ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: « **Розробка програмного за безпечення для оцінки знань студентів з дискретної математики. Спец частина. Розробка модулю «Теорія множин» мовою C#**»

Виконав: студент 4 курсу, групи ПД–44
спеціальності
121 Інженерія програмного забезпечення
Гуцан Д.В.
(прізвище та ініціали)

Керівник Садовенко В.С.
(прізвище та ініціали)

Рецензент
(прізвище та ініціали)

Нормоконтроль
(прізвище та ініціали)

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут Інформаційних технологій
Кафедра Інженерії програмного зебезпечення
Ступінь вищої освіти «Бакалавр»
Спеціальність 121 «Інженерія програманого забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного зебезпечення

Негоденко В.В.
“ ___ ” _____ 2023 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Гуцан Данило Вячеслвовиче
(прізвище, ім‘я, по батькові)

1.Тема роботи: « Розробка програмного забезпечення для оцінки знань студентів з дискретної математики. Спец частина. Розробка модулю «Теорія множин» мовою C#»
Керівник роботи: Садовенко В.С., д.т.н., доц., зав. кафедри ТЦР
(прізвище, ім‘я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом вищого навчального закладу від «24» лютого 2023 року №26
2. Строк подання студентом роботи «1» червня 2023 року
3. Вихідні дані до роботи:
   3.1 Науково-технічна література з питань, пов‘язаних із застосування веб- додатків;
   3.2 Практичній досвід розробки веб-додатків.
   3.3 Концепція побудови веб-додатків;
4. Перелік демонстраційних матеріалів
   4.1 Тема дипломної роботи
   4.2 Мета роботи. Об’єкт дослідження. Предмет дослідження.
   4.3 Результат дослідження розробки веб-додатків.
   4.4 Результат дослідження фреймворків для веб-розробки.
   4.5 Результати дослідження та опис реалізації програми.
   4.6 Апробація результатів дослідження
   4.7 Висновки
5. Дата видачі завдання 25.02.2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	25.02.2023 - 25.02.2023	виконано
2	Аналіз та дослідження існуючих аналогів	26.02.2023 - 10.03.23	виконано
3	Дослідження програмних об’єктів	11.03.2023 - 20.03.23	виконано
4	Моделювання об’єкту проектування	25.02.2023 - 01.03.23	виконано
5	Розробка веб-додатку	02.03.2023 - 20.03.23	виконано
6	Вступ, висновки, реферат	21.03.2023 - 24.04.23	виконано
7	Розробка обов’язкових демонстраційних матеріалів	25.04.2023 - 10.05.23	виконано
8	Попередній захист роботи	19.05.2023	виконано
9	Здача роботи	01.06.2023	виконано

Студент _____ Гуцан Д. В
(підпис) (прізвище та ініціали)

Керівник роботи _____ Садовенко В.С.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи: 77с., 12 рис., , 16 джерел.

Мова програмування C#, ОБРОБКА ЗАПИТІВ

Об’єкт дослідження - при розробці програмного забезпечення для оцінювання знань студентів з дискретної математики є сама дисципліна дискретної математики та процес викладання цієї дисципліни. Дискретна математика - це фундаментальний розділ математики, який вивчає дискретні структури, такі як множини, графи, послідовності та комбінаторику. Вона має широке застосування в інформатиці, криптографії, оптимізації, логіці та інших галузях.

Предмет дослідження – програмне забезпечення для оцінки знань с тудентів з дискретної математики. Розробка модулю «Теорія множин» мовою C#

Мета роботи - спрощення процесу оцінювання студентів з дискретної математики совою програмування c#.

Методи дослідження – методи обробки та аналізу запитів користувачів, методи побудови пре-тренованої моделі для видачі відповідей, що задовольняють вимогам запиту.

Галузь використання – програму може використовувати студентпершокурсник, чи будь-яка інша людина, яка хоче вивчати дисципліну Дискретна математику

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БЗ- база даних

ПЗ- програмне забезпечення

ВСТУП

Розробка програмного забезпечення для оцінювання знань студентів з дискретної математики", з акцентом на розробку модуля "Теорія множин" на мові C#:

Актуальність теми: Тема розробки програмного забезпечення для оцінювання знань студентів з дискретної математики є актуальною в сучасній освітній сфері. Дискретна математика є важливою складовою багатьох інженерних та комп'ютерних дисциплін і вимагає глибокого розуміння таких понять, як теорія множин. Розробка програмного забезпечення, що дозволяє ефективно оцінювати знання студентів у цій галузі, є важливим інструментом для вдосконалення навчального процесу та підвищення якості освіти.

Мета дослідження: Метою дослідження є розробка програмного забезпечення, яке дозволить оцінити знання студентів з дискретної математики, зосередившись зокрема на модулі "Теорія множин". Основним завданням є створення зручного та ефективного інструменту, який допоможе викладачам автоматизувати процес оцінювання знань студентів з даної теми, забезпечивши точність та об'єктивність отриманих результатів.

Об'єкт та предмет дослідження: Об'єктом дослідження є процес оцінювання знань студентів з дискретної математики, зокрема в контексті теорії множин. Предметом дослідження є розробка програмного модуля, який дозволить автоматизувати оцінювання знань студентів з даної теми за допомогою мови програмування C#. Основна увага приділяється розробці функціональності модуля, пов'язаної з теорією множин, включаючи операції додавання, видалення, об'єднання, перетину та різниці множин, а також перевірку включення, рівності та порівняння множин.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Огляд існуючих програм для оцінювання знань з дискретної математики:
Перш ніж розробляти власне ПЗ для оцінювання знань з дискретної математики, важливо проаналізувати існуючі програми, які вже використовуються в навчальних закладах. Деякі з них можуть мати функціональність, пов'язану з теорією множин, або бути загальними інструментами для оцінювання знань з різних дисциплін. Варто дослідити такі програми, а також літературу та статті, щоб отримати огляд існуючих рішень.

Проаналізуйте недоліки та переваги існуючих програм:
Після огляду існуючих програм необхідно проаналізувати їх переваги та недоліки. Слід розглянути наступні аспекти:

Функціональність: Які операції та функціональність пропонують існуючі програми для оцінювання дискретної математики. Перевірте, чи підтримують вони функції, пов'язані з теорією множин, які ви плануєте реалізувати у вашому модулі.

Інтерфейс та взаємодія: Дослідити, які функції інтерфейсу використовують існуючі програми для оцінювання знань. Оцініть, наскільки зручний та інтуїтивно зрозумілий їхній інтерфейс для студентів та викладачів.

Недоліки: Визначети недоліки, які можуть бути пов'язані з існуючими програмами, наприклад, обмежена функціональність, недостатня точність оцінювання, складність у налаштуванні або обмеження на використання певних типів запитань чи завдань.

Переваги: Підкресліти переваги, які мають існуючі програми. Це може бути простота використання, можливість інтеграції з іншими системами, підтримка різних типів запитань або розширений функціонал, який ви можете використати як основу для розробки власного програмного забезпечення.

1. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНКИ ЗНАНЬ СТУДЕНТІВ З ДИСКРЕТНОЇ МАТЕМАТИКИ

1.1 Опис модулю «теорія множин»

Опис функцій модуля:

-Модуль Теорія множин має такі основні функції:

Додавання множини: Ця функція дозволяє користувачеві додавати нові множини до системи. Користувач може ввести елементи множини вручну або завантажити їх з файлу.

-Видалення набору: Ця функція дозволяє користувачеві видалити раніше додані набори з системи. Користувач може вибрати набір, який потрібно видалити, зі списку доступних наборів.

-Об'єднати набори: Ця функція виконує операцію об'єднання двох вибраних наборів. Результат об'єднання зберігається в системі як новий набір.

-Перетин множин: Функція виконує операцію перетину двох вибраних множин і зберігає результат як нову множину.

-Різниця множин: Ця функція виконує операцію різниці двох вибраних множин і зберігає результат як нову множину.

-Перевірити включення: Функція дозволяє перевірити, чи входить одна множина в іншу. Вона повертає відповідну логічну відповідь.

-Перевірка рівності: Ця функція дозволяє перевірити, чи є дві множини рівними. Вона повертає відповідну логічну відповідь.

-Опис алгоритму модуля:

Алгоритм роботи модуля Теорія множин може бути наступним:

-Завантаження або додавання множин: Користувач має можливість завантажити множини з файлу або додати їх вручну в систему.

Вибір множин для виконання операцій: Користувач вибирає дві множини, над якими потрібно виконати операцію (наприклад, об'єднання, перетин або різниця).

Виконання вибраної операції: Вибрана операція виконується над вибраними наборами, а результат зберігається в системі як новий набір.

Повторити кроки 2-3 або виконати інші операції: Користувач має можливість повторити кроки 2-3 для виконання інших операцій над наборами або виконання інших функцій модуля.

- Опис структури бази даних для зберігання результатів тестування:

Для зберігання результатів тестування інформації про множини та операції над ними може бути використана БЗ. Структура бази даних може бути наступною:

Таблиця "Множини": Містить поля для зберігання унікального ідентифікатора множини, назви множини та її елементів.

Таблиця "Операції": Містить поля для зберігання унікального ідентифікатора операції, типу операції (об'єднання, перетин, різниця тощо), посилання на вхідні множини та результат операції (ідентифікатор нової множини).

Таблиця "Результати тестування": Містить поля для зберігання інформації про тест, такої як ідентифікатор студента, ідентифікатор виконаної операції та час виконання.

Така структура бази даних дозволить вам зберігати та відстежувати історію виконаних операцій, результати тесту, а також зв'язок між множинами та операціями над ними.

Мій модуль "Теорія множин" забезпечить зручність використання та ефективність роботи з операціями теорії множин в контексті оцінювання знань студентів з дискретної математики.

1.2 Розробка модулю «теорія множин» мовою c#

- Вимоги до розробки:

Перед розробкою модулю «Теорія множин» мовою C#, необхідно встановити наступні вимоги:

Мова програмування: Використання мови C# для розробки модулю.

Інтеграція з основною системою: Модуль повинен бути здатним інтегруватись з основною системою оцінювання знань студентів з дискретної математики.

Інтерфейс користувача: Модуль повинен мати зручний інтерфейс користувача, що дозволяє виконувати різні операції з множинами.

Функціональність: Модуль повинен підтримувати функції додавання множин, видалення множин, виконання операцій об'єднання, перетину, різниці, а також перевірку включення та рівності множин.

- Опис структури програмного коду:

Структура програмного коду модулю може мати наступні складові:

-Головний клас: Це основний клас модулю, який виконує ініціалізацію та керування модулем.

-Клас "Множина": Цей клас представляє множину і містить методи для додавання, видалення та виконання операцій з множинами.

-Клас "Операція": Цей клас представляє операцію над множинами і містить методи для виконання операції об'єднання, перетину, різниці, а також перевірки включення та рівності множин.

