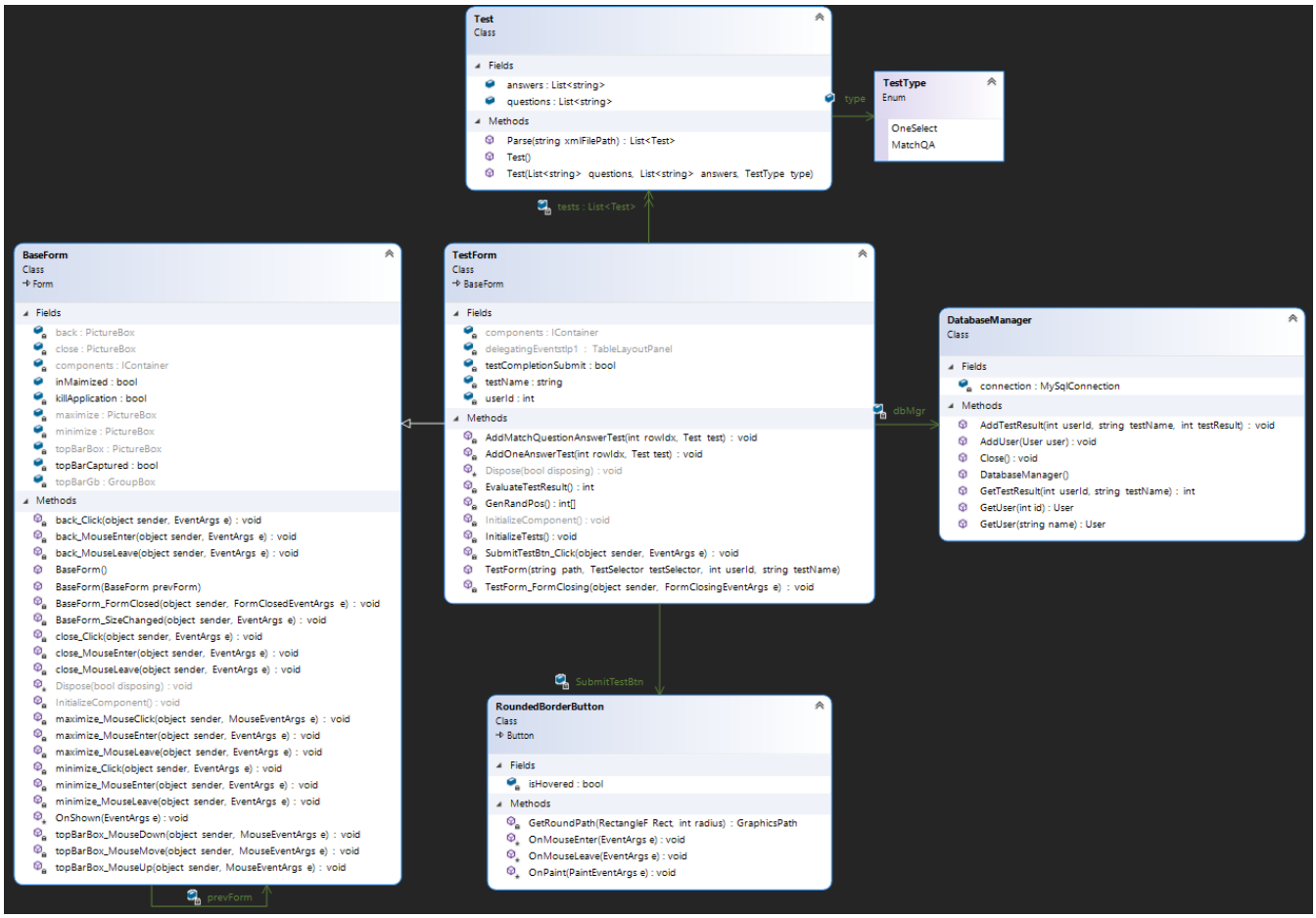

Рисунок 3.7 – Діаграма класу тестової форми

На рис. 3.8 представлено діаграму класу форми перегляду результатів тестування. Форма містить поле типу DatabaseManger, котре забезпечує зв’язок з базою даних, необхідний для зчитування результатів тестування користувача.

Також форма містить методи оновлення результатів тестування та методи забезпечення взаємодії з полем пошуку користувача за іменем.

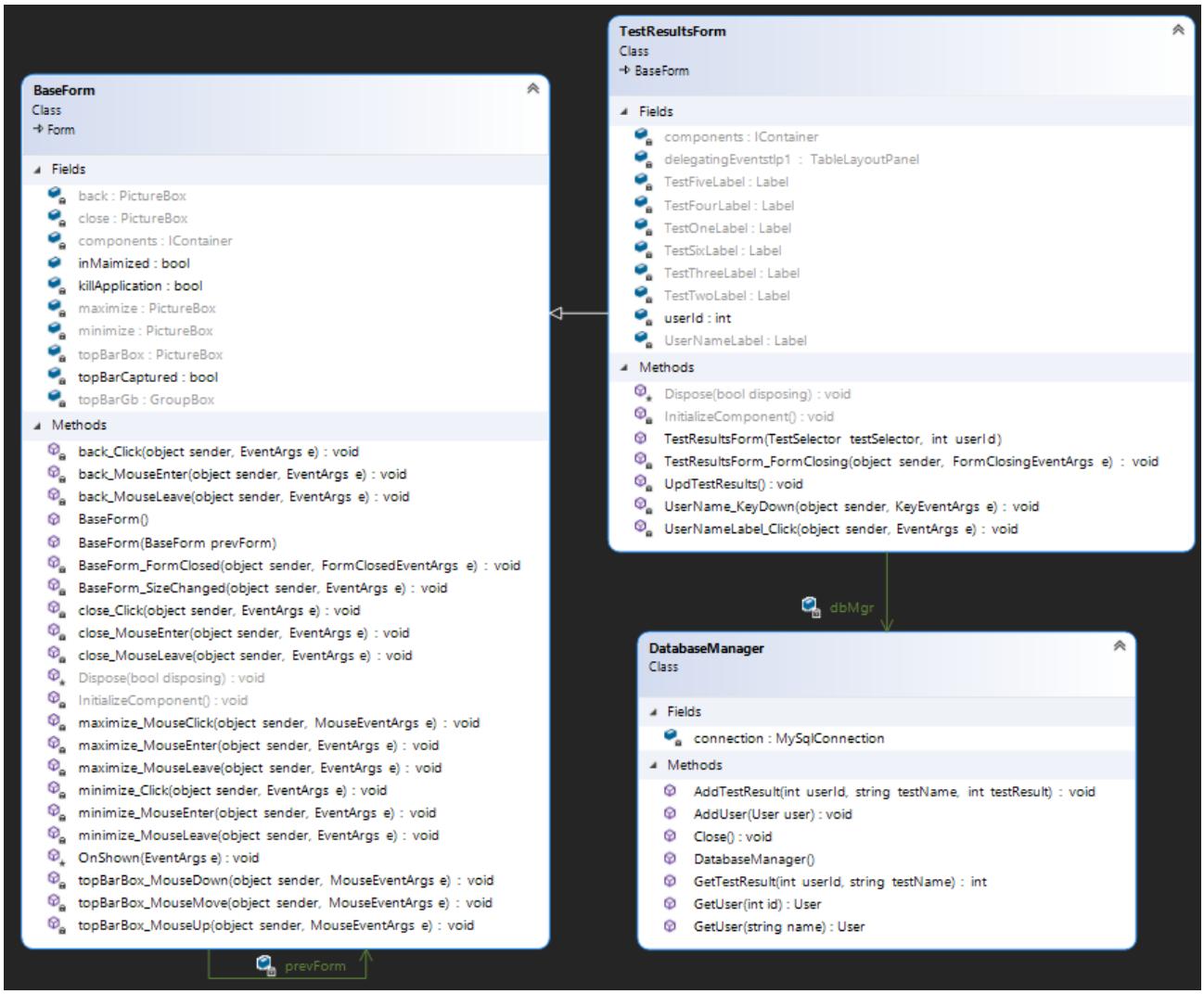

Рисунок 3.8 – Діаграма класу форми перегляду результатів тестування

3.2.2 Діаграми класів кастомних елементів інтерфейсу додатку

Клас, представлений на рис. 3.9, перевизначає властивість CreateParams. Властивість CreateParams повертає параметри створення вікна, які використовуються при створенні елементу керування. У перевизначеному методі CreateParams спочатку отримуються параметри створення базового класу RichTextBox. Потім до отриманих параметрів додається флаг 0x20, що встановлює стиль для вікна, який робить його відповідним для введення пароля.

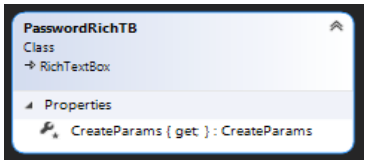

Рисунок 3.9 – Діаграма класу поля введення паролю користувача

Докладний опис полів та методів класу (див. рис. 3.10) представлено нижче:

- isHovered: Приватне поле, що вказує, чи наведений курсор на кнопку.
- GetRoundPath: Метод, який повертає об'єкт GraphicsPath, який представляє округлу форму кнопки на основі заданого прямокутника та радіуса.
- OnPaint: Перевизначений метод, який відповідає за візуальне відображення кнопки. Використовується для малювання округлої форми кнопки, градієнтної заливки фону(колір якої залежить від значення поля isHovered), малювання тексту та рамки.
- OnMouseEnter: Перевизначений метод, який відповідає за реакцію на наведення курсору на кнопку. Встановлює флаг isHovered в true та інвалідує кнопку для оновлення вигляду.
- OnMouseLeave: Перевизначений метод, який відповідає за реакцію на вихід курсору за рамки кнопки. Встановлює флаг isHovered в false та інвалідує кнопку для оновлення вигляду.

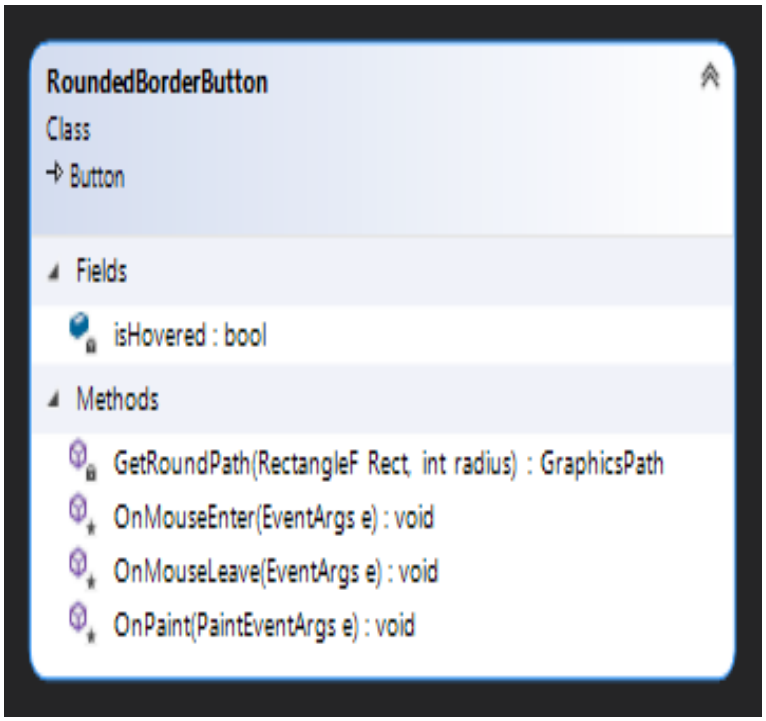

Рисунок 3.10 – Діаграма класу кнопки з заокругленими краями та градієнтною заливкою

3.2.3 Діаграми класів взаємодії з базою даних

Клас User (див. рис. 3.11)

Цей клас представляє користувача системи і містить наступні властивості:

- UserId (типу int): ідентифікатор користувача
- Name (типу string): ім'я користувача
- Password (типу string): пароль користувача

Клас має конструктор, який приймає значення для всіх трьох властивостей і ініціалізує їх.