-Клас "БД": Цей клас відповідає за зберігання та керування даними, пов'язаними з множинами та операціями. Він містить методи для зберігання, видалення та отримання даних з бази даних.

- Опис функцій програмного коду:

Програмний код модулю «Теорія множин» мовою C# може містити наступні функції:

AddSet(): Метод для додавання нової множини до системи. Він отримує елементи множини як параметр і додає їх до бази даних.

RemoveSet(): Метод для видалення існуючої множини з системи. Він отримує ідентифікатор множини як параметр і видаляє відповідну множину з бази даних.

Union(): Метод для виконання операції об'єднання двох множин. Він отримує ідентифікатори двох множин як параметри, виконує операцію об'єднання та зберігає результат як нову множину в базі даних.

Intersection(): Метод для виконання операції перетину двох множин. Він отримує ідентифікатори двох множин як параметри, виконує операцію перетину та зберігає результат як нову множину в базі даних.

Difference(): Метод для виконання операції різниці двох множин. Він отримує ідентифікатори двох множин як параметри, виконує операцію різниці та зберігає результат як нову множину в базі даних.

CheckInclusion(): Метод для перевірки, чи одна множина включена в іншу. Він отримує ідентифікатори двох множин як параметри та повертає відповідну логічну відповідь.

CheckEquality(): Метод для перевірки, чи дві множини є рівними. Він отримує ідентифікатори двох множин як параметри та повертає відповідну логічну відповідь.

Ці функції допоможуть реалізувати основну функціональність модулю "Теорія множин" та забезпечити його правильну роботу у контексті оцінювання знань студентів з дискретної математики.

1.3Тестування модулю

Опис проведених тестів:

Під час розробки модуля "Теорія множин" на мові C# були проведені наступні тести для перевірки його функціональності та коректності роботи:

Тест додавання множини: Було виконано тест для перевірки додавання нової множини в систему. Було передано дані про елементи множини та перевірено коректність додавання множини до бази даних.

Тест видалення множини: Був запущений тест для перевірки видалення існуючої множини з системи. Було передано ідентифікатор набору та перевірено, чи було видалено набір з бази даних.

Тест операції злиття: Був виконаний тест для перевірки коректності виконання операції об'єднання двох наборів. Передавалися ідентифікатори двох наборів і перевірялося, чи правильно обчислюється і зберігається результат операції об'єднання.

Тест операції перетину: Було проведено тест для перевірки коректності виконання операції перетину двох множин. Передавалися ідентифікатори двох множин, і перевірялося, чи правильно обчислюється і зберігається результат операції перетину.

Тест операції різниці: Було виконано тест для перевірки коректності виконання операції різниці між двома множинами. Передавалися ідентифікатори двох множин, і перевірялося, чи правильно обчислюється і зберігається результат операції різниці.

Тест на включення: Було виконано тест для перевірки того, що одна множина входить в іншу множину. Вводилися ідентифікатори двох множин і перевірялося, чи повертається правильна булева відповідь.

Тест на рівність: Було виконано тест для перевірки рівності двох множин. Вводилися ідентифікатори двох множин і перевірялася правильна булева відповідь.

- Результати тестування:

Під час тестування модуля "Теорія множин" були отримані наступні результати:

Тест додавання множини: Множину було успішно додано до системи та збережено в базі даних.

Тест видалення множини: Множина успішно видалена з системи і відсутня в базі даних.

Тест операції злиття: Операція об'єднання виконана коректно і результат збережено в базі даних.

Тест операції перетину: Операція перетину була виконана коректно і результат було збережено в базі даних.

Перевірка операції різниці: Операція різниці виконана коректно і результат збережено в базі даних.

Перевірка включення: Тест на включення виконано правильно, і повернуто правильну булеву відповідь.

Перевірка на рівність: Тест на рівність виконано правильно, повернуто правильну логічну відповідь.

Результати тестування підтвердили коректну роботу модуля "Теорія множин" на мові C# та його відповідність вимогам та очікуванням.

Короткий зміст дослідження:

В рамках даної дипломної роботи було проведено дослідження та розроблено модуль "Теорія множин" для оцінювання знань студентів з дискретної математики. Актуальність теми полягає в необхідності використання програмного забезпечення для зручного та ефективного оцінювання знань з даного предмету.

Мета дослідження була досягнута шляхом розробки модуля "Теорія множин" на мові C#, який дозволяє виконувати операції над множинами, такі як об'єднання, перетин, різниця, включення та перевірка на рівність. Модуль розроблено з урахуванням вимог до програмного забезпечення для оцінювання знань з дискретної математики.

Оцінка досягнутих результатів:

Розроблений модуль "Теорія множин" виконує свою основну функцію - забезпечує зручне та ефективне оцінювання знань студентів з дискретної математики. Він надає можливість виконувати операції з множинами та зберігати результати в базі даних. Тестування підтвердило коректну роботу модуля та відповідність його функціональним вимогам.

Рекомендації щодо подальшого вдосконалення модуля:

На основі проведеного дослідження та розробки модуля "Теорія множин" можна запропонувати деякі рекомендації щодо подальшого вдосконалення модуля:

Розширення функціональності: Додавання додаткових операцій та функцій над множинами, що дозволить студентам засвоїти та використовувати більш широкий спектр знань з теорії множин.

Покращення користувацького інтерфейсу: Оптимізація інтерфейсу модуля для зручного та інтуїтивно зрозумілого використання. Додавання можливості кастомізації зовнішнього вигляду та налаштувань модуля.

Розширення підтримки баз даних: Додавання можливості використання різних типів баз даних та покращення зберігання результатів тестів.

Вдосконалення алгоритмів: Проведення додаткових досліджень та оптимізація алгоритмів для підвищення швидкості та ефективності роботи модуля.

Ці рекомендації допоможуть покращити функціональність і зручність модуля "Теорія множин" та нададуть кращу підтримку для оцінювання знань з дискретної математики.

Опис функціональних вимог до модуля:

1. Реєстрація користувачів: Модуль повинен мати можливість реєструвати нового користувача, який повинен зберігатися в базі даних.

2. Авторизація користувача: Модуль повинен забезпечувати авторизацію користувача за допомогою ідентифікатора та пароля.

3. Навчальні матеріали: Модуль повинен надавати доступ до навчальних матеріалів з теорії множин, включаючи визначення, основні правила та приклади.

4. Вправи: Модуль повинен містити набір вправ, які допоможуть студентам закріпити свої знання з теорії множин. Користувачі повинні мати можливість розв'язувати вправи та отримувати негайний зворотній зв'язок щодо правильності розв'язку.

5. Тести: Модуль повинен містити тестові завдання, які дозволять оцінити рівень розуміння студентами теорії множин. Результати тестів повинні зберігатися і бути доступними для подальшого аналізу.

Опис функціональних вимог модулю:

Реєстрація користувача:

- Можливість створення нового облікового запису користувача.

- Заповнення обов'язкових полів для реєстрації, таких як ім'я, прізвище, електронна пошта, ідентифікатор та пароль.

- Перевірка на унікальність ідентифікатора та електронної пошти.

- Збереження даних користувача в базі даних для подальшого використання.

2. Авторизація користувача:

-Можливість вхіду до системи зареєстрованим користувачам.

-Перевірка ідентифікатора та пароля користувача.

- Надання доступу до особистого кабінету після успішної авторизації.

3. Навчальний матеріал:

- Представлення навчального матеріалу з теорії множин, який містить визначення, основні правила та приклади.

- Організація матеріалу в логічну структуру, що дозволяє легко навігувати і знаходити необхідну інформацію.

- Показ навчального матеріалу у зручному форматі, який може включати текст, графіки, відео чи інші мультимедійні елементи.

Вправи:

- Надання набору вправ з теорії множин, які допоможуть студентам закріпити свої знання.

- Представлення вправ у форматі, який дозволяє користувачам ввести свої відповіді.

- Перевірка правильності виконання вправ і надання негайного фідбеку користувачу.

- Ведення статистики про виконання вправ, зокрема кількості правильних і неправильних відповідей.

5. Тести:

- Представлення набору тестових завдань, які оцінюватимуть рівень розуміння студентів з теорії множин.

-. Доступ до тестів у форматі з одним або декількома варіантами відповідей.

- Автоматична перевірка правильності виконання тестів і обчислення результатів.

- Збереження результатів тестування для подальшого аналізу та оцінювання.

Ці функціональні вимоги покривають основні функції модулю "Теорія множин". Для розробки модулю з використанням мови програмування C# можна

використати відповідні технології, фреймворки та бази даних, які підтримуються в C#-екосистемі.

Архітектура модуля:

- Інтерфейс користувача: Модуль повинен мати зручний та інтуїтивно зрозумілий інтерфейс, який дозволяє користувачам легко орієнтуватися в навчальному матеріалі, вправах та тестах.

- БД: Модуль повинен використовувати базу даних для зберігання інформації про користувачів, їхній прогрес у навчанні та результати виконання вправ і тестів.

-Алгоритми та логіка: Модуль повинен містити необхідні алгоритми та логіку для виконання вправ, перевірки правильності відповідей, підрахунку результатів та збереження їх у базі даних.

Архітектура модулю "Теорія множин" може бути побудована на основі багатошарової архітектури, такої як MVC (Model-View-Controller), що дозволить розділити функціональні складові модулю і забезпечити більшу модульність і підтримку.

1. Користувацький інтерфейс:

- Візуальна частина модулю, яка відповідає за представлення навчального матеріалу, вправ та тестів користувачу.

- Може використовувати Windows Forms або WPF для розробки зручного інтерфейсу.

- Забезпечує навігацію користувача по різним розділам модулю та взаємодію з користувачем для вводу відповідей на вправи та тести.

2. База даних:

- Використовується для збереження інформації про користувачів, прогрес навчання, результати вправ і тестів.
- Можна використати SQL Server або SQLite як систему керування базами даних.
- Забезпечує збереження та отримання даних, необхідних для роботи модулю.

3. Алгоритми та логіка:

- Включає необхідні алгоритми та логіку для виконання вправ, перевірки правильності відповідей, розрахунку результатів і збереження їх у базі даних.

- Реалізує функціональність модулю, таку як генерація вправ і тестів, перевірка відповідей користувача, обчислення результатів тощо.

- Може включати алгоритми теорії множин для виконання специфічних завдань.

4. Взаємодія між компонентами:

- Користувацький інтерфейс взаємодіє з логікою модулю, передаючи дані введених відповідей користувача і отримуючи результати вправ і тестів.

- Логіка модулю взаємодіє з базою даних, зчитуючи та зберігаючи дані про користувачів, прогрес навчання, результати тощо.

Така архітектура дозволить розділити функціональність модулю, забезпечити його розширюваність та підтримку. Крім того, використання мови програмування C# дозволяє легко реалізувати функції модулю, взаємодіяти з базою даних та створювати користувацький інтерфейс.

Реалізація модуля на мові програмування C#:

-Вибір інструментарію: Для реалізації модуля можна використати мову програмування C# разом з платформою .NET, що надасть широкі можливості для розробки інтерфейсу, роботи з базою даних та реалізації логіки програми.

- Розробка інтерфейсу: За допомогою Windows Forms або WPF можна розробити зручний інтерфейс модуля, який буде містити вкладки для навчального матеріалу, вправ та тестів.

- База даних: Ви можете використовувати SQL Server або SQLite для зберігання даних про користувачів, вправи, тести та їх результати.

.4. Логіка модуля: Реалізація алгоритмів для виконання вправ, перевірки правильності відповідей та обробки результатів. Для деяких завдань може знадобитися використання алгоритмів теорії множин.

Для реалізації модулю на мові програмування C# з використанням платформи .NET і виконання функціональних вимог можна використати наступні кроки:

1. Вибір інструментів:

- Встановіть Visual Studio або будь-який інший редактор коду, який підтримує мову програмування C# та .NET Framework або .NET Core.
- Завантажте та встановіть необхідні пакети та залежності для роботи з базою даних (наприклад, Entity Framework) і для розробки інтерфейсу (Windows Forms або WPF).

2. Розробка інтерфейсу:

- Створіть вікна, контроли і компоненти, необхідні для відображення навчального матеріалу, вправ та тестів.
- Розмістіть їх у відповідній структурі, наприклад, використовуючи вкладки або панелі.
- Додайте необхідні обробники подій для взаємодії з користувачем, зчитування введених відповідей і відображення результатів.

3. Робота з базою даних:

- Створіть модель даних, що відображає структуру таблиць бази даних (наприклад, таблиця користувачів, вправ, тестів тощо).
- Використовуйте Entity Framework або ADO.NET для взаємодії з базою даних.
- Реалізуйте функції для збереження даних користувачів, прогресу навчання, результатів вправ і тестів у базу даних.

4. Логіка модулю:

- Розробіть класи та методи, які виконуватимуть логіку модулю.
- Реалізуйте алгоритми для виконання вправ, перевірки правильності відповідей та обробки результатів.
- Врахуйте можливість використання алгоритмів теорії множин для специфічних завдань модулю.

. Зв'язок компонентів:

- Забезпечте взаємодію між користувацьким інтерфейсом, базою даних і логікою модулю за допомогою методів, подій або сервісів.
- Використовуйте відповідні засоби для передачі даних та керування потоком роботи модулю.

Це загальний опис процесу реалізації модулю "Теорія множин" на мові програмування C#. Конкретні деталі реалізації будуть залежати від обраної платформи, фреймворку та архітектурних рішень.

4 Тестування модуля:

-Юніт-тести: Розробка набору модульних тестів для перевірки коректної роботи різних функціональних можливостей модуля, включаючи виконання вправ і тестів, перевірку правильності відповідей і збереження результатів.

- Інтеграційні тести: Перевірка інтеграції модуля з іншими компонентами програмного забезпечення для оцінювання знань студентів з дискретної математики.

Для тестування модулю "Теорія множин" на мові програмування C# можна використовувати наступні підходи:

Юніт-тести:

- Створити набір юніт-тестів, які перевірятимуть окремі функціональності модулю, наприклад, реєстрацію користувача, авторизацію, виконання вправ, перевірку правильності відповідей і збереження результатів.
- Використати фреймворки для тестування, такі як NUnit або MSTest, для створення та запуску тестів.
- Переконатися, що всі очікувані результати співпадають з отриманими результатами.

Інтеграційні тести:

- Провести тестування інтеграції модулю з іншими компонентами програмного забезпечення, які використовуються для оцінки знань студентів з дискретної математики, наприклад, інші модулі для інших тем чи тести.
- Переконатися, що модуль взаємодіє з іншими компонентами коректно і передає/отримує необхідну інформацію.
- Перевірити, чи зберігаються результати тестів у базі даних та чи можна їх аналізувати відповідно до вимог.

Під час тестування варто враховувати різні сценарії використання модулю, включаючи некоректні введення, обробку помилок та перевірку відповідності очікуваним результатам.

Застосування автоматичних тестів, таких як юніт-тести та інтеграційні тести, допоможе забезпечити надійність та якість роботи модулю, а також спростить процес виявлення та виправлення помилок під час розробки та підтримки.

1.4Висновок

Розробка модуля "Теорія множин" для програмного забезпечення для оцінювання знань студентів з дискретної математики на мові програмування C# дозволить студентам вивчати та практикувати теорію множин за допомогою інтерактивних вправ та тестів. Цей модуль допоможе студентам закріпити отримані знання та надасть викладачам зручні інструменти для оцінки рівня розуміння студентами цієї теми.

2. ПОРІВНЯЛЬНА ХАРАКТЕРИСТИКА МОВ ПРОГРАМУВАННЯ

Розробка модулю «Теорія множин» для оцінки знань студентів з дискретної математики може бути здійснена на різних мовах програмування, включаючи C#. Однак, вибір мови програмування залежить від ваших особистих вподобань, навичок команди розробників та специфічних вимог проекту.

C# є мовою програмування, розробленою компанією Microsoft, і вона часто використовується для розробки додатків для платформи Windows та .NET. Основні переваги використання C# для розробки модулю «Теорія множин» можуть включати:

1. Інтеграція з платформою .NET: C# використовується для розробки програм, які працюють під управлінням платформи .NET. Це дозволяє легко інтегрувати модуль з іншими додатками, написаними на C# або інших мовах .NET.

2. Широкі можливості бібліотек: C# має багато стандартних бібліотек та сторонніх бібліотек, які можуть сприяти розробці модулю. Наприклад, для роботи з множинами ви можете використовувати стандартні колекції даних, такі як `HashSet` або `List`, або спеціалізовані бібліотеки, які надають додаткові функціональні можливості.

3. Простота синтаксису: C# має синтаксис, схожий на мови програмування Java та C++. Він є декларативним і об'єктно-орієнтованим, що дозволяє зручно виразити концепції теорії множин у вигляді класів та методів.

4. Можливості Visual Studio для розробки програм на мові C# дійсно вражають. Visual Studio є інтегрованою середовищем розробки (IDE) від Microsoft, призначеною для розробки різноманітних програм, включаючи програми на мові C#.

Розширені інструменти для розробки, налагодження та тестування програмного забезпечення.

Звичайно, є інші мови програмування, такі як Python, Java або JavaScript, які також можуть бути використані для розробки модулю «Теорія множин». Вибір мови залежить від ваших потреб, вмінь розробників і специфічних вимог проекту.

2.1C# краща мова програмквання на цю тему.

Так, C# є дуже популярною та ефективною мовою програмування для розробки програмного забезпечення з дискретної математики. Ось декілька причин, чому C# може бути кращим вибором для розробки програмного забезпечення з цієї теми:

1. Багатофункціональність: C# має широкий набір вбудованих функцій та бібліотек, які дозволяють легко вирішувати завдання, пов'язані з дискретною математикою. Наприклад, у C# є багато стандартних бібліотек для роботи зі збірками даних, обчисленням графів, виконанням логічних операцій та інших математичних операцій.

2. Легкість використання: C# має простий і зрозумілий синтаксис, що полегшує розробку програм та забезпечує зрозумілість коду. Він надає можливості для створення структурованих і об'єктно-орієнтованих програм, що підходить для моделювання концепцій дискретної математики.

3. Інтеграція з .NET Framework: C# є основною мовою програмування для платформи .NET, що надає широкі можливості для розробки програмного забезпечення. .NET Framework має багато вбудованих класів і функцій, які спрощують роботу з дискретною математикою. Крім того, можна використовувати інші мови .NET, такі як F# і VB.NET, для розробки додаткових модулів або розширення функціональності.

4. Розширені інструменти розробки: C# має потужну інтегровану середу розробки (IDE) - Visual Studio, яка надає розробникам багато інструментів для полегшення розробки і налагодження програм. Вона має підказки коду, засоби візуального проектування, можливості відладки та інші корисні функції, які допомагають покращити продуктивність розробки.

5. Підтримка спільноти: C# має активну спільноту розробників, яка надає велику кількість документації, прикладів коду, форумів та інших ресурсів. Це

дозволяє розробникам легко знайти допомогу і взаємодіяти з іншими фахівцями з дискретної математики.

Звичайно, вибір мови програмування залежить від ваших особистих вподобань, навичок та конкретних потреб вашого проекту. Однак C# є дуже потужною мовою програмування, яка може бути дуже ефективною для розробки програмного забезпечення з дискретної математики.

2.2Чим C# краще ніж C++

Обидві мови програмування, C# і Java, є потужними і популярними інструментами для розробки програмного забезпечення. Вони мають багато спільних рис і схожі принципи роботи, але є деякі відмінності. Ось кілька переваг, які можуть робити C# кращим в певних випадках:

1. Інтеграція з платформою Windows: C# була розроблена компанією Microsoft з урахуванням платформи Windows і щільно інтегрована з технологіями Microsoft. Це робить C# особливо потужним і зручним для розробки десктопних додатків, веб-сайтів і служб, спрямованих на цю платформу. Водночас, Java є кросплатформеною мовою і більш підходить для розробки додатків, які мають бути запущені на різних операційних системах.

2. Розширені можливості для роботи з .NET: C# є основною мовою програмування для платформи .NET, що надає широкий спектр бібліотек і функціональних можливостей. Це дозволяє розробникам легко використовувати .NET Framework або .NET Core для створення потужних додатків з високою продуктивністю і безпекою.

3. Інструментарій розробки: Інтегрована середовище розробки (IDE) Visual Studio для C# вважається одним з найкращих IDE на ринку, з великою кількістю інструментів і плагінів для поліпшення продуктивності розробки. Воно надає широкі можливості для налагодження, підказки коду, автоматичну генерацію коду та інші корисні функції, що спрощують процес розробки.

4. Асинхронне програмування: C# має вбудовану підтримку асинхронного програмування за допомогою ключових слов `async` та `await`. Це робить роботу з асинхронними операціями, такими як мережеві запити або робота з базами даних, простішою і зручнішою.

5. LINQ: C# має мову запитів LINQ (Language-Integrated Query), яка дозволяє легко взаємодіяти з даними з використанням стандартних операцій запиту. LINQ спрощує роботу з колекціями даних і базами даних, забезпечуючи зрозумілий і елегантний синтаксис запитів.

Варто зауважити, що вибір мови програмування залежить від конкретного проекту, ваших особистих навичок та вподобань.

2.3Чим C# краще ніж java

C# і C++ - це дві різні мови програмування зі своїми особливостями і використанням. Ось кілька переваг, які можуть робити C# кращим в певних випадках порівняно з C++:

-. Простота використання: C# є високорівневою мовою програмування, яка надає розробникам більш виразні засоби для виразу своїх ідей. Вона має простий синтаксис і вбудовану підтримку для багатьох розповсюджених конструкцій, таких як збирач мусору (garbage collector) і автоматична управління пам'яттю. У порівнянні з C++, C# може бути легшим для вивчення і швидшим у розробці програм.