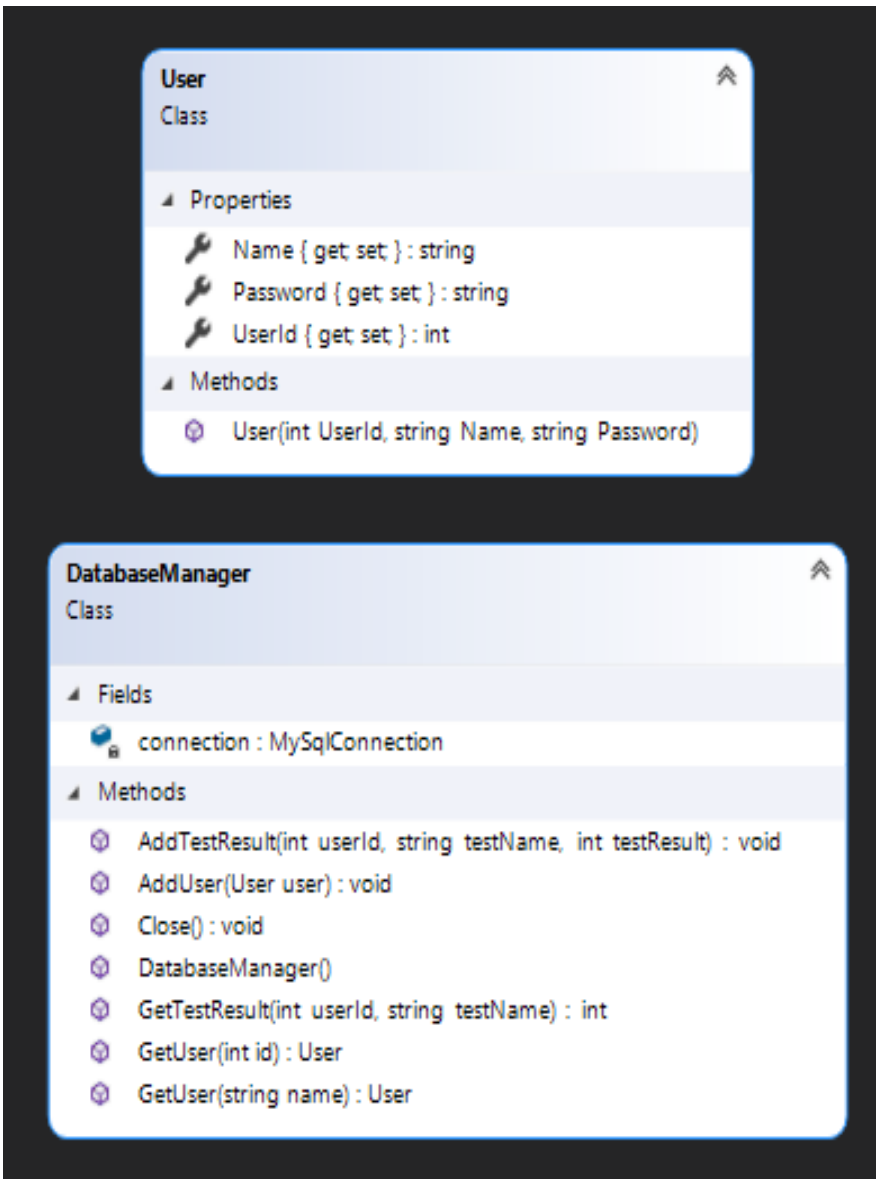

Рисунок 3.11 – Діаграма класу, що забезпечує взаємодію додатку з базою даних

Клас DatabaseManager (див. рис. 3.11)

Клас забезпечує взаємодію між локальним клієнтом додатку і віддаленою базою даних на сервері MySQL.

Конструктор класу

Конструктор DatabaseManager виконує підключення до бази даних і створює таблиці, якщо вони ще не існують. Ось декілька кроків, які він виконує:

1. Він створює об'єкт MySqlConnectionStringBuilder і налаштовує його параметри для встановлення з'єднання з базою даних. Ці параметри включають IP-адресу сервера, ідентифікатор користувача, пароль, назву бази даних та вимкнення SSL.
2. Після цього він відкриває з'єднання до бази даних, використовуючи налаштування з csb.
3. Далі він виконує два SQL-запити для створення таблиць. Перший запит створює таблицю "Users" зі стовпцями Id, Name і Password. Другий запит створює таблицю "Tests" зі стовпцями Id, UserId, Name і Result.

Метод AddUser: метод додає нового користувача до бази даних. Він отримує об'єкт типу User і виконує SQL-запит для вставки значень Name і Password у таблицю "Users". Запит формується з використанням значень властивостей об'єкта user.

Метод GetUser: метод отримує об'єкт типу string (ім'я користувача) і виконує SQL-запит для отримання користувача з бази даних за його ім'ям. Він використовує параметрізований запит з параметром @name, щоб уникнути SQL-ін'єкції. Якщо знайдений користувач, результати зчитуються і створюється новий об'єкт User з отриманих даних та повертається. Якщо користувач не знайдений, повертається значення null.

Метод GetUser: метод отримує значення типу int (ідентифікатор користувача) і виконує SQL-запит для отримання користувача з бази даних за його ідентифікатором. Запит формується з використанням значення id. Якщо знайдений

користувач, результати зчитуються і створюється новий об'єкт User з отриманих даних та повертається. Якщо користувач не знайдений, повертається значення null.

Метод AddTestResult: метод додає результат тесту для користувача до бази даних. Він перевіряє, чи вже існує результат тесту для заданого користувача та назви тесту. Якщо результат тесту не існує, виконується SQL-запит для вставки нового рядка зі значеннями userId, testName і testResult в таблицю "Tests". Якщо результат тесту вже існує, перевіряється, чи новий результат більший за існуючий, і якщо так, виконується SQL-запит для оновлення значення Result в таблиці "Tests" для відповідного рядка.

Метод GetTestResult: метод отримує значення типу int (ідентифікатор користувача) і рядок (назва тесту) і виконує SQL-запит для отримання результату тесту для заданого користувача та назви тесту. Якщо результат тесту знайдено, він повертає значення Result з результатів. Якщо результат тесту не знайдено, повертається значення 0.

Метод Close: метод закриває з'єднання з базою даних.

Цей клас дозволяє локальному клієнту додатку взаємодіяти з базою даних, виконуючи операції, такі як додавання користувачів, отримання користувачів за ім'ям або ідентифікатором, додавання результатів тестів та отримання результатів тестів. При завершенні роботи з базою даних, потрібно викликати метод Close, щоб закрити з'єднання.

3.3 Розробка іконок додатку

Для покращення якості графічного інтерфейсу та з урахуванням створення кастомного тайтлбару, для додатку було створено набір іконок, котрий можна переглянути нижче. На рис. 3.12 зображено який вигляд має ярлик додатку.

Іконка, що використовується у якості ярлика додатку має розмір 64x64 та представляє собою червоний хрест на білому тлі. Дана емблема належить до універсальних Емблем Гуманності та є одним з найпоширеніших символів, що асоціюються з медициною.

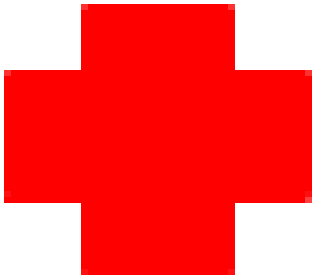

Рисунок 3.12 – Ярлик додатку

Іконки тайтлбару вікна мають розмір 30х30. При розробці зовнішнього вигляду іконок було враховано необхідність створення їх інтуїтивно зрозумілими та необхідність підсвічування іконки у випадку наведення на неї курсору, задля полегшення роботи користувача з додатком (див. рис. 3.14 – рис. 3.15).

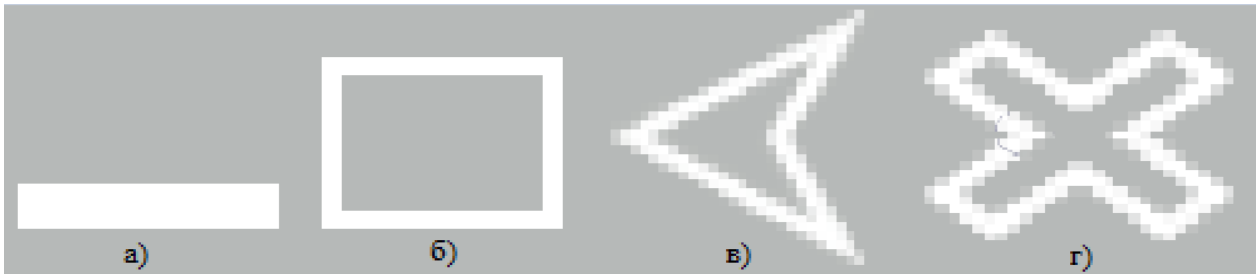

Рисунок 3.14 – Іконки тайтлбару у випадку, коли курсор не наведено на них(а – іконка мінімізації вікна, б – іконка максимізації вікна, в – іконка повернення до попереднього вікна, г – іконка закриття додатку)

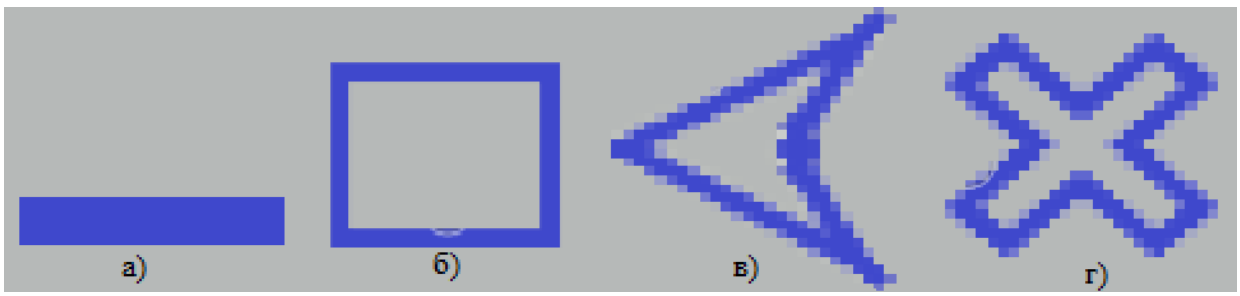

Рисунок 3.15 - Іконки тайтлбару у випадку, коли курсор наведено на них(а – іконка мінімізації вікна, б – іконка максимізації вікна, в – іконка повернення до попереднього вікна, г – іконка закриття додатку)

3.4 Розробка вікон додатку

Кожне з описаних нижче вікон виконує специфічні задачі, що призначені для забезпечення плавної та зручної взаємодії користувача з додатком. Вони забезпечують управління різними аспектами взаємодії, включаючи процес авторизації, можливість читання статей за різними розділами теорії надання першої долікарської допомоги, проходження тестів, з подальшим збереженням їх результатів до БД на віддаленому сервері та перегляд власних результатів тестування для звичайних користувачів і результатів бідь-якого користувача для адміністратора додатку, в нашому випадку – викладача.

Базова форма, яка служить основою для всіх інших вікон, відрізняється наявністю кастомного тайтлбара. Цей тайтлбар включає декілька кастомних кнопок, зокрема "закрити програму", "згорнути вікно", "максимізувати вікно", а також "повернутися до попереднього вікна". Для цих кнопок були розроблені кастомні іконки та логіка взаємодії з ними: підсвічування іконки при наведенні на неї та обробка натиснення на них. Даний тайтлбар було розроблено з метою з метою забезпечення кращого візуального сприйняття і зручності для користувача.

Додатково, для оптимізації взаємодії з додатком, була реалізована можливість перетягування вікна по робочому столу. Це дозволяє користувачу з легкістю пересувати вікно програми, забезпечуючи більшу гнучкість при роботі з додатком і надаючи можливість краще організувати своє робоче простір.

Розробка цих вікон додатку є ключовим елементом надання користувачу динамічного та інтуїтивно зрозумілого інтерфейсу. Кожне вікно було детально пророблено з урахуванням конкретних потреб користувачів та особливостей їх взаємодії з додатком. Такий підхід було обрано з урахуванням того факту, що кожен елемент є суттєвим для загального досвіду користувача, сприяє підвищенню ефективності та задоволеності від використання додатку.