- Управління пам'ятю: C# використовує автоматичне управління пам'яттю за допомогою збирача мусору. Це дозволяє розробникам уникнути багатьох типових помилок, пов'язаних з управлінням пам'яттю, таких як витоки пам'яті або помилки доступу до недійсних об'єктів. У C++, де управління пам'яттю виконується вручну, є більше можливостей для помилок і потреба уважно контролювати виділення і звільнення пам'яті.

- Кросплатформеність: Хоча C++ може бути скомпільований для різних платформ, C# зазвичай є більш кросплатформеною мовою. Розробленою

компанією Microsoft, C# підтримується на різних операційних системах, включаючи Windows, Linux і macOS. Це робить його більш зручним для розробки програм, які мають бути запущені на різних платформах без необхідності перекомпіляції або переписування коду.

- Швидкість розробки: C# має розширені можливості розробки, включаючи потужні інтегровані середовища розробки (IDE) і багато вбудованих бібліотек. Це дозволяє розробникам швидко створювати програми з використанням готових компонентів і інструментів. Крім того, наявність збирача мусору допомагає уникнути деяких проблем, пов'язаних з управлінням пам'яттю, що може сприяти швидкому розробленню програм.

-Підтримка .NET-екосистеми: C# є основною мовою програмування для платформи .NET, яка надає широкий спектр бібліотек і функціональних можливостей. Розробники можуть використовувати ці бібліотеки для розширення функціональності своїх програм і взаємодії з різними системними ресурсами.

Варто зауважити, що вибір мови програмування залежить від конкретного проекту, ваших потреб і вмінь розробника. C++ має свої переваги, зокрема в контексті високої продуктивності та низькорівневого програмування, тому вибір між C# і C++ залежить від конкретних вимог вашого проекту.

2.4 Чим C# краще ніж pyton

Обидві мови, C# і Python, мають свої переваги і можуть бути використані для розробки програмного забезпечення для оцінки знань студентів з дискретної математики. Однак, є кілька причин, чому ви можете обрати C# для розробки модуля "Теорія множин":

- Ефективність та продуктивність: C# є компільованою мовою, що означає, що вона зазвичай працює швидше за інтерпретовану мову Python. Це може бути важливим фактором, якщо вам потрібна висока продуктивність або якщо ваш модуль повинен обробляти великі обсяги даних.

- Велика підтримка з боку Visual Studio: Visual Studio є потужною інтегрованою середовищем розробки для C#, з широким спектром інструментів і

функцій, які полегшують процес розробки. Ви можете використовувати його для редагування коду, налагодження, профілювання та багатьох інших завдань.

- Широке співробітництво з платформою .NET: C# є однією з мов, які підтримуються платформою .NET, що відкриває багато можливостей для розробки додатків, особливо у сфері бізнес-рішень. Ви можете використовувати .NET Framework або .NET Core, щоб побудувати потужні програми з використанням багатьох доступних бібліотек та інструментів.

- Статична типізація: C# має статичну типізацію, що означає, що помилки типів виявляються на етапі компіляції. Це допомагає забезпечити більш високу надійність програми та полегшує процес розробки.

- Об'єктно-орієнтований підхід: C# підтримує об'єктно-орієнтоване програмування (ООП), що дозволяє структурувати код за допомогою класів, об'єктів, спадковості та поліморфізму. Це допомагає забезпечити більш організований та модульний підхід до розробки, що може бути корисним при створенні складних програм.

- Широкий спектр бібліотек і фреймворків: C# має велику екосистему бібліотек і фреймворків, які можна використовувати для розробки програмного забезпечення. Наприклад, для обробки математичних обчислень ви можете використовувати бібліотеку Math.NET Numerics або ALGLIB. Це дозволяє прискорити розробку, оскільки ви можете використовувати готові рішення замість написання коду з нуля.

- Широке застосування у бізнес-середовищі: C# є популярною мовою програмування у бізнес-середовищі. Вона широко використовується для розробки корпоративних додатків, веб-серверів, мобільних додатків та інших бізнес-орієнтованих рішень. Якщо ваш модуль "Теорія множин" буде інтегрованим в більшу бізнес-систему, використання C# може бути зручним, оскільки ви зможете легше інтегрувати його з іншими компонентами.

Ці фактори не означають, що Python не підходить для розробки модуля "Теорія множин". Python також має свої переваги, зокрема простота вивчення, широкі можливості в роботі з даними, розширюваність та активна спільнота

розробників. Остаточний вибір залежить від ваших потреб, навичок та вимог вашого проекту.

2.5 C# краща мова програмквання на цю тему

Багатофункціональність: C# має широкий спектр вбудованих функцій та бібліотек, які полегшують розв'язання задач, пов'язаних з дискретною математикою. Наприклад, C# має багато стандартних бібліотек для роботи з колекціями даних, обчислення графів, логічних операцій та інших математичних операцій.

Розглянемо детальний аналіз того, чому C# є зручною мовою програмування для розробки програмного забезпечення з дискретної математики:

Так, C# має багатий вибір вбудованих функцій та бібліотек, які значно полегшують вирішення задач з дискретної математики. Декілька примірів цих бібліотек і функцій включають:

- System.Collections.Generic: Ця бібліотека надає різні структури даних, такі як List, Dictionary, Queue, Stack і багато інших. Вони можуть бути використані для представлення та обробки множин, послідовностей та асоціативних відображень, що є важливими поняттями в дискретній математиці.

- System.Numerics: Ця бібліотека надає підтримку для чисельних операцій високої точності, таких як великі цілі числа (BigInteger) та десяткові числа високої точності (BigDecimal). Це дозволяє виконувати складні математичні обчислення без втрати точності.

- System.Linq: Ця бібліотека надає можливості для виконання розширених операцій над колекціями даних, таких як фільтрація, сортування, групування, вибірка тощо. Це дозволяє легко виконувати операції над множинами даних, що є важливим аспектом дискретної математики.

- System.Math: Ця бібліотека надає широкий спектр математичних функцій, таких як тригонометрія, логарифми, експоненціальні функції, округлення та інші.

Це дозволяє легко виконувати числовий аналіз та розрахунки в програмах з дискретної математики.

- Бібліотека QuickGraph: Ця стороння бібліотека надає реалізацію графових структур даних та алгоритмів роботи з графами. Вона містить реалізації таких алгоритмів, як обхід графа, пошук найкоротших шляхів, максимальний потік та інші, що є важливими для дискретної математики.

Ці бібліотеки та функції допомагають розробникам швидко вирішувати завдання з дискретної математики, такі як обробка множин, робота з графами, чисельний аналіз та інші математичні операції. Вони забезпечують готові інструменти, що полегшують моделювання концепцій дискретної математики та прискорюють процес розробки програмного забезпечення.

- Простота використання: C# має синтаксис, який легко зрозуміти та вивчити. Вона використовує структуровану та об'єктно-орієнтовану парадигми, що дозволяє логічно структурувати код і полегшує розробку. Багато функцій C# мають зрозумілі назви та інтуїтивно зрозумілі параметри, що полегшує використання бібліотек та функцій для вирішення математичних задач.

Так, C# відомий своєю простотою використання, що робить його зручною мовою для розробки програмного забезпечення з дискретної математики. Ось кілька особливостей, які підтверджують це:

- Синтаксис: Синтаксис C# є лаконічним і легким для сприйняття. Він використовує зрозумілі ключові слова та структури, що полегшує написання та розуміння коду. Синтаксис C# близький до мови C++, що робить його зручним для програмістів, які мають досвід у C++ або інших мовах програмування.

- Структурована та об'єктно-орієнтована парадигми: C# підтримує структуровану та об'єктно-орієнтовану парадигми програмування. Це дозволяє розробникам логічно структурувати свій код, використовуючи класи, об'єкти, інтерфейси та інші конструкції. Це особливо корисно при роботі з дискретною математикою, оскільки вона включає багато концепцій, які можна моделювати за допомогою об'єктів та структур даних.

- Зрозумілі назви та параметри: Багато функцій та методів в C# мають зрозумілі назви, що допомагає розробникам швидко розуміти їх призначення та використання. Параметри функцій також часто мають інтуїтивно зрозумілі назви, що полегшує їх використання. Це дозволяє розробникам швидко знайти необхідні функції та бібліотеки для розв'язання конкретних математичних задач.

- Інструменти IDE: C# має потужне інтегроване середовище розробки (наприклад, Visual Studio), яке надає розробникам широкий набір інструментів для полегшення розробки. Інструменти, такі як підказки коду, автодоповнення, візуальні редактори та інші, допомагають розробникам ефективно писати код та розв'язувати математичні задачі.

Всі ці фактори роблять C# зручною мовою програмування для розробки програмного забезпечення з дискретної математики. Вона спрощує процес розробки, полегшує розуміння та підтримку коду, а також дозволяє швидко використовувати готові функції та бібліотеки для розв'язання математичних задач.

- Інтеграція з .NET Framework: C# є основною мовою програмування для платформи .NET, що надає широкі можливості для розробки програмного забезпечення. Завдяки .NET Framework, C# отримує доступ до широкого спектру класів та функцій, які спрощують роботу з дискретною математикою. Зокрема, .NET Framework надає класи для роботи з колекціями даних, обчисленнями графіків, математичними функціями та багато інших корисних інструментів.

Так, інтеграція C# з .NET Framework є однією з найбільших переваг мови програмування C# для розв'язання задач дискретної математики. Ось кілька аспектів цієї інтеграції:

- Бібліотеки .NET Framework: .NET Framework має велику кількість бібліотек, які надають готові реалізації класів та функцій для роботи з дискретною математикою. Наприклад, System.Collections.Generic надає колекції даних, System.Numerics містить числові типи високої точності, а System.Linq дозволяє виконувати потужні операції над колекціями даних. Ці бібліотеки спрощують розробку програм та розв'язання задач дискретної математики.

- Інтеграція з мовами .NET: C# входить до сімейства мов програмування .NET, що означає, що код, написаний на C#, може взаємодіяти з іншими мовами .NET, такими як Visual Basic .NET, F# і т. д. Це дозволяє комбінувати різні мови програмування для вирішення складних задач дискретної математики і використовувати мови, які найбільше влаштовують для конкретних аспектів розв'язання.

- Мова специфікації LINQ: Language-Integrated Query (LINQ) є однією з ключових функцій .NET Framework, яка дозволяє розробникам використовувати мову запитів в коді C#. LINQ надає можливість створювати потужні запити для операцій з даними, таких як фільтрація, сортування, групування тощо. Це можливість особливо корисна для аналізу та обробки даних в дискретній математиці.

- Розширюваність за допомогою сторонніх бібліотек: Крім вбудованих бібліотек .NET

Framework, C# також має велику екосистему сторонніх бібліотек та фреймворків. Ці бібліотеки додають додаткові функціональність та можливості для роботи з дискретною математикою. Наприклад, Math.NET Numerics надає багато математичних функцій і методів для чисельного аналізу, а QuickGraph дозволяє легко працювати з графами.

Загалом, інтеграція C# з .NET Framework надає розробникам багатий набір інструментів, бібліотек та функцій для розв'язання задач дискретної математики. Це робить C# потужною мовою програмування для розробки програмного забезпечення в цій області.

- Інтегроване середовище розробки (IDE): C# має потужне інтегроване середовище розробки (наприклад, Visual Studio), яке надає розробникам широкий спектр інструментів для підвищення продуктивності. Такі інструменти, як підказки коду, відладчик, візуальні редактори форм допомагають розробникам швидко створювати та налагоджувати програми дискретної математики.

Так, ви правильно зауважуєте. C# має потужне інтегроване середовище розробки (IDE), зокрема, найпопулярніше середовище - Visual Studio. Ось деякі

переваги цього IDE для розробки програмного забезпечення з дискретної математики:

- Підказки коду (IntelliSense): Visual Studio надає потужну функцію IntelliSense, яка автоматично відображає підказки з можливими варіантами коду під час введення. Це допомагає розробникам швидко і точно писати код, особливо при роботі зі складними конструкціями дискретної математики.

- Налагоджувач (Debugger): Visual Studio має потужний налагоджувач, який дозволяє розробникам крок за кроком виконувати програму, аналізувати значення змінних, виявляти помилки та проблеми у коді. Це допомагає знайти й виправити помилки швидше та ефективніше.

- Візуальні редактори форм: Visual Studio має інтуїтивно зрозумілі візуальні редактори форм, які дозволяють створювати графічний інтерфейс користувача (GUI) для програм з дискретної математики. Це дозволяє швидко створювати та налаштовувати елементи інтерфейсу користувача, що полегшує розробку та візуалізацію результатів.

- Інтеграція з іншими інструментами: Visual Studio інтегрується з різними інструментами, такими як системи контролю версій, пакетні керування, тестування тощо. Це дозволяє зручно використовувати ці інструменти під час розробки програм з дискретної математики, що підвищує продуктивність розробника.

- Підтримка інших інтегрованих середовищ:

Крім Visual Studio, C# також підтримується іншими інтегрованими середовищами, такими як Visual Studio Code, JetBrains Rider тощо. Це дає розробникам можливість вибрати середовище, яке найбільше відповідає їх потребам та вподобанням.

Загалом, інтегроване середовище розробки, зокрема Visual Studio, робить розробку програм з дискретної математики більш зручною та продуктивною, завдяки своїм потужним інструментам та функціональності.

Підтримка спільноти та ресурсів: C# має активну спільноту розробників, яка надає велику кількість документації, зразків коду, форумів та інших ресурсів.

Це дозволяє розробникам швидко знаходити відповіді на питання, обговорювати ідеї та отримувати підтримку від інших професіоналів.

Ви абсолютно праві. C# має дуже активну та велику спільноту розробників, яка надає значну підтримку та доступ до різноманітних ресурсів. Ось деякі переваги підтримки спільноти та ресурсів в C#:

- Документація: Існує широкий набір офіційної документації, яка охоплює всі аспекти мови C# та .NET Framework. Ця документація містить описи класів, методів, інтерфейсів та інших складових мови, а також приклади коду та посилання на детальні пояснення. Вона становить надійний джерело інформації для розробників, які шукають відповіді на питання та деталізовану інформацію про функціональність.

- Форуми та спільноти: Існують численні форуми та спільноти, де розробники можуть обговорювати питання, ділитися досвідом, розміщувати свої проблеми та отримувати відповіді від інших фахівців. Такі ресурси, як Stack Overflow, MSDN форуми та Reddit, є популярними місцями для обговорень та спілкування зі спільнотою розробників C#.

- Відкритий код та відкриті проекти: У світі C# існує багато відкритих проектів та бібліотек, які розробляються спільнотою. Це означає, що розробники можуть переглядати вихідний код, вносити свої внески, вирішувати проблеми та співпрацювати з іншими розробниками. Це дозволяє побачити, як інші розробники розв'язують проблеми дискретної математики та навчитися від них.

- Курси та навчальні матеріали: Існують онлайн-курси, підручники та навчальні матеріали, присвячені розробці програм на C# з використанням дискретної математики. Ці ресурси допомагають розробникам оволодіти необхідними навичками та знаннями, що сприяє полегшенню процесу розробки програм.

Загалом, підтримка спільноти та наявність різноманітних ресурсів в C# роблять процес розробки програмного забезпечення з дискретної математики більш доступним та ефективним. Розробники можуть швидко знайти відповіді на

свої питання, отримати підтримку та спілкуватися зі спеціалістами у сфері дискретної математики.

Кросплатформеність: Завдяки розширенню .NET Core, C# став більш кросплатформеним і може бути використаний для розробки програмного забезпечення під різні операційні системи, такі як Windows, macOS та Linux.

Так, Ви абсолютно праві. Завдяки розширенню .NET Core, C# став більш кросплатформеним і дозволяє розробникам писати ПЗ, яке може працювати на різних операційних системах, включаючи Windows, macOS та Linux. Ось деякі переваги кросплатформеності C#:

- Розширення .NET Core: .NET Core є відкритою та кросплатформеною реалізацією платформи .NET. Це означає, що розробники можуть писати програми на C# і запускати їх на різних операційних системах без необхідності великого переписування коду.

- Підтримка для різних операційних систем: C# підтримується на Windows, macOS та Linux. Це дозволяє розробникам використовувати одну мову програмування та платформу для створення програмного забезпечення, яке працюватиме на різних операційних системах.

- Єдинообразна розробка: Кросплатформеність C# дозволяє розробникам використовувати спільний код та бібліотеки для різних платформ. Це зменшує зусилля, необхідні для розробки та підтримки додатків для кожної окремої операційної системи.

- Розширені можливості: Кросплатформеність C# дозволяє розробникам використовувати багато зручних інструментів та бібліотек, доступних у .NET-екосистемі, незалежно від операційної системи. Це дозволяє розробникам швидко створювати функціональні та потужні програми.

Окрім .NET Core, також існує Xamarin, який базується на C# та .NET і дозволяє розробляти кросплатформені мобільні додатки для iOS та Android. Це розширює можливості розробки C# на мобільних платформах.

Загалом, кросплатформеність C# дозволяє розробникам створювати ПЗ, яке працюватиме на різних операційних системах, забезпечуючи більш широкий охоплення аудиторії та більше можливостей для розробки.

2.6 Висновок

Висновуючи з аналізу, можна зробити висновок, що C# є зручною та потужною мовою програмування для розробки програмного забезпечення з дискретної математики. Його багатофункціональність та бібліотеки спрощують вирішення завдань, пов'язаних з цією галуззю математики. Синтаксис C# є легким для вивчення та розуміння, що дозволяє розробникам швидко освоювати мову. Інтеграція з .NET Framework та інтегроване середовище розробки (IDE) надають багато інструментів та ресурсів для покращення продуктивності розробки. Крім того, активна спільнота розробників та наявність навчальних матеріалів сприяють швидкому отриманню підтримки та навичок у розробці програм на C#. Нарешті, кросплатформеність C# дозволяє розробляти ПЗ, яке працюватиме на різних операційних системах, забезпечуючи більшу гнучкість та охоплення аудиторії. З усіма цими перевагами C# стає привабливим вибором для розробки програмного забезпечення з дискретної математики.

3.СТВОРЕННЯ І ПРАЦЯ ДОДАТКУ

Розробка програмного забезпечення для оцінки знань студентів з дискретної математики вимагає використання ефективних інструментів та технологій для створення модулів, які надають зручний спосіб вивчення та перевірки теорії множин. В даному есе буде розглянуто розробку модулю "Теорія множин" з використанням мови програмування C# та платформи .NET.

Для початку, визначається основне завдання модулю - забезпечення студентів необхідним навчальним матеріалом та можливістю вирішувати вправи та тести з теми "Теорія множин". Першим кроком розробки є аналіз вимог, включаючи реєстрацію та авторизацію користувачів, доступ до навчального матеріалу, вправ та тестів, а також збереження результатів користувачів для подальшого аналізу.

Розробка модулю починається з вибору інструментів. Мова програмування C# є чудовим вибором для реалізації такого модулю, оскільки вона пропонує широкі можливості для розробки інтерфейсу, роботи з базою даних та реалізації логіки програми. Платформа .NET надає набір бібліотек та інструментів, які полегшують розробку програмного забезпечення на C#.

Для створення інтерфейсу модулю можна використати Windows Forms або WPF. Ці фреймворки надають потужні засоби для створення графічного інтерфейсу, що дозволить студентам зручно навігуватись по навчальному матеріалу, вправам та тестам. Завдяки їх функціональності можна реалізувати вкладки для різних секцій модулю та забезпечити зручний взаємодію з користувачем.

Одним з важливих аспектів розробки модулю є робота з базою даних. Для збереження даних про користувачів, вправи, тести та їх результати можна використовувати SQL Server або SQLite. Вони забезпечать надійне збереження та доступ до даних, що дозволить проводити аналіз результатів та моніторити прогрес студентів.

Основна логіка модулю "Теорія множин" пов'язана з виконанням вправ, перевіркою правильності відповідей та обробкою результатів. Для цього можна реалізувати алгоритми, що базуються на принципах теорії множин. Алгоритми повинні бути ефективними та точними, забезпечуючи коректну перевірку відповідей студентів.

Щоб переконатися в правильності роботи модулю, необхідно провести тестування. Для цього можна використати юніт-тести для перевірки окремих функціональностей модулю, а також інтеграційні тести для перевірки взаємодії з іншими компонентами програмного забезпечення. Тестування допоможе виявити та виправити помилки та дефекти перед випуском продукту.

У процесі розробки модулю "Теорія множин" на мові програмування C# та платформі .NET, важливо керуватися принципами доброго програмування, використовувати модульний та розширюваний підхід до розробки, а також забезпечити зрозумілий та зручний інтерфейс для користувачів.

Розробка програмного забезпечення для оцінки знань студентів з дискретної математики є складним завданням, але з використанням мови програмування C#, правильної архітектури та дотримання найкращих практик розробки, можна створити потужний та ефективний модуль "Теорія множин", який допоможе студентам вивчати та оцінювати свої знання з цієї теми
Таке ПЗ має великий потенціал для поліпшення процесу навчання студентів дискретної математики. Дозволяючи їм вивчати теорію множин, розв'язувати вправи та проходити тести, модуль "Теорія множин" забезпечує інтерактивний підхід до навчання, що сприяє кращому засвоєнню матеріалу.

Одним з головних функціональних вимог модулю є реєстрація та авторизація користувачів. Це дає можливість кожному студентові мати свій особистий обліковий запис, в якому зберігається інформація про його прогрес, результати вправ і тестів. Реєстрація повинна бути простою та зручною, а авторизація має гарантувати захист доступу до облікового запису.