Форма входу(див. рис. 3.16) допомагає впізнати користувача та встановити його унікальність. На цій формі розташовані два поля для введення даних - "ім'я

користувача" та "пароль". Ці дані потім перевіряються в базі даних (БД) на віддаленому сервері, що сприяє безпеці даних користувача і контролює доступ до наступних сегментів програми.

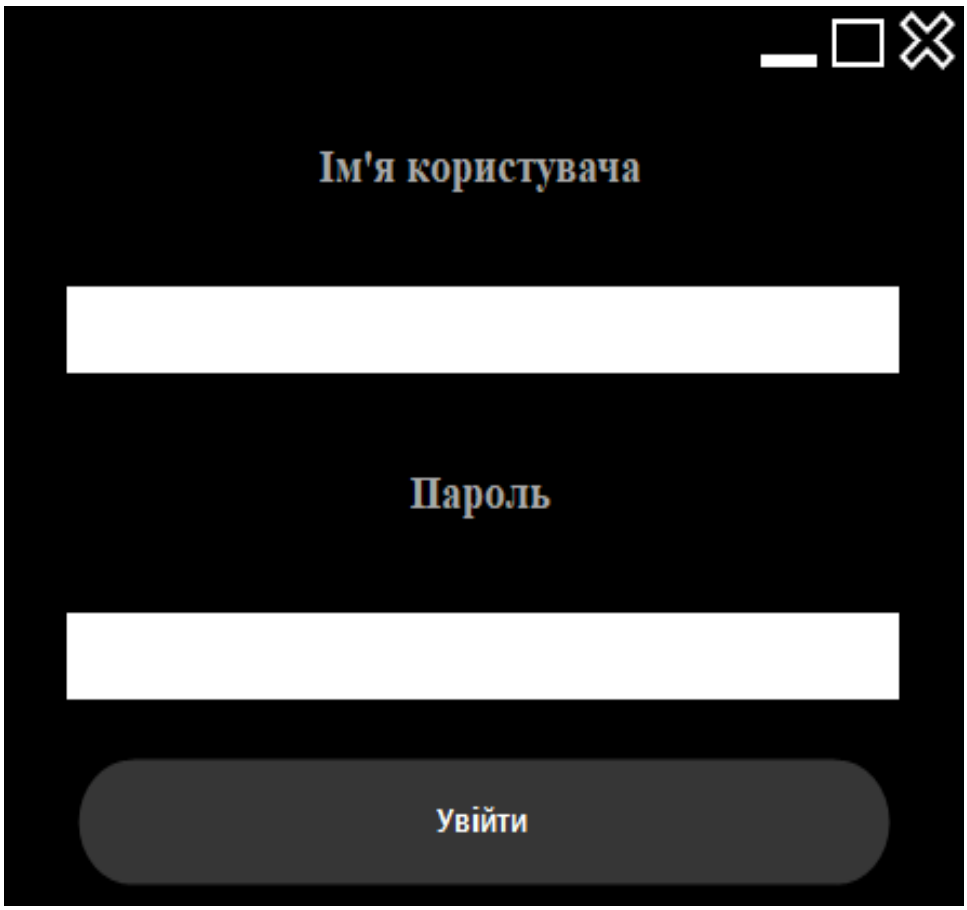

Рисунок 3.16 - Форма автентифікації користувача

Форма (див. рис. 3.17) активується у випадку, коли введені реквізити не співпадають з жодним записом в БД. У цьому випадку вікно виводить повідомлення, яке пропонує користувачеві створити новий обліковий запис. Текст повідомлення виглядає як: «Ви бажаєте зареєструвати користувача на ім'я {ім’я користувача}?» і вікно містить дві кнопки вибору - «так» і «ні». При натисненні на кнопку «так», новий обліковий запис користувача створюється в БД, і програма переходить до наступного вікна.

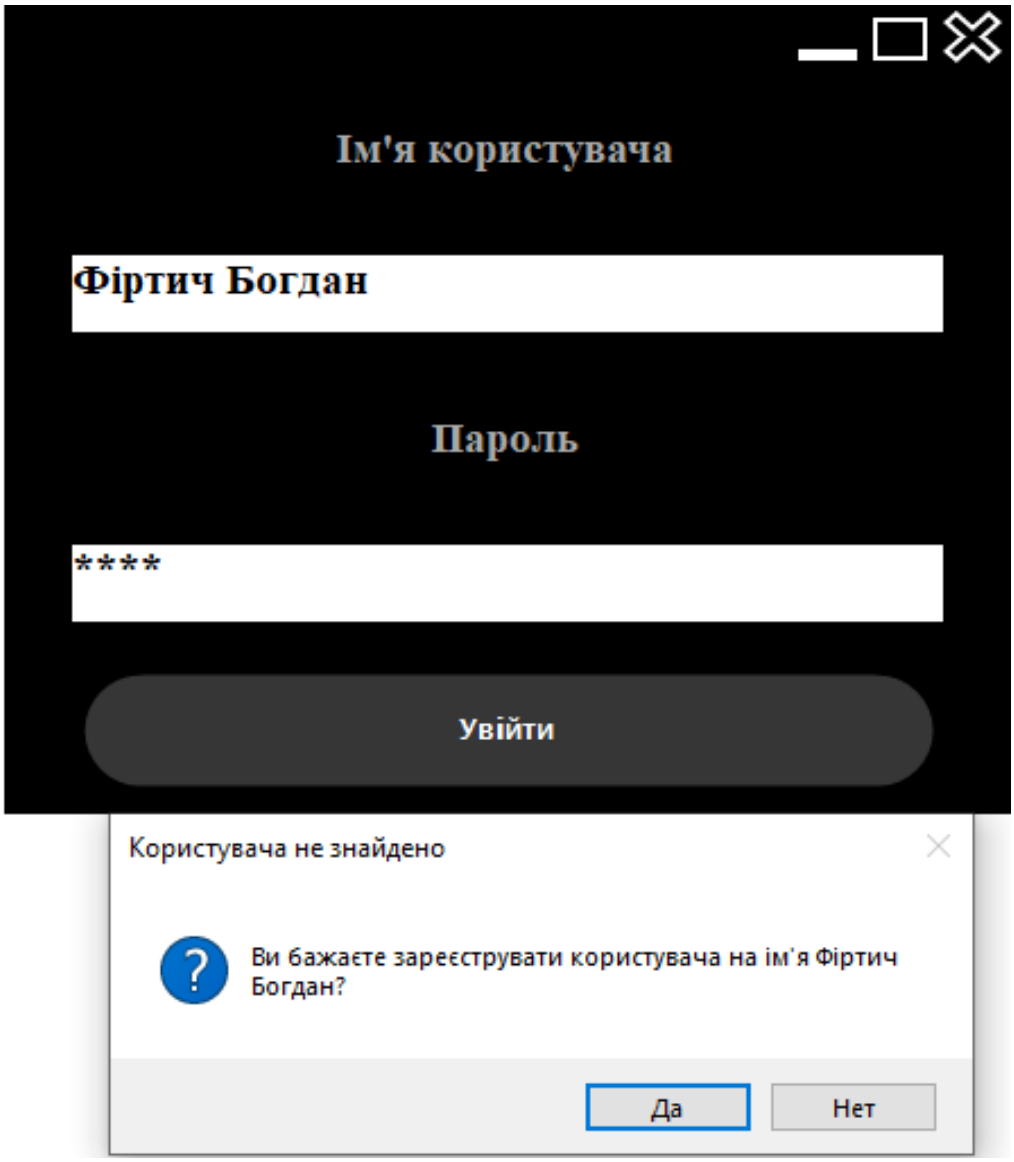

Рисунок 3.17 - Форма автентифікації користувача з вікном повідомлення про відсутність користувача з заданими реквізитами в базі

Форма (див. рис. 3.18) надає користувачеві можливість вибору між двома варіантами навчальної активності - "Навчання" та "Тести". Вибір відбувається за допомогою кнопок, що активують перехід до відповідної форми вибору навчальної статті чи тесту. Слід також зазначити, що на даній формі кнопка повернення до попереднього вікна відсутня, адже можливість входу від імені іншого користувача без попереднього завершення сеансу користування додатком для поточного користувача, не передбачена.

Рисунок 3.18 - Форма вибору навчальної активності

Форма (див. рис. 3.19) надає користувачеві довільний вибір між різними темами навчання за допомогою ряду кнопок, а саме кнопки: «Виклик швидкої для потерпілого», «ПДД при потраплянні сторонніх тіл до організму, укусах тварин та епілепсії», «ПДД при отруєнні», «ПДД при переломах, вивихах, забиттях і розтягах зв’язок», «ПДД при опіках, тепловому і сонячному ударах, обмороженні», «ПДД при пораненні та кровотечах», «ПДД при ураженні струмом» та «Що таке перша долікарська допомога?». Кожна кнопка представляє різні аспекти навчання з первинної медичної допомоги. При натисканні на кнопку, відбувається перехід до форми читача, де відображається відповідний навчальний матеріал.

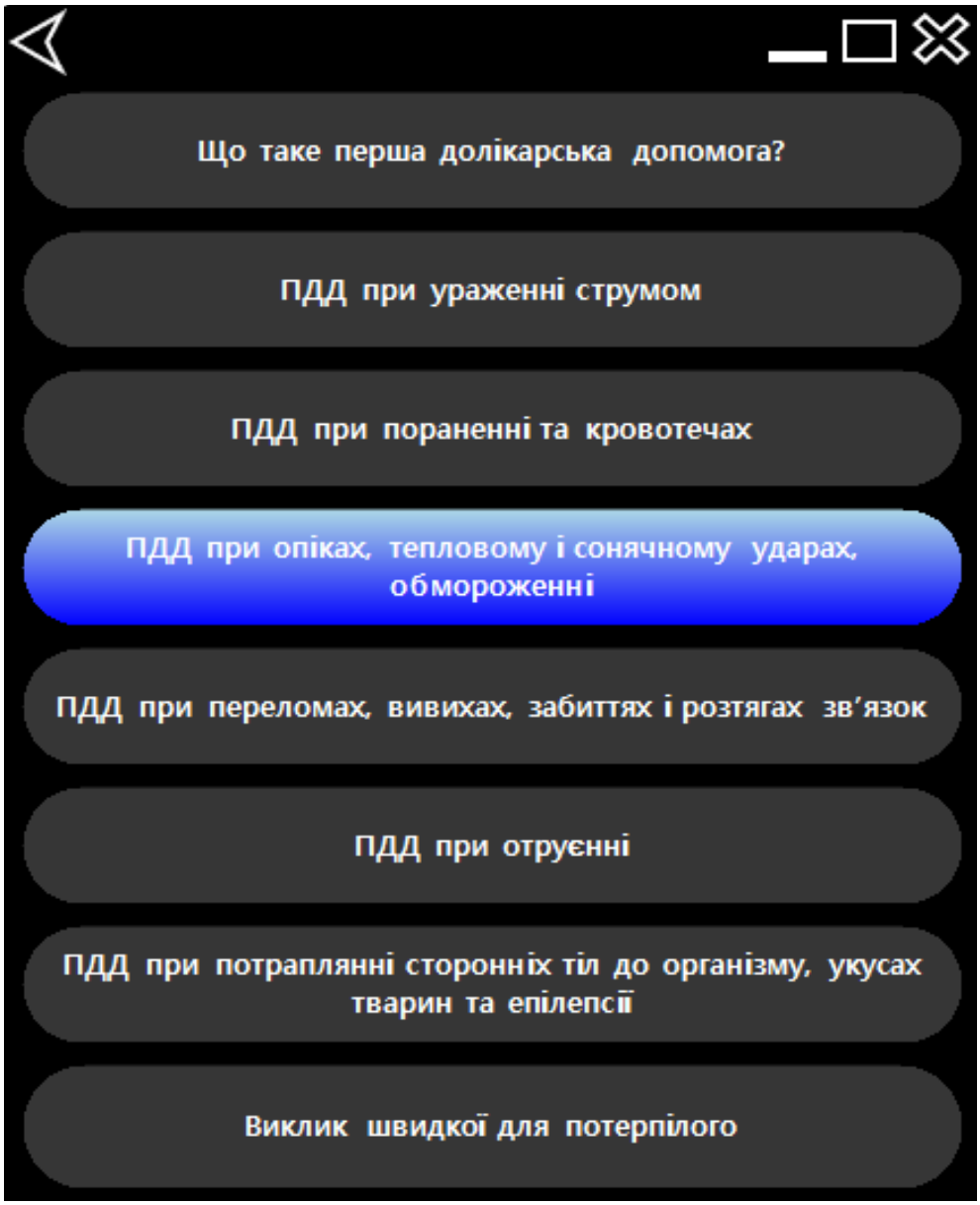

Рисунок 3.19 - Форма вибору навчальної статті

На формі (див. рис. 3.20) представлено текстовий матеріал, поділений на сторінки для зручного читання. Якщо обсяг тексту є завеликим, для його розміщення у відповідному елементі, вмикається функція вертикального скролінгу. Користувач може переглядати наступні або попередні сторінки, використовуючи кнопки «Наступна сторінка» та «Попередня сторінка», котрі зникають, якщо відкрито відповідно першу або останню сторінки.