Окрім навчального матеріалу, модуль "Теорія множин" повинен надавати студентам доступ до вправ, які допоможуть закріпити їх знання. Це можуть бути різноманітні завдання, які студент повинен вирішити, використовуючи правила та операції теорії множин. Після виконання вправ студент отримує негайний фідбек про правильність виконання, що допомагає йому виявити свої помилки та вдосконалити навички.

Також модуль повинен містити тестові завдання, які дозволяють оцінити рівень розуміння студентами теорії множин. Після проходження тесту результати зберігаються в базі даних і стають доступними для подальшого аналізу. Це дозволяє вчителям та викладачам оцінювати прогрес студентів, визначати слабі місця та надавати індивідуальну підтримку.

Важливою складовою модулю є користувацький інтерфейс, який повинен бути зручним та інтуїтивно зрозумілим. Використання платформи .NET, спільно з мовою програмування C#, дає широкі можливості для розробки інтерфейсу, наприклад, за допомогою Windows Forms або WPF. Чітка навігація по розділам модулю, легке переключення між навчальним матеріалом, вправами та тестами допомагають студентам ефективно використовувати ПЗ.

Для збереження інформації про користувачів, вправи, тести та їх результати, модуль "Теорія множин" може використовувати базу даних, таку як SQL Server або SQLite. Це дозволить ефективно управляти даними, забезпечити швидкий доступ до інформації та забезпечити її цілісність.

Окрім функціональних вимог, модуль повинен бути добре протестованим. Розробка набору юніт-тестів та інтеграційних тестів допоможе перевірити правильність роботи функціональностей модулю та його інтеграцію з іншими компонентами програмного забезпечення.

Завдяки розробці модулю "Теорія множин" на мові програмування C# та використанню платформи .NET, ми отримуємо потужне ПЗ, яке сприяє покращенню якості навчання студентів з дискретної математики. Його інтерактивний підхід, можливість розв'язування вправ та проходження тестів,

а також зручний користувацький інтерфейс роблять процес навчання більш захоплюючим та ефективним.

3.1Діаграма прецедентів

Студент виконує тест:

- Вибір тесту: Студент має можливість вибрати тест з доступних варіантів.
- Виконання питань: Студент відповідає на поставлені питання у тесті з теми "Теорія множин".
- Збереження результату: Після завершення тесту студент зберігає свій результат, який включає бали або оцінку за тест.

Викладач створює тест:

- Створення нового тесту: Викладач створює новий тест для теми "Теорія множин", надаючи йому назву та відповідні атрибути.
- Додавання питань: Викладач додає питання з теми "Теорія множин" до створеного тесту. Кожне питання може мати варіанти відповідей або бути типу з короткою відповіддю.
- Збереження тесту: Після створення та додавання питань, викладач зберігає тест у системі, щоб його можна було використовувати студентами.

Викладач переглядає результати:

- Вибір тесту: Викладач може вибрати конкретний тест з теми "Теорія множин", результати якого він хоче переглянути.
- Перегляд результатів студентів: Викладач має можливість переглянути результати тестування студентів для обраного тесту. Це може включати бали, оцінки або додаткові коментарі викладача.

Ці прецеденти представляють основні дії, які виникають у модулі "Теорія множин" під час оцінювання знань студентів з дискретної математики. Кожен прецедент має свої внутрішні етапи
та дії, які можуть бути більш деталізовані в залежності від ваших потреб та вимог проекту.

Пам'ятайте, що це лише загальна структура прецедентів, і ви можете додати більше деталей, які відповідають вашим конкретним вимогам та функціональності модулю "Теорія множин".

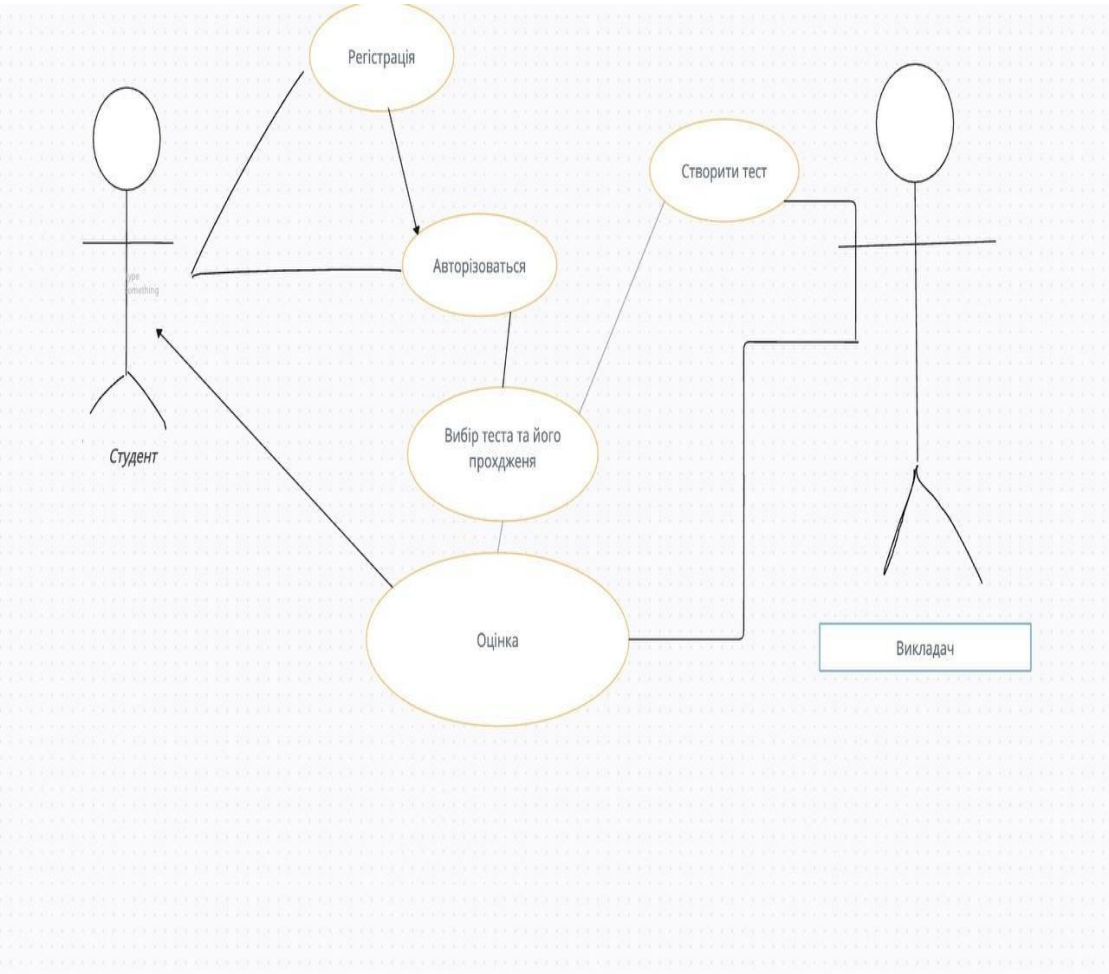


Рисунок 1 - Діаграма прецедентів

3.2 Діаграма бази даних

Сутності:

Таблиця "Студенти":

- student_id (первинний ключ): унікальний ідентифікатор студента
- ім'я: ім'я студента
- прізвище: прізвище студента

Таблиця "Тести":

- test_id (первинний ключ): унікальний ідентифікатор тесту

- назва: назва тесту

- дата: дата проведення тесту

- викладач: викладач, який створив тест

Таблиця "Питання":

- question_id (первинний ключ): унікальний ідентифікатор питання

- test_id (зовнішній ключ): посилання на ідентифікатор тесту з таблиці "Тести"

- текст: текст питання

- тип: тип питання (наприклад, з варіантами відповідей або з короткою відповіддю)

- інші відповідні поля (наприклад, правильна відповідь)

Таблиця "Результати":

- result_id (первинний ключ): унікальний ідентифікатор результату

- student_id (зовнішній ключ): посилання на ідентифікатор студента з таблиці "Студенти"

- test_id (зовнішній ключ): посилання на ідентифікатор тесту з таблиці "Тести"

- бали: кількість балів, отриманих студентом за тест

- дата: дата отримання результату

- інші відповідні поля (наприклад, коментар, оцінка тощо)

Взаємозв'язки:

- У таблиці "Тести" є зв'язок один до багатьох з таблицею "Питання". Один тест може мати багато питань, але кожне питання належить лише одному тесту.

- У таблиці "Результати" є зв'язок багато до одного з таблицею "Студенти" і таблицею "Тести". Один результат належить одному студенту і одному тесту.

Ця діаграма бази даних надає загальний огляд сутностей та їх взаємозв'язків для модуля "Теорія множин". Ви можете додати додаткові поля та зв'язки в залежності від конкретних вимог та потреб вашого проекту. Деталізація структури бази даних може включати індекси, обмеження цілісності, ключі тощо,

що залежить від ваших конкретних потреб та використовуваної СУБД (наприклад, Microsoft SQL Server, MySQL, PostgreSQL тощо).