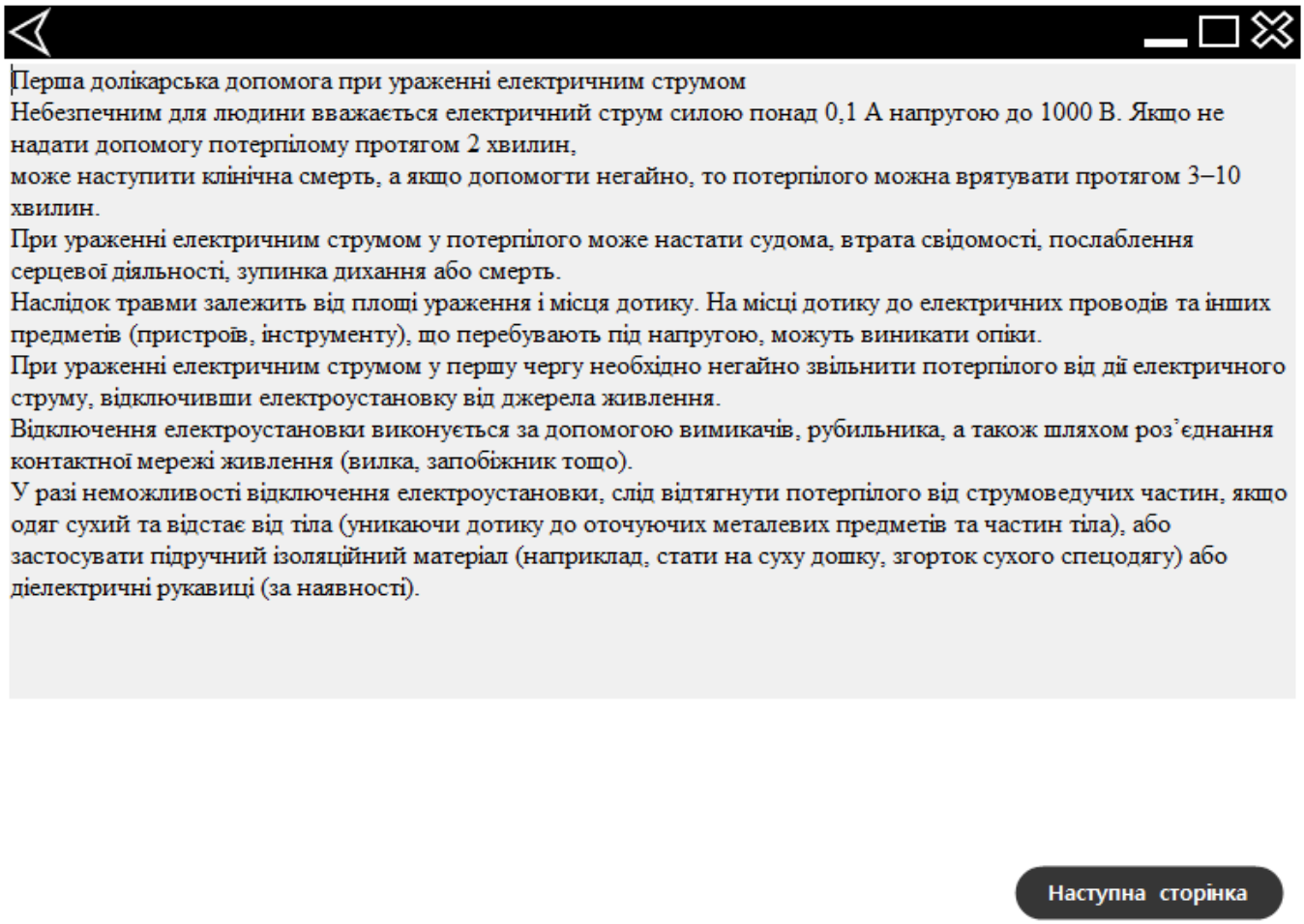

Рисунок 3.20 - Форма читача

На формі (див. рис. 3.21), окрім текстового матеріалу, відображаються зображення, що ілюструють деякі аспекти навчання з первинної медичної допомоги. Зображення допомагають користувачеві краще зрозуміти і запам'ятати матеріал.

Слід зазначити, що перед відображенням на екрані, для зображення проводиться масштабування, котре гарантує збереження пропорцій зображення, таким чином запобігаючи його деформації, котра не тільки ускладнює перегляд ілюстрацій, а й псує її зовнішній вигляд, що знизило б загальний рівень задоволеності додатком серед користувачів.

Також, для елементів перегляду текстового та ілюстративного матеріалів зберігається стане співвідношення 50/50, що запобігає перекриванню одного елементу іншим та гарантує доступ до обох типів навчального матеріалу.

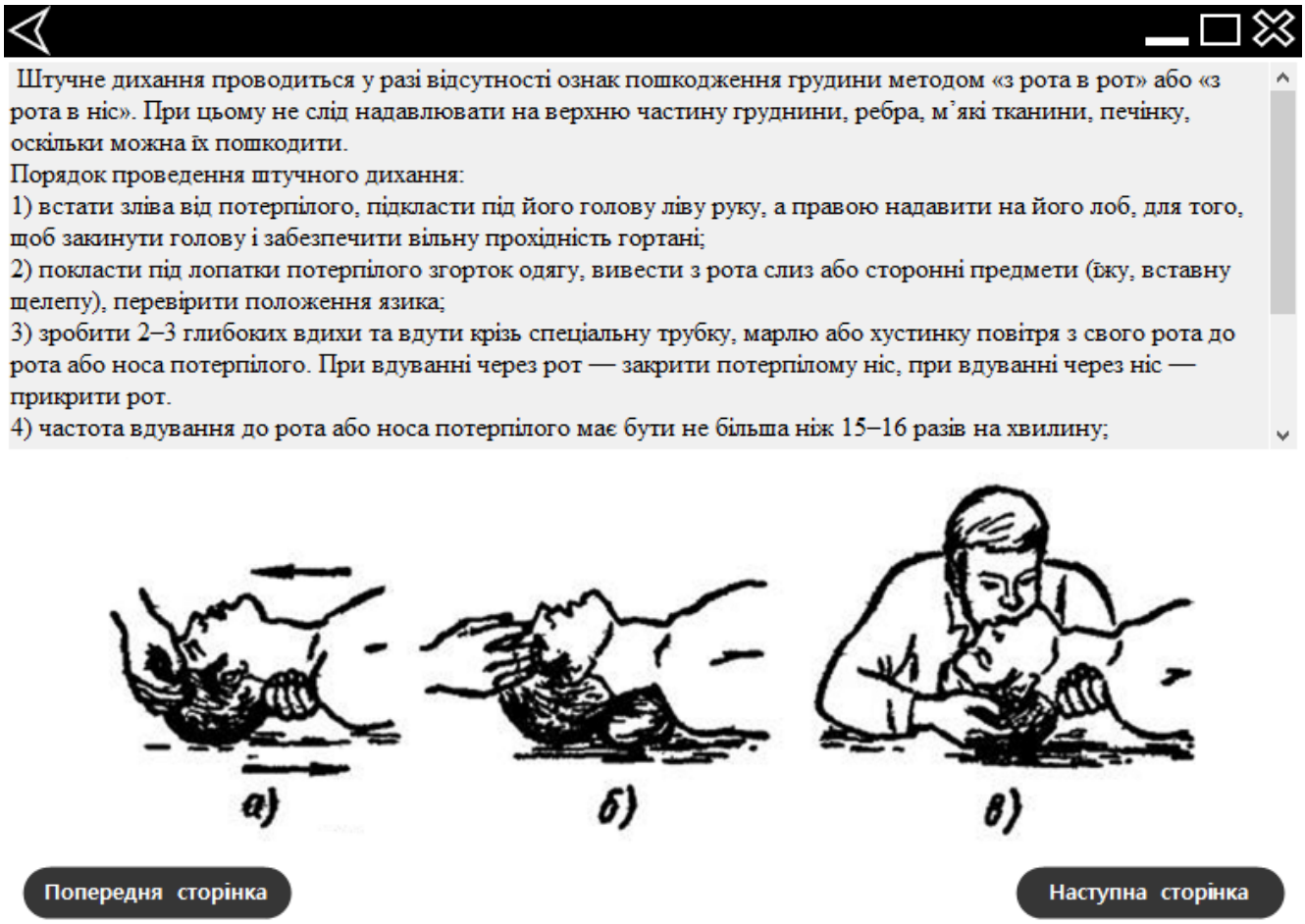

Рисунок 3.21 - Форма читача

На формі (див. рис. 3.22) користувач може вибрати тест для перевірки своїх знань з різних аспектів навчання з первинної медичної допомоги. Форма містить ряд кнопок, а саме кнопки: «Пройти тестування до розділу "ПДД при ураженні струмом"», «Пройти тестування до розділу "ПДД при пораненні та кровотечах"», «Пройти тестування до розділу "ПДД при опіках, тепловому і сонячному ударах, обмороженні"», «Пройти тестування до розділу "ПДД при переломах, вивихах, забиттях і розтягах зв'язок"», «Пройти тестування до розділу "ПДД при отруєнні"», «Пройти тестування до розділу "ПДД при потраплянні сторонніх тіл до організму, укусах тварин та епілепсії"», «Переглянути результати тестування», кожна з яких представляє тест на відповідну тему, окрім останньої. При виборі тесту, програма переходить до форми тестування. Окрім цього, користувач може переглянути свої результати тестування, натиснувши відповідну кнопку.

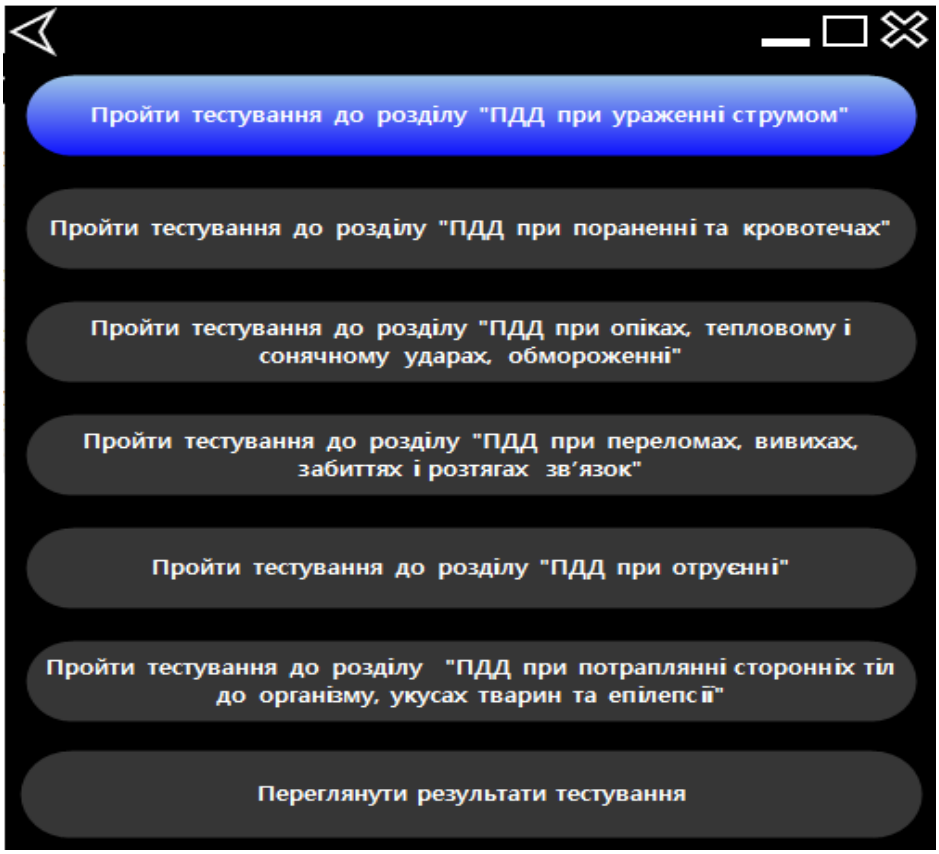

Рисунок 3.22 - Форма вибору тесту

Форма (див. рис. 3.23) містить два типи питань - на співставлення відповідей та на вибір правильної відповіді серед 4-х варіантів. Також присутня кнопка «Завершити виконання тесту», при натисненні на яку, за умови, що це найкращий результат тестування користувача за цим тестом, результати тестування записуються до БД.

Слід зазначити, що усі тести зберігаються в .xml форматі у зашифрованому вигляді, та, записуються у каталог встановлення додатку в процесі установки. При відкритті даного вікна, файл з даними тесту дешифрується, записується у тимчасовий файл, його зміст зчитується, та динамічно генеруються елементи тестової форми.

Зберігання даних тесту в зашифрованому вигляді гарантує, що користувач не зможе їх редагувати чи переглядати правильні варіанти відповідей, таким чином отримуючи високий бал недоброчесним шляхом, а динамічна генерація елементів тестової форми дозволяє запобігти створенню шаблонів форм вручну та дублюванню коду.

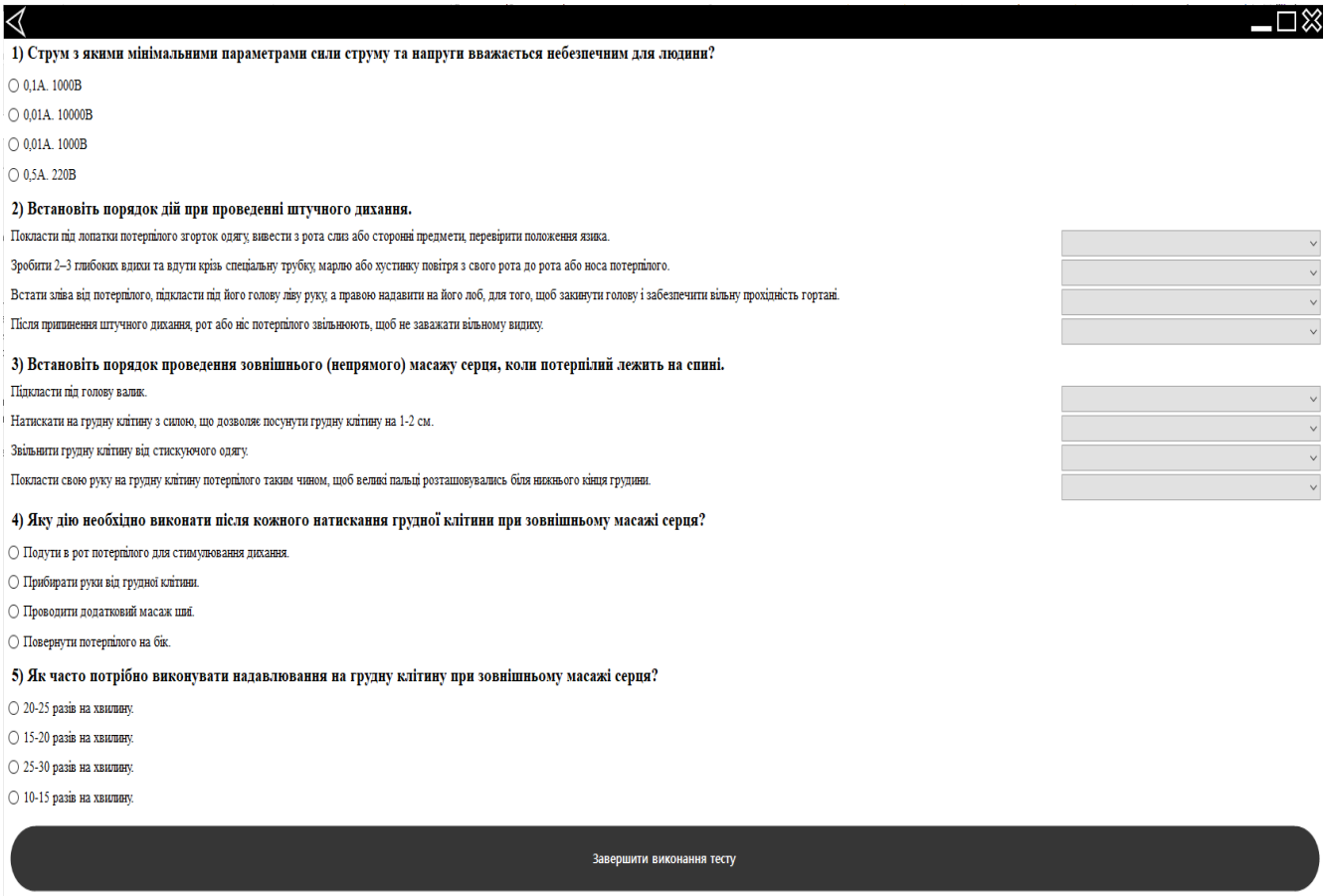

Рисунок 3.23 - Форма проходження тесту

Питання тесту (див. рис. 3.24) виділяються жирним шрифтом, у той час як варіанти відповідей не мають жодного виділення, що сприяє зручності користування вікном та надає можливість візуально ділити тест на частини.

Вибрані варіанти відповідей виділяються на формі. Для питань на вибір правильної відповіді серед 4-х варіантів, обраний варіант виділяється чорною крапкою всередині кола елементу RadioButton.

У питанні типу на співставлення відповідей навпроти відповіді з першого стовпця відображається обрана відповідь з другого стовпця.

Важливою частиною функціоналу генератора елементів форми є те, що він при кожному наступному запуску тесту випадково генерує порядок розташування варіантів відповіді у запитаннях, що ускладнює списування та сприяє здобуванню власних балів доброчесним шляхом.

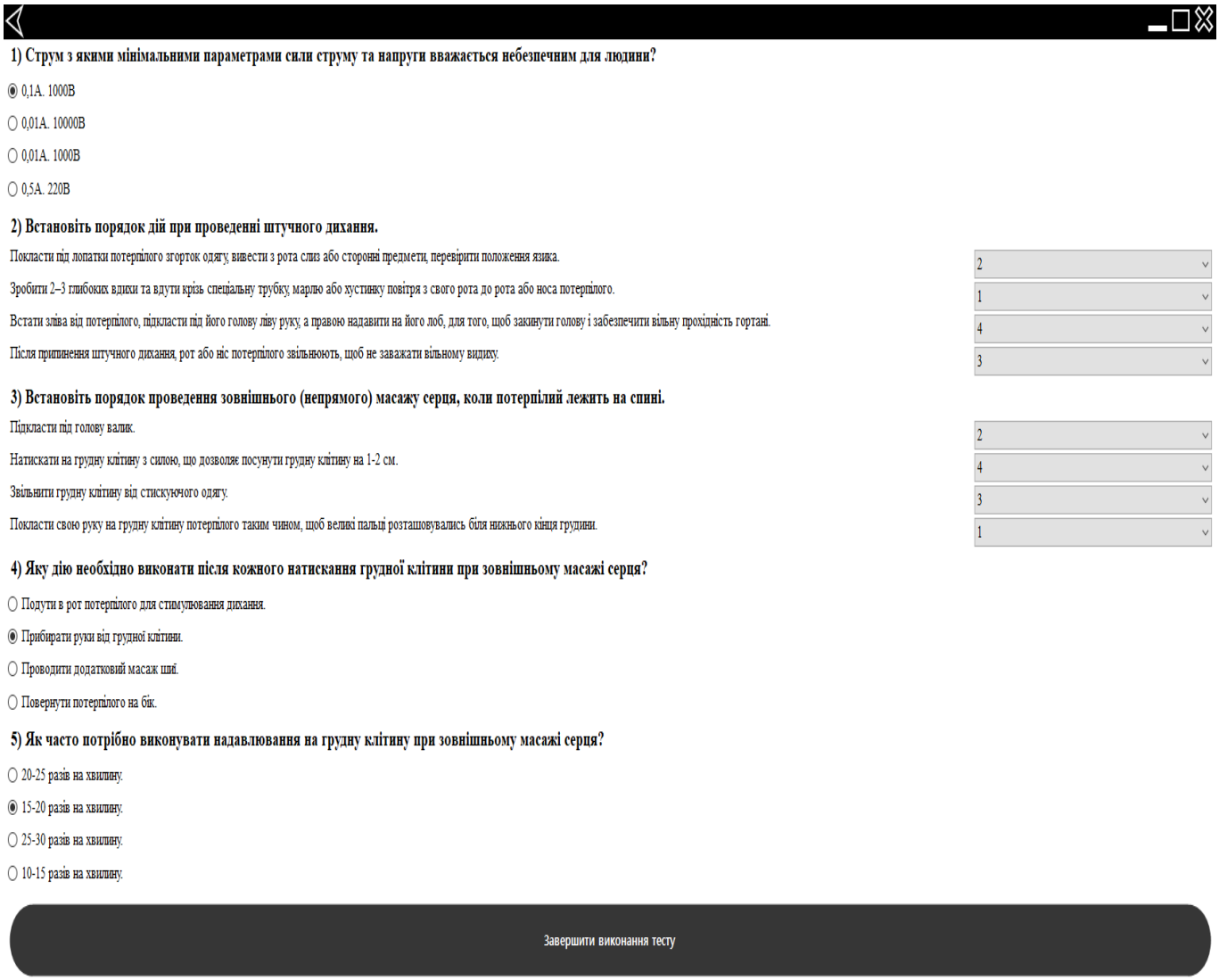

Рисунок 3.24 - Форма проходження тесту з обраними варіантами відповідей

При закритті форми (див. рис. 3.25) інакше, ніж через кнопку «Завершити виконання тесту» з'являється вікно з повідомленням «Бажаєте зберегти результати тестування?» з двома кнопками «так» і «ні». При натисненні на кнопку «так» та за умови, що це найкращий результат тестування користувача за цим тестом, результати проходження тесту записуються до БД.

Використання вікна з повідомленням про збереження результатів тестування при закритті форми не через кнопку «Завершити виконання тесту» пропонує додатковий рівень захисту від втрати даних. Користувачі можуть несподівано закрити форму або випадково натиснути неправильну кнопку, що може призвести до втрати результатів тестування. Завдяки цій функції, додаток надає користувачам

можливість зберегти свої результати перед закриттям, забезпечуючи, що їхні зусилля не були марним.

Також, великою перевагою додатку є те, що в БД зберігається лише найкращий результат тестування. Це допомагає уникнути перевантаження бази даних непотрібною інформацією, а також дозволяє користувачам не хвилюватися на рахунок того, що вони можуть пройти тест гірше, ніж попереднього разу і стимулює їх перечитати навчальний матеріал, та перепройти тест на максимальний бал.