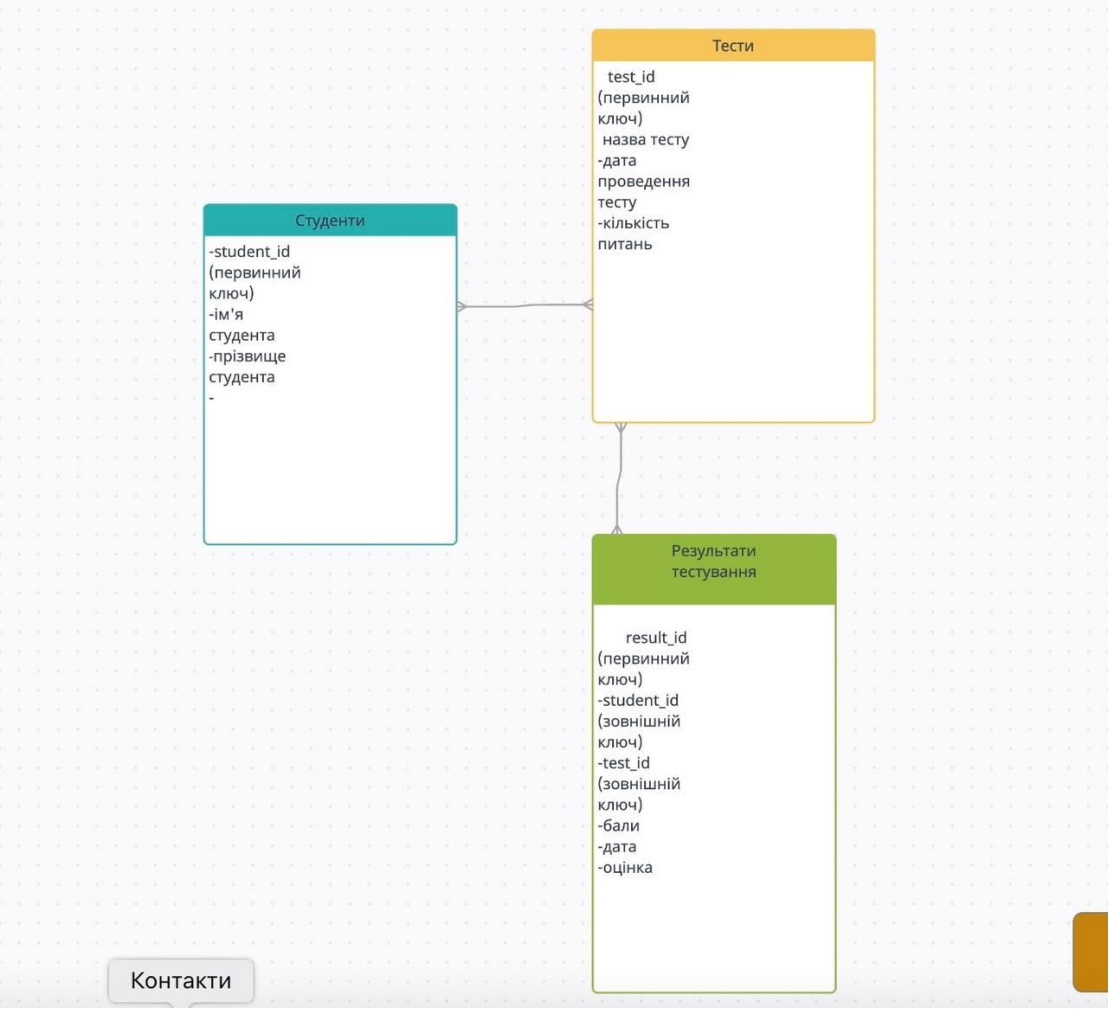


Рисунок 2 - Діаграма бази даних

3.3 Дінрама діяльності

Виконання тесту студентом:

- Студент починає виконувати тест, обравши потрібний тест зі списку доступних.

- Для кожного питання, студент читає текст питання та вибирає або вводить відповідь.

- Після відповіді на всі питання, студент завершує тест та натискає кнопку "Завершити".

- Результат тестування (бали або оцінка) відображається студентові.

Створення тесту викладачем:

- Викладач починає створення нового тесту, вказуючи назву тесту та інші відповідні атрибути.

- Викладач додає питання до тесту, обравши питання з пулу питань або створивши нове питання.

- Для кожного питання, викладач вказує текст питання та варіанти відповідей (якщо необхідно).

- Після додавання всіх питань, викладач зберігає тест у системі для подальшого використання.

Перегляд результатів викладачем:

- Викладач вибирає конкретний тест, результати якого він хоче переглянути.

- Система відображає список студентів, які проходили цей тест, та їх результати.

- Викладач може переглянути детальнішу інформацію про відповіді студентів на окремі питання.

- За необхідності, викладач може надати додаткові коментарі або встановити оцінки для кожного студента.

Ця діаграма діяльності надає загальний огляд послідовності дій, які відбуваються в модулі "Теорія множин". Кожна дія має свої вхідні та вихідні дані, а також взаємодіє з іншими елементами системи (наприклад, базою даних). Залежно від вашого конкретного сценарію використання та потреб вашого проекту, ви можете додати більше деталей та галужень до діаграми діяльності.

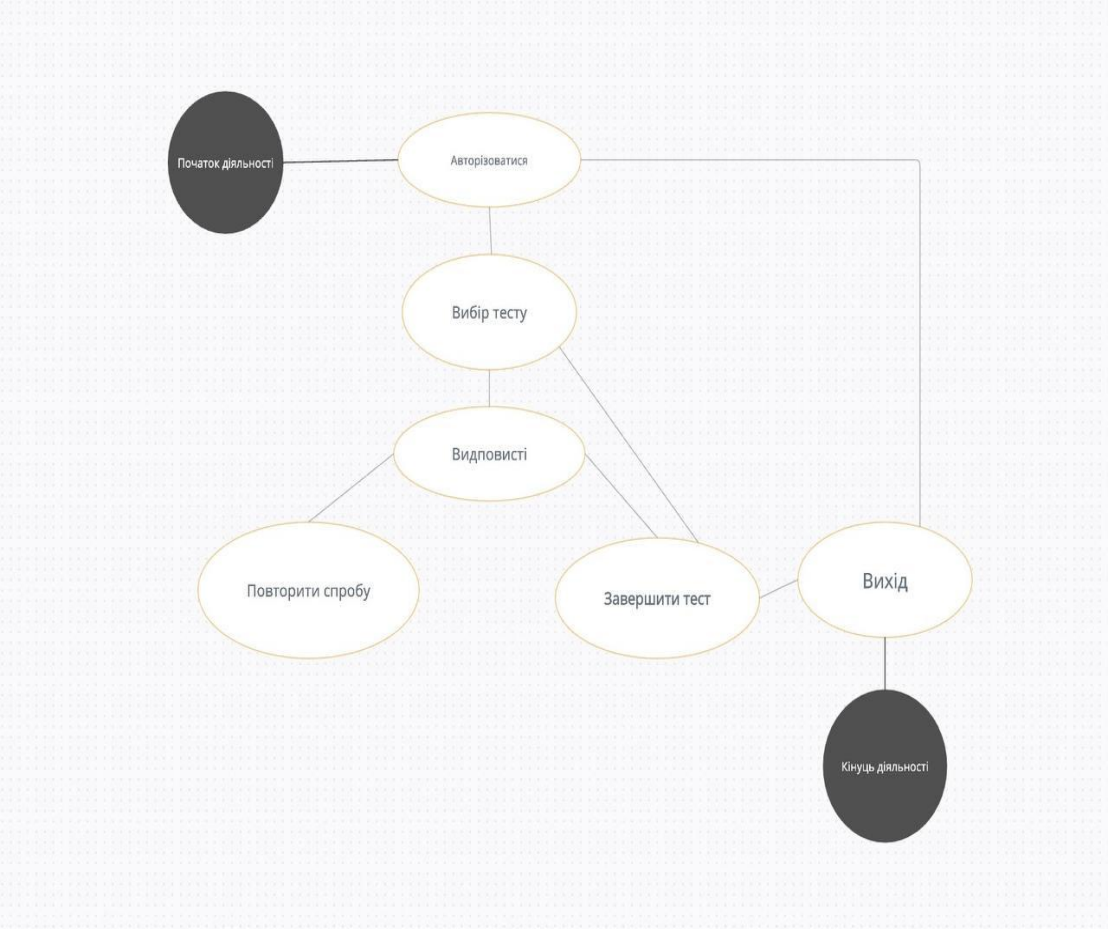
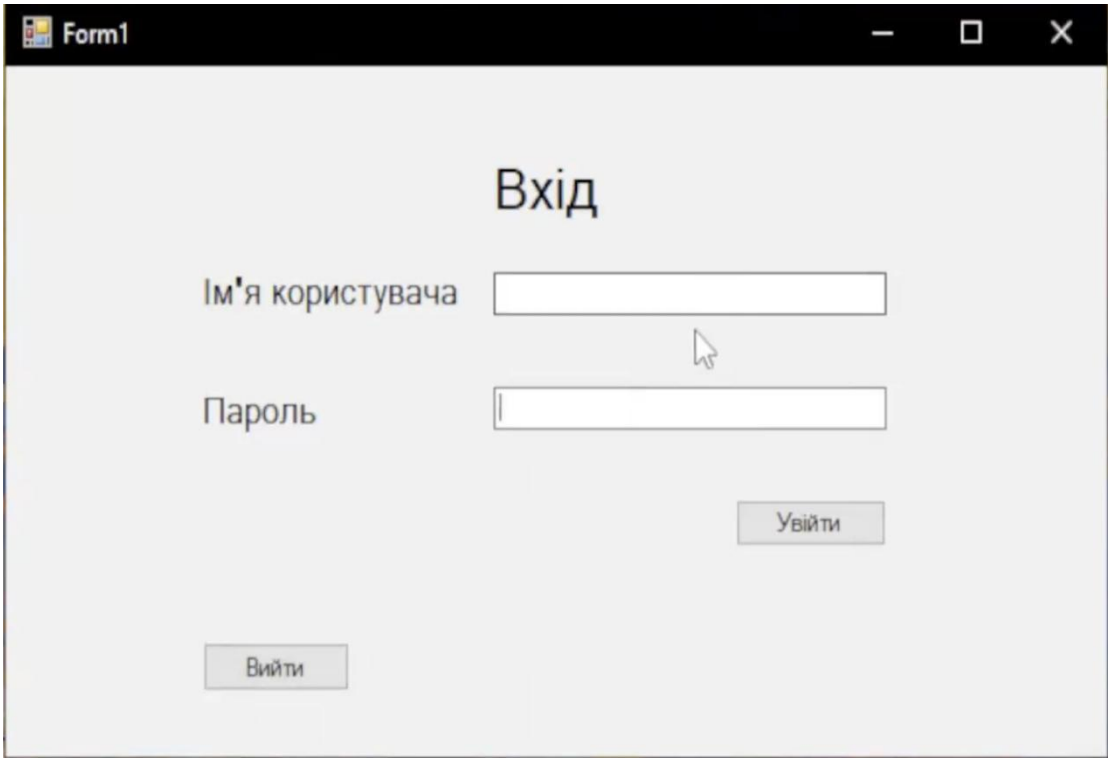


Рисунок 3 - Дінрама діяльності

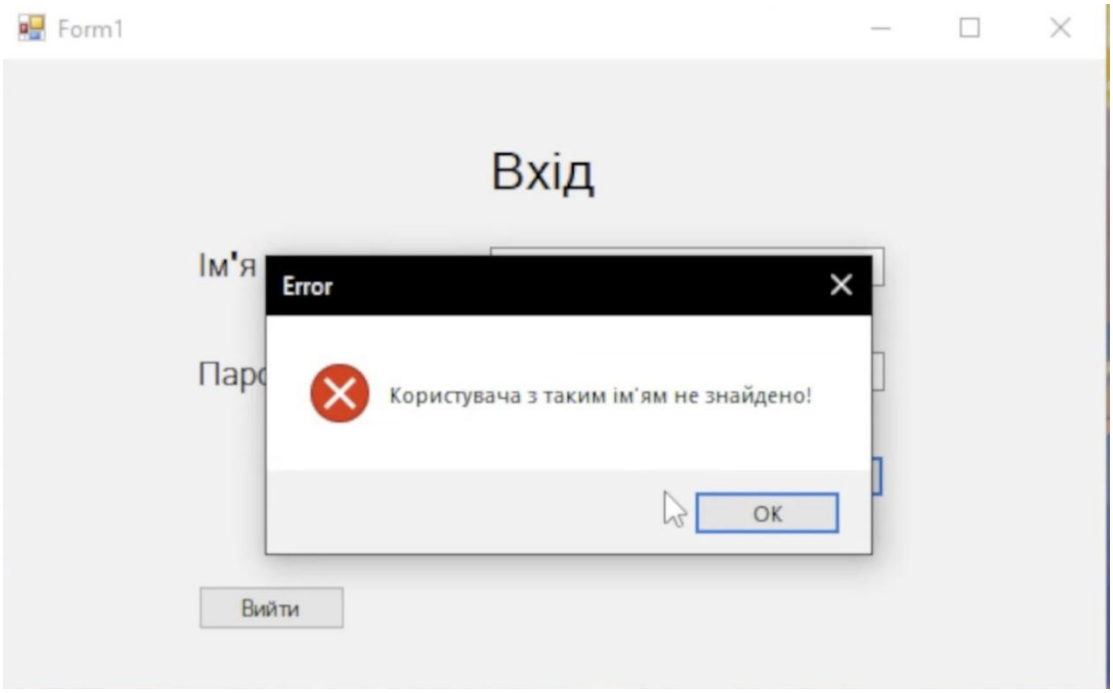
3.4 Опис роботи программи

Перше при запуску программи зразу пропонуе увійти в додаток. Треба ввести своє <<імя користувача>> та <<пароль>>. На даному ктапі доступні дві кнопки <<увійти>> та <<вийти>>ю



Ричунок 4 - Вхід у додаток

Якщо авторизація не була пройдена, то программа буде казати <<Користувач з таким імям не знайдено>>. При такому випадку потрібно буде звернутися до викладача.