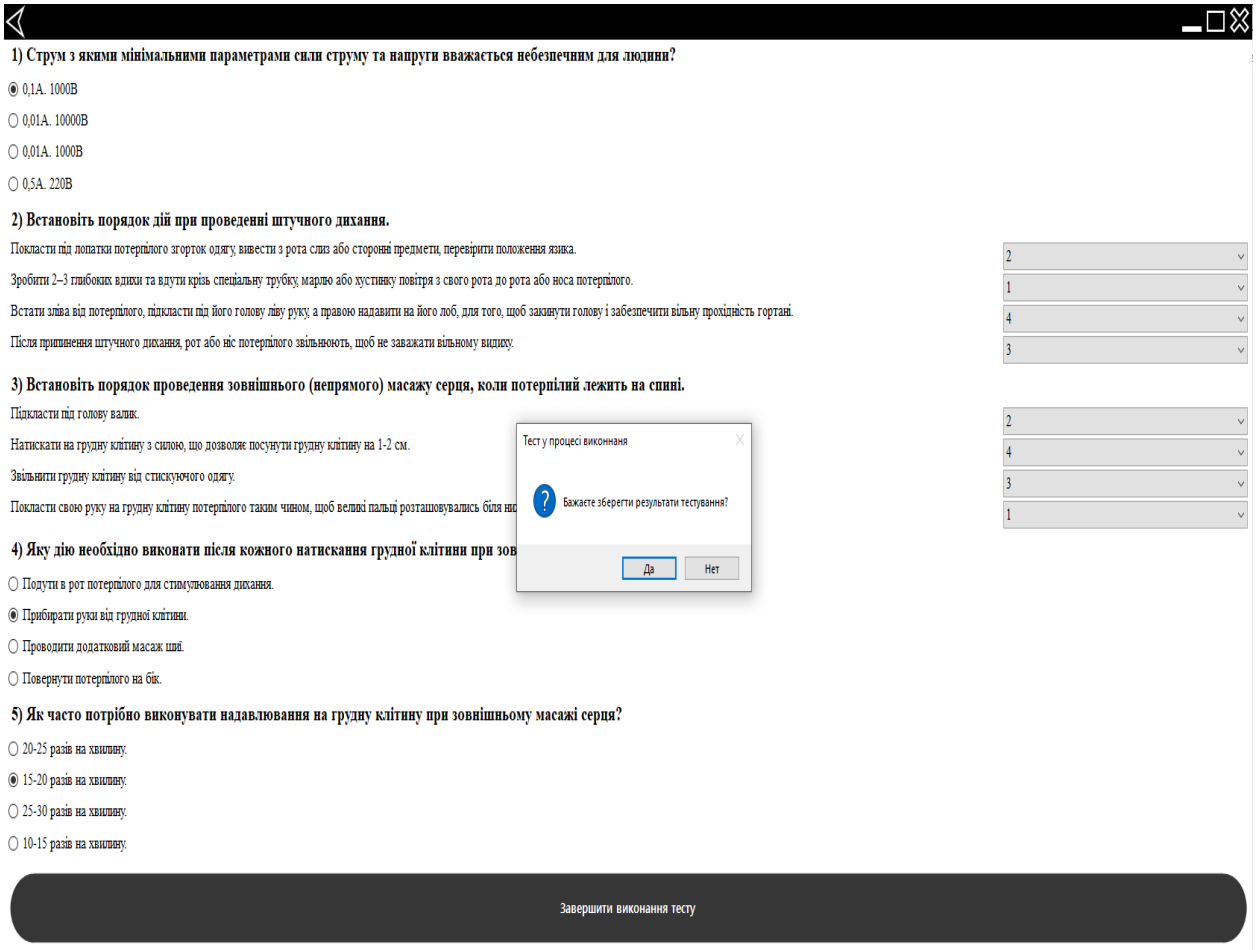

Рисунок 3.25 - Форма проходження тесту з обраними варіантами відповідей

На формі (див. рис. 3.26) відображаються результати проходження тестів користувачем. Якщо тест не був пройдений, за замовчуванням відображається 0. Якщо результат вищий за 60, колір шрифту змінюється на зелений, інакше колір є червоний.

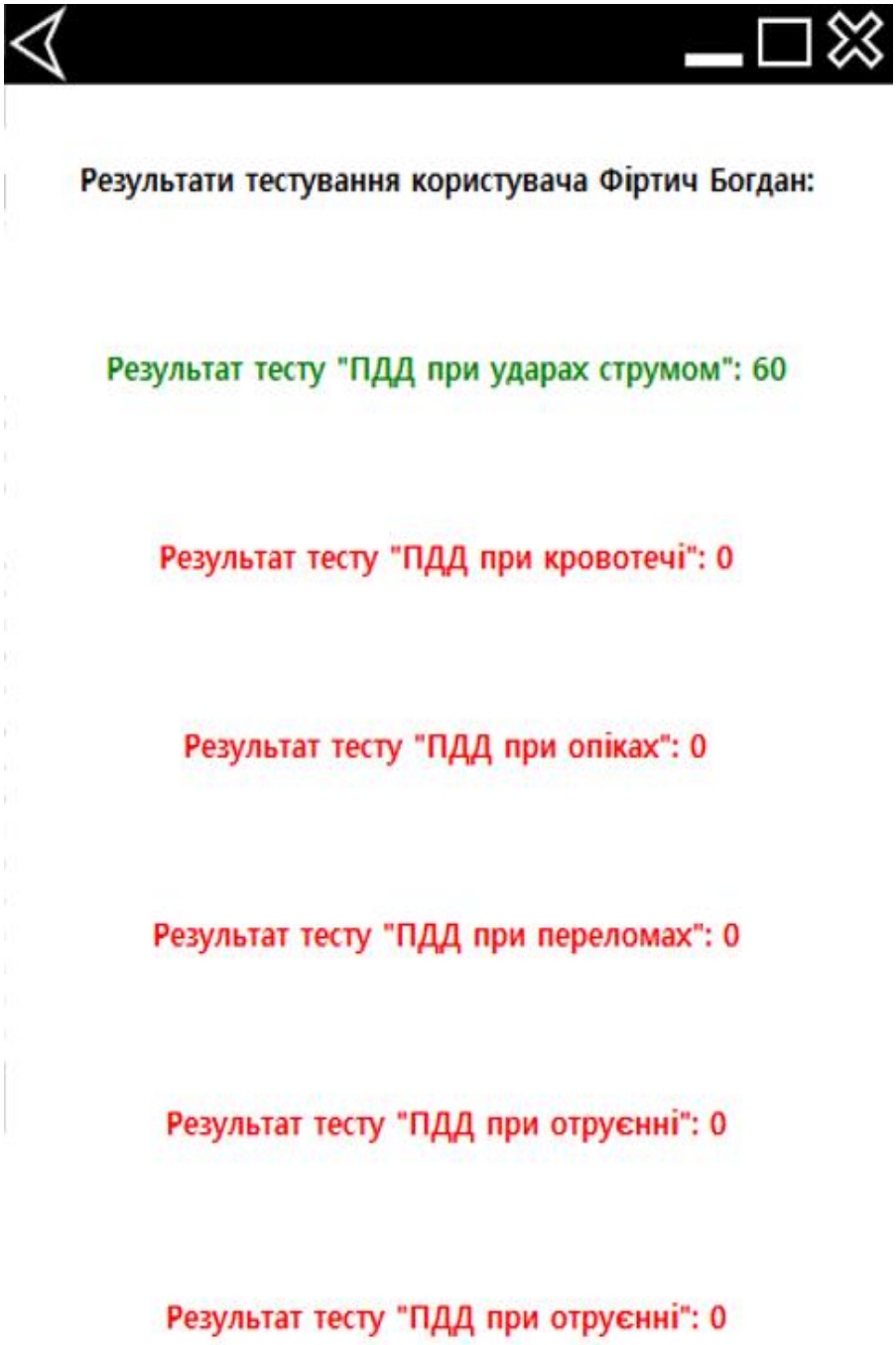

Рисунок 3.26 - Форма проходження тесту з обраними варіантами відповідей

На формі (див. рис. 3.27) додатково з’являється поле пошуку користувача за іменем. За замовчуванням при першому відображенні форми відображаються оцінки адміністратора.

Даний функціонал надає викладачеві чи інструктору можливість переглядати результати тестування своїх підлеглих та оцінювати таким чином, наскільки добре ними було засвоєно навчальний матеріал.

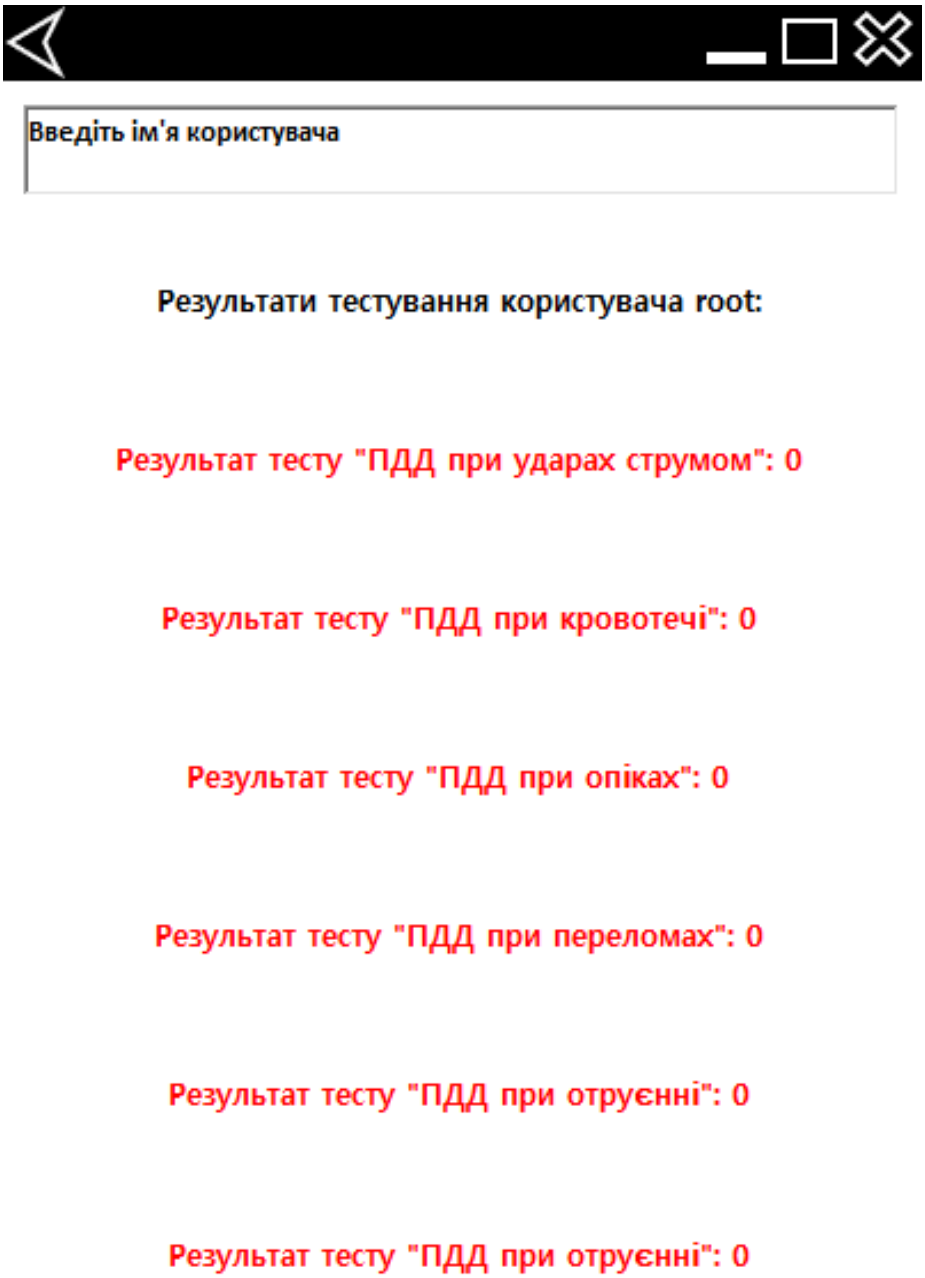

Рисунок 3.27 - Форма результатів тестування користувача при вході в додаток під обліковим записом адміністратора

Якщо користувача знайдено, на формі відображаються його оцінки (див. рис. 3.28). Якщо користувача не знайдено, відображається повідомлення про відсутність користувача.

У разі, якщо користувача було знайдено в базі, текст «Результати тестування користувача {ім'я користувача}» доповнюється його іменем, задля розуміння, для якого саме користувача виведено результати.

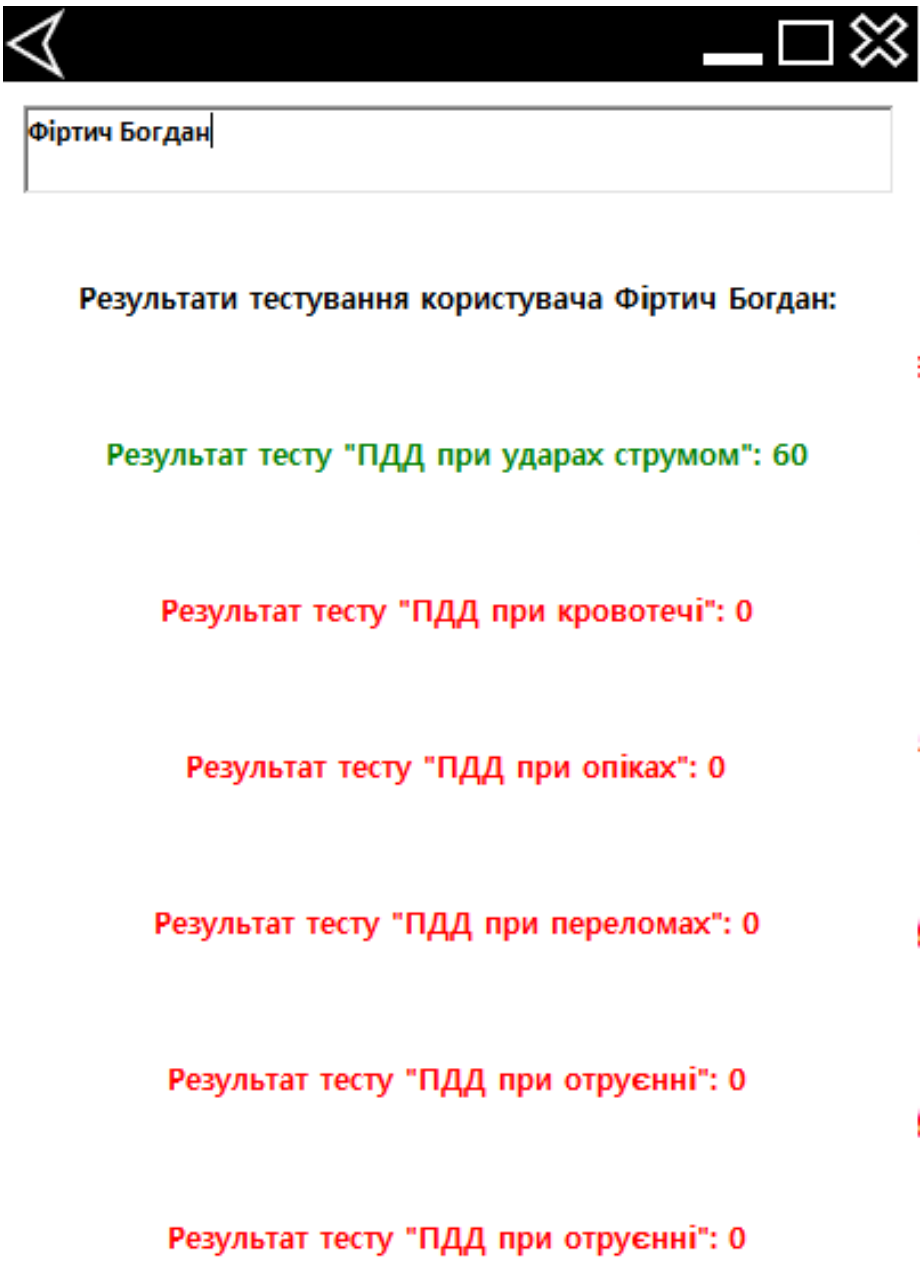

Рисунок 3.28 - Форма результатів тестування користувача при вході в додаток під обліковим записом адміністратора та позитивним результатом пошуку користувача

Якщо користувача не знайдено (див. рис. 3.29), відображається повідомлення з попередженням про відсутність шуканого користувача з текстом «Користувача {ім’я користувача} не знайдено в базі».

Така логіка сповіщення дає викладачеві змогу зрозуміти, що він допустив помилку в написанні імені студента, або студент ще не зареєструвався в системі.

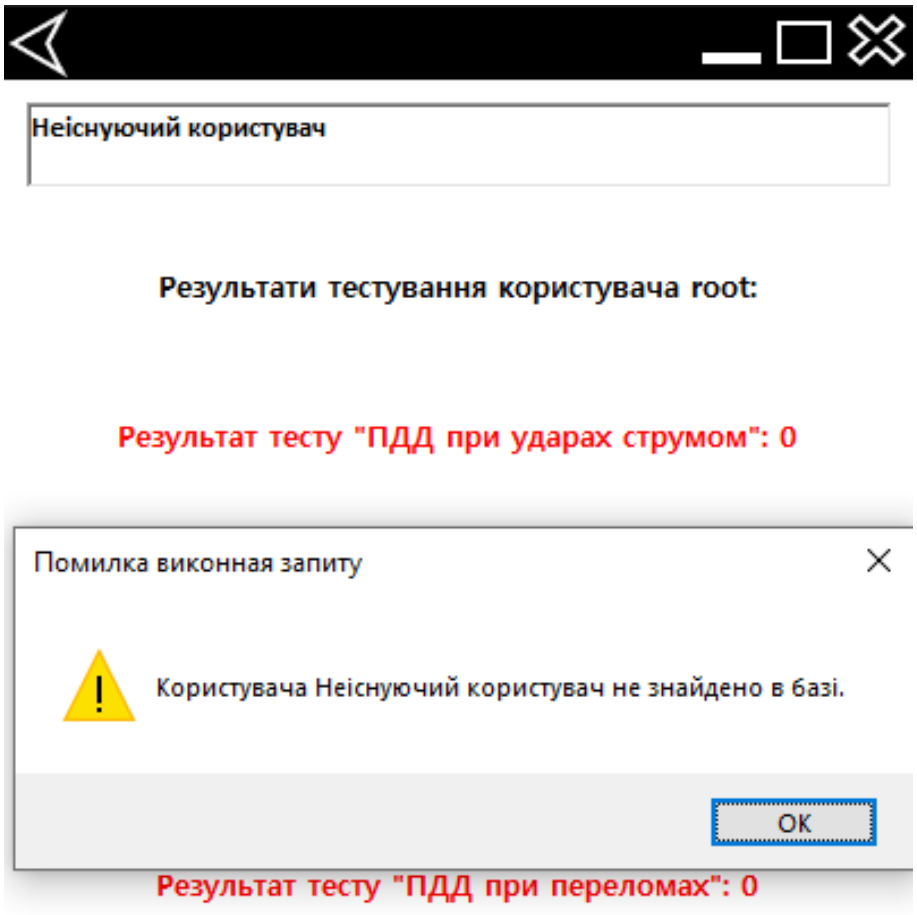

Рисунок 3.29 - Форма результатів тестування користувача при вході в додаток під обліковим записом адміністратора та позитивним результатом пошуку користувача

3.5 Функціонально-вартісний аналіз програмного продукту

Метою створення цього розділу було проведення функціонально-вартісного аналізу розробки програмного продукту задля мінімізації витрат. Після визначення основного функціоналу додатку було створено морфологічну карту, яку можна побачити на рис. 3.30.

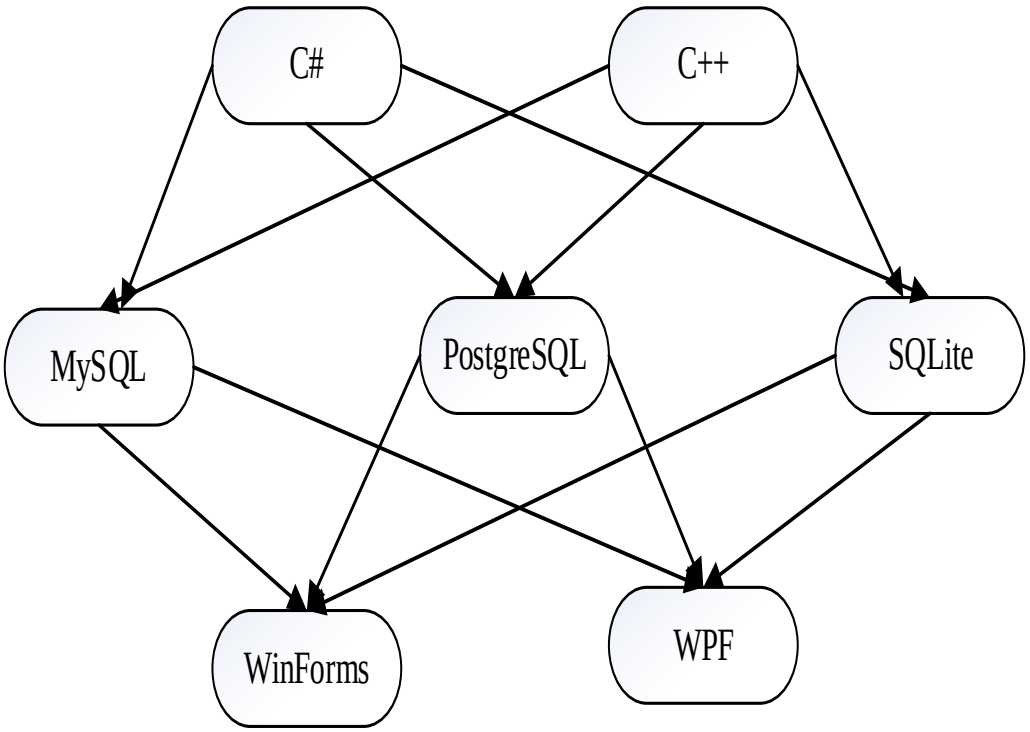

Рисунок 3.30 – Морфологічна карта варіантів реалізації функцій

З використанням даних про основні функції програмного продукту було визначено чотири найбільш перспективні варіанти його реалізації. Найефективнішим виявився четвертий варіант реалізації, який забезпечує максимальний коефіцієнт техніко-економічного рівня. Вартість його реалізації становить 424 241,42 грн.

Цей варіант передбачає:

- мову програмування C#;
- використання СУБД SQLite;
- використання фреймворку WinForms;

3.6 Безпека життєдіяльності та охорони здоров’я

В даному розділі було проаналізовано потенційні небезпеки, що можуть чатувати на студентів та працівників навчального кабінету (див. рис. 3.31), де буде використовуватись розроблена програмна система.

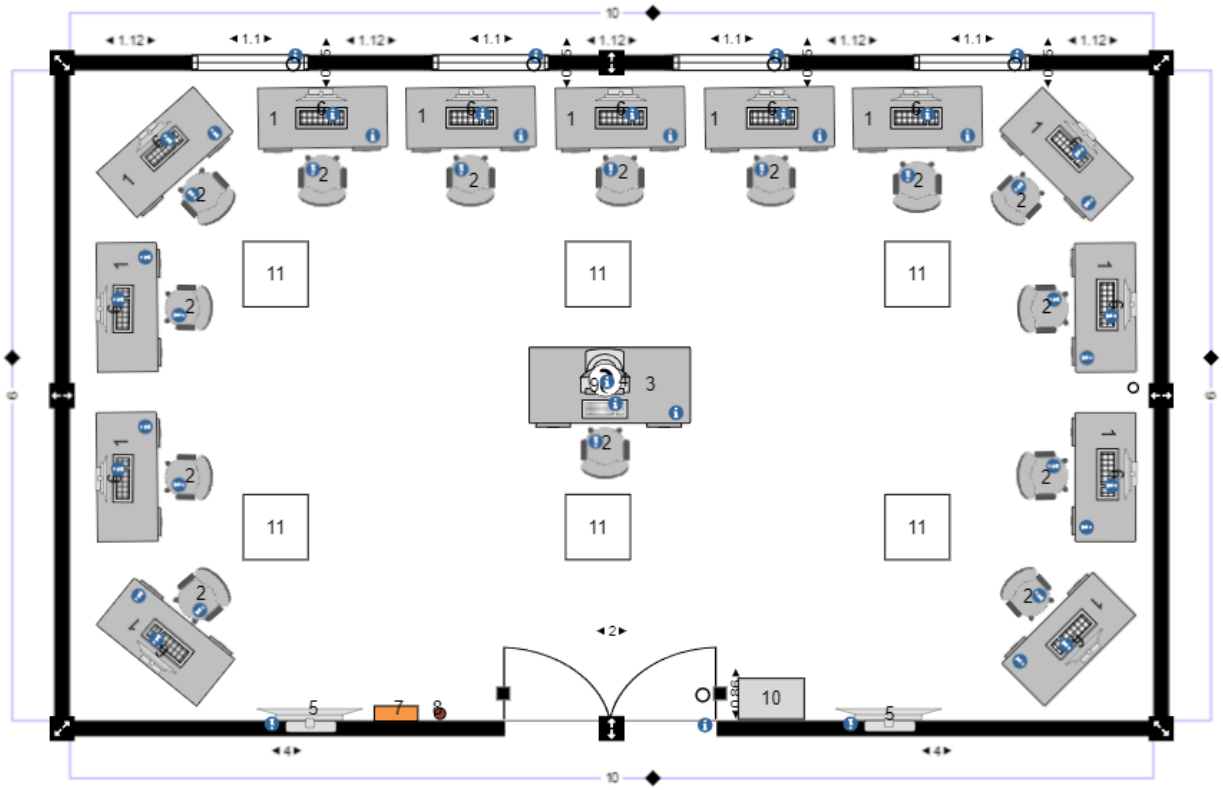

Рисунок 3.31 – Схема навчального кабінету

За результатами аналізу схеми кабінету, та заходів з попередження надзвичайних ситуацій в кабінеті було зроблено наступні висновки:

- Згідно з ДСТУ 3855-99 в даному навчальному кабінеті дотримані всі необхідні правила для забезпечення пожежної безпеки.
- Згідно з ДСТУ 3961:2019 "Експлуатація електроустановок", в даному навчальному кабінеті дотримані всі необхідні правила для забезпечення безпеки експлуатації електроприладів.

Висновок до розділу 3

У цьому розділі було представлено хід практичної роботи з розробки програмного застосунку для організації навчання з надання першої долікарської допомоги. Було проведено різні аспекти, включаючи розробку діаграм класів додатку, опис етапів роботи програми, розробку унікальних іконок та проведення економічного та безпекового аналізу.

Початок роботи полягав у побудові діаграм класів додатку, які візуалізували структуру програми та взаємозв'язки між класами. Це дозволило зрозуміти організацію коду та логіку роботи додатку. За допомогою діаграм класів, розробникам та іншим зацікавленим сторонам легше аналізувати та розуміти систему, а також виявляти потенційні проблеми або покращення.

Наступним кроком було описання етапів роботи програми, що дозволило уявити собі взаємодію користувача з додатком на кожному етапі. Це включало введення вхідних даних, обробку та аналіз цих даних, а також відображення результатів. Описані етапи роботи сприяли кращому розумінню функціональності програми та її взаємодії з користувачем.

Окрім того, були розроблені унікальні іконки, спеціально створені для додатку. Це додало естетичного вигляду та професійності до додатку, допомагаючи йому виділятися серед інших програм та забезпечувати консистентність з усім інтерфейсом.

Важливою складовою розділу був економічний аналіз вартості розробки додатку. Це включало оцінку витрат на розробку, впровадження та підтримку програмного застосунку, а також очікувані доходи та оцінку вигод від використання додатку. Економічний аналіз дав можливість оцінити фінансову стійкість та ефективність розробки додатку.

Також було проведено аналіз безпеки життєдіяльності, який включав оцінку потенційних загроз та заходів безпеки, що використовуються для забезпечення конфіденційності, цілісності та доступності даних. Аналіз безпеки дав можливість

виявити потенційні вразливості та прийняти заходи для їх запобігання або мінімізації ризиків.

Загалом, розділ практичної частини дипломної роботи надав повну картину процесу розробки програмного застосунку для організації навчання з надання першої долікарської допомоги. Включені діаграми класів, опис етапів роботи програми, розроблені іконки, економічний аналіз та аналіз безпеки життєдіяльності забезпечують повноту та цілісність дипломної роботи. Результатом роботи є функціональний і естетичний програмний застосунок, який може бути використаний для організації навчання з надання першої долікарської допомоги з врахуванням економічних та безпекових аспектів.

ЗАГАЛЬНІ ВИСНОВКИ

За результатами виконання роботи маємо повноцінний застосунок для навчання користувачів правилам першої долікарської допомоги, котра надає змогу користувачам не лише навчатись, а й перевіряти свої знання, викладачі в свою чергу отримують можливість переглядати результати тестування користувачів.

У порівнянні з існуючими аналогами додаток має свої переваги та недоліки. Першим, та, мабуть найбільшим недоліком є те, що застосунок працює лише на ОС Windows, що суттєво звужує сферу його застосування, але, у той же час робить більш стабільним та спрощує процес його підтримки. Рішенням цієї проблеми та можливістю поліпшити застосунок в цілому є його реімплементація, для підтримки кросс-платформової роботи.

Другим помітним недоліком є відсутність варіативності тестового матеріалу: додаток має обмежений набір тестів, котрі зберігаються у XML – форматі локально, а, отже, для розширення бази тестів необхідно випускати нову версію додатку. Чудовим рішенням цієї проблеми є зберігання тестів у базі даних на віддаленому хості, що дозволить розширювати тестову базу для всіх клієнтів водночас, без необхідності випуску нової версії додатку.

Третій недолік є дещо схожим на другий: текстові та ілюстративні матеріали навчальних статей зберігаються локально, тож для випуску нових чи редагування старих необхідно теж випускати нову версію додатку. Рішення третьої проблеми є аналогічним рішенню другої – зберігати матеріали статей на віддаленому хості, що надасть змогу підвантажувати нові, чи оновлювати старі дані без переустановлення додатку.

У ході виконання роботи було виконано усі поставлені на початку цілі, а саме:

1) Було проведено аналіз літературних джерел на тему розробки додатків на ОС Windows. За результатами аналізу джерел було виокремлено переваги та недоліки тих чи інших технологій розробки, котрі можна було б використати для реалізації додатку.

2) Порівняно між собою аналоги продукту, у результаті порівняння було виявлено їх слабкі та сильні сторони та ураховано отриману інформацію в подальшій розробці застосунку.

3) Спроектовано набір діаграм, котрі повністю описують компоненти застосунку, задля підвищення ефективності розробки та виявлення слабких місць системи на ранньому її етапі.

4) Власне розроблено програмний застосунок для навчання правилам надання першої долікарської допомоги, котрий включає в себе можливості читання статей, перегляду ілюстративних матеріалів, проходження тестів за засвоєним матеріалом, з подальшим збереженням результатів тестування в базу даних на віддаленому сервері. Також було реалізовано можливість перегляду оцінок користувачів, при входженні в систему з облікового запису адміністратора(викладача), що надає змогу оцінювати знання студентів, та планувати їх подальше навчання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Albahari J. C# 7.0 in a Nutshell / J. Albahari, B. Albahari., 2017. – (O'Reilly Media, Inc.). – ISBN: 9781491987650.
2. Marshall D. Programming Microsoft Visual C#2008: The Language / Donis Marshall., 2008. – ISBN: 9780735625402.
3. Perkins B. Beginning C# 7 Programming with Visual Studio 2017 / B. Perkins, J. Hammer, J. Reid., 2018. – ISBN: 978-1-119-45866-1
4. Troelsen A. Pro C# 5.0 and the .NET 4.5 Framework / Andrew Troelsen., 2012. – ISBN : 978-1-4302-4233-8.
5. Dubois P. MySQL Cookbook: Solutions for Database Developers and Administrators / Paul Dubois. – Себастопол, Каліфорнія, США.: O'Reilly Media, 2014. ISBN: 1449374026.
6. Bakharia A. Microsoft Visual C# 2005 Express Edition Programming for the Absolute Beginner. / Aneesha Bakharia. – Boston, Massachusetts, United States: Cengage Learning PTR, 2006. IBSN: 1592008186.
7. Microsoft Documentation for WinForms. [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/?view=netdesktop-7.0.MySQL Documentation.
8. Japikse P. Pro C# 7: With .NET and .NET Core / Philip Japikse. – New-York, United States: Apress, 2017. IBSN: 1484230175.
9. Petzold C. Programming Microsoft Windows with C#. / Charles Petzold. – United States: Microsoft Press., 2008. IBSN: 0735613702.
10. Sells C. Windows Forms 2.0 Programming / Chris Sells. – London, the UK.: Addison-Wesley Professional., 2002.
11. Ferrone H. Learning C# by Developing Games with Unity / Harrison Ferrone. – Birmingham, the UK.: Packt Publishing, 2015. IBSN: 1484230175.
12. Japikse P. C# 6.0 and the .NET 4.6 Framework / Philip Japikse. – New-York, United States: Apress, 2016. IBSN: 1484213335.

13. Nathan A. WPF 4.5 Unleashed / Adam Nathan. – Caramel, Indiana, Unites States.: Sams Publishing, 2013. IBSN: 0672336979.
14. Freeman A. Pro ASP.NET Core MVC 2 / Adam Freeman. – New-York, United States: Apress, 2019. IBSN: 9781484231494.
15. Stokes D. MySQL and JSON: A Practical Programming Guide / David Stokes. – New-York, United States: McGraw-Hill Education., 2018. IBSN: 9781260135442.
16. Bhushan V. First Aid for the USMLE Step 1 / V. Bhushan, K. Panagiotis. – New-York, United States: McGraw-Hill Education., 2021. IBSN: 126046752X.
17. British Red Cross. First Aid Training by British Red Cross [Електронний ресурс] / British Red Cross – Режим доступу до ресурсу: https://www.redcrossfirstaidtraining.co.uk/.
18. American Health Care Academy. CPR & First Aid Certification Course [Електронний ресурс] / American Health Care Academy – Режим доступу до ресурсу: https://www.nationalcprfoundation.com/courses/standard-cpr-aed-first-aid/.
19. Australian Red Cross. First Aid App by Australian Red Cross [Електронний ресурс] / Australian Red Cross – Режим доступу до ресурсу: https://apps.apple.com/au/app/first-aid-australian-red-cross/id69688097
20. Крейдич І. М. Економіка та організація виробництва «Рекомендації до виконання економіко-організаційного розділу дипломних робіт для студентів всіх технічних спеціальностей» [Електронний ресурс] / І. М. Крейдич, Н. В. Семенченко, Н. В. Рощина. – 2018. – Режим доступу до ресурсу: https://ela.kpi.ua/handle/123456789/26580.
21. Albahari J., Albahari B. C# 7.0 in a Nutshell. Sebastopol, California, USA: O'Reilly Media, 2017. ISBN: 9781491987650. (Англійська)
22. Marshall D. Programming Microsoft Visual C# 2008: The Language. Redmond, Washington, USA: Microsoft Press, 2008. ISBN: 9780735625402. (Англійська)
23. Perkins B., Hammer J., Reid J. Beginning C# 7 Programming with Visual Studio 2017. Indianapolis, Indiana, USA: John Wiley & Sons, 2018. ISBN: 978-1-119-45866-1. (Англійська)

24. Troelsen A. Pro C# 5.0 and the .NET 4.5 Framework. Berkeley, California, USA: Apress, 2012. ISBN: 978-1-4302-4233-8. (Англійська)
25. Dubois P. MySQL Cookbook: Solutions for Database Developers and Administrators. Sebastopol, California, USA: O'Reilly Media, 2014. ISBN: 1449374026. (Англійська)