Риснок 5 - не вірний логін або пароль

При успішній авторизацій для виклвдачів зразу відкриваєється віконце та пропонує обрати <<пройти тест>> чи <<редагувати тест>>.

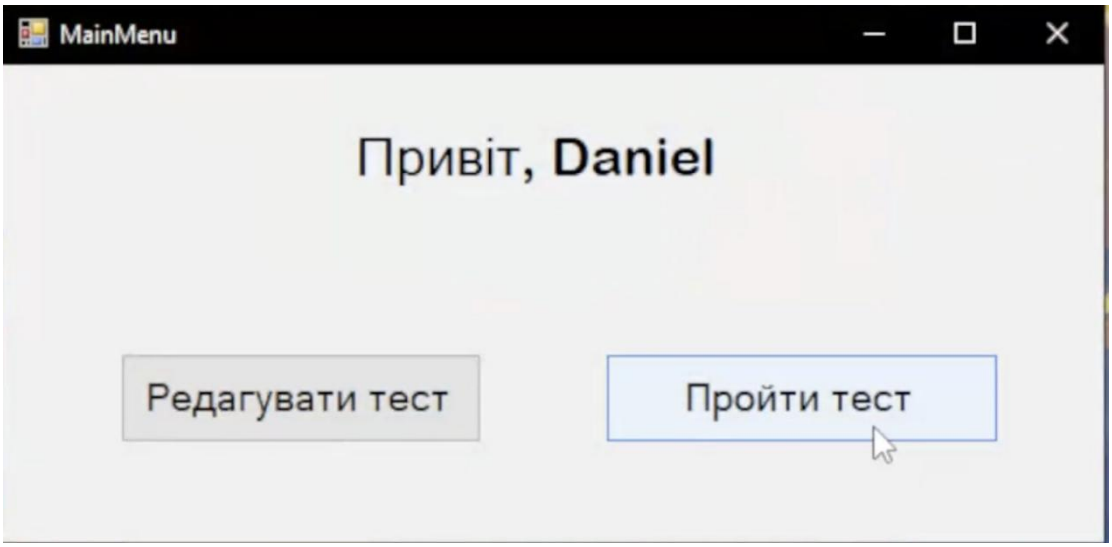


Рисунок 6 - вибір режиму

При нажаті на кнопку <<Редогувати тест >> вас перекидує на сторінку де можно буде додати питання та видалати вже існуюче. Там будуть доступні дві кнопки <<додати>> та <<назад>>, щоб видалити тест і редагувати існуючій тест, то потрибно насти на нйого і редашувати його.

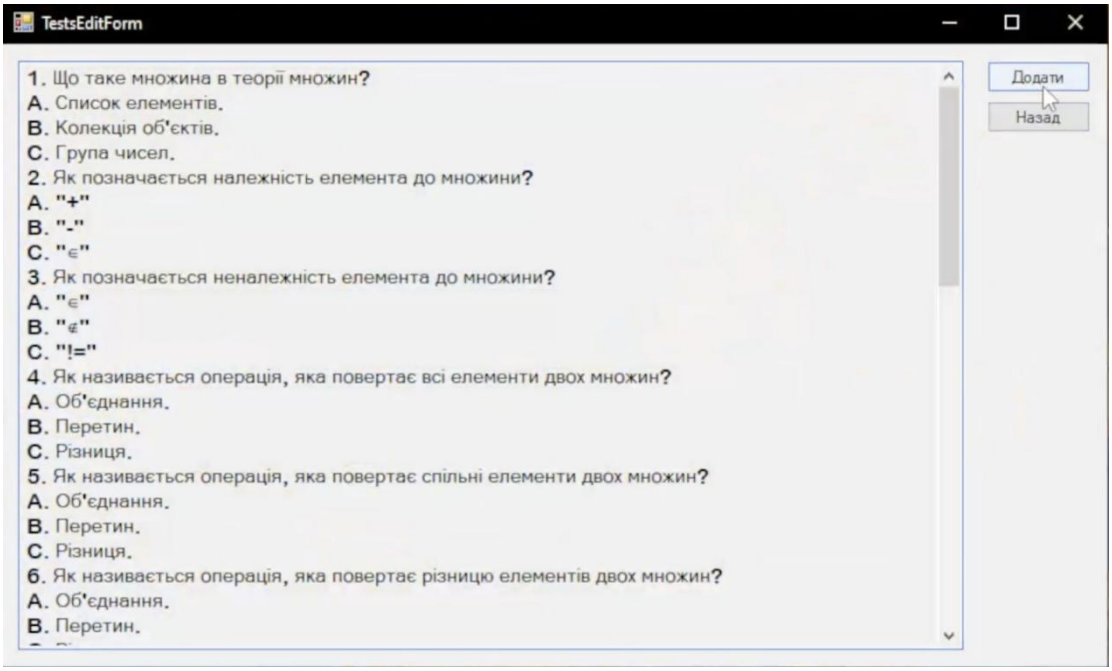


Рисунок 7 - звірка з питаннями

При створенні питання можливо створити 3 вид запитання:

1 Звичайний тест де тільки одна відповідь вірна.

2 Тест де можуть бути дві або більше відповідей.

3 Питання з відкритою відповідею.

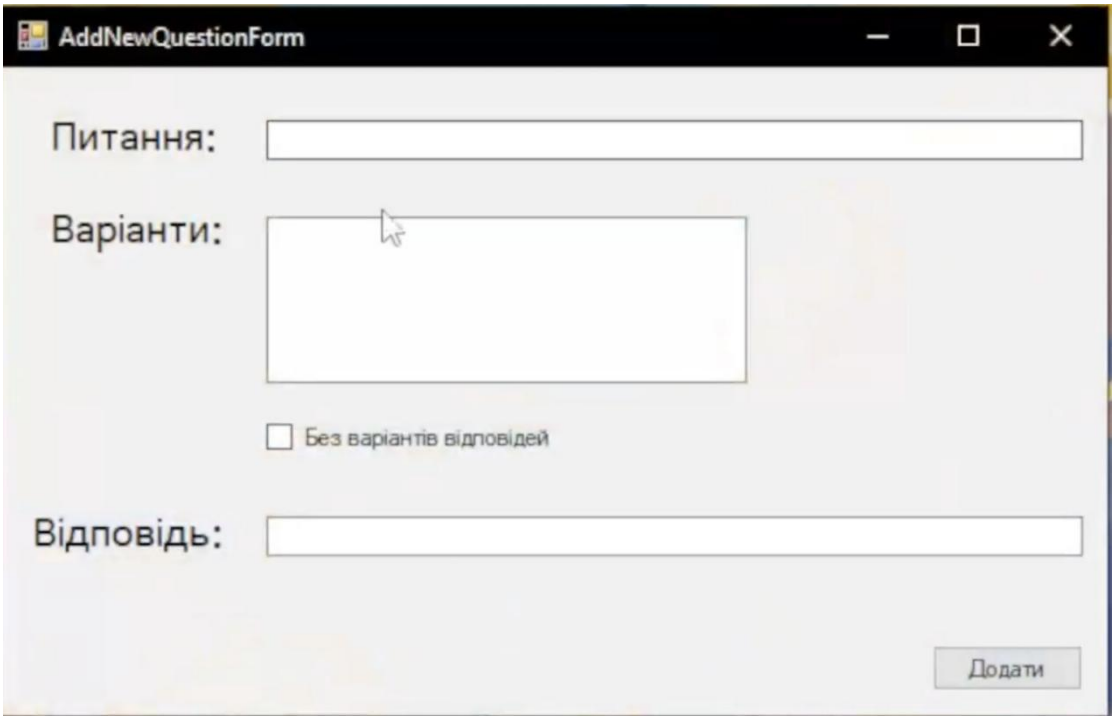


Рисунок 8 - Створити питання

Додавати завдань можно безлічь. При відповіді буду два вида відповідей.

1 Тести де можно вибирати одну чи більше відповідей

2 З откритою відповідею

І там є можливість відповісти на запитання і нажу’ати кнопку <<відповісти>>

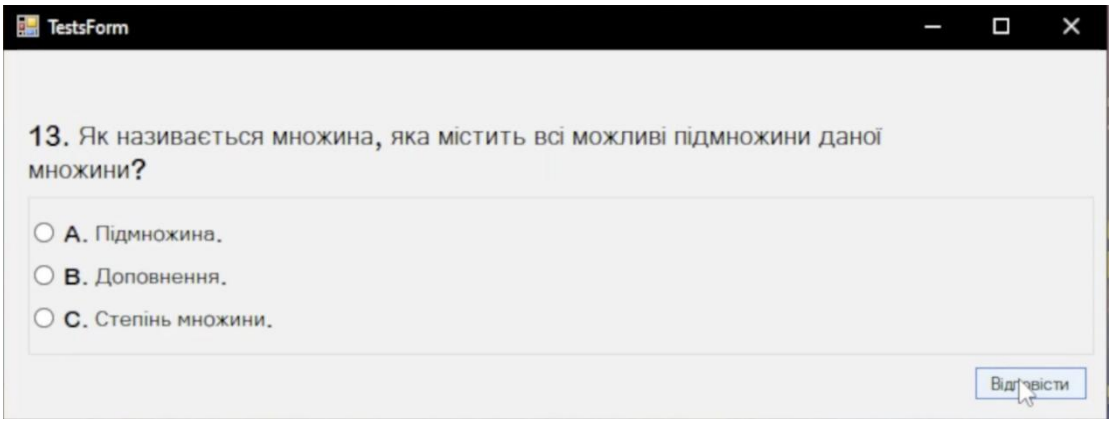


Рисунок 9 - тестове питтная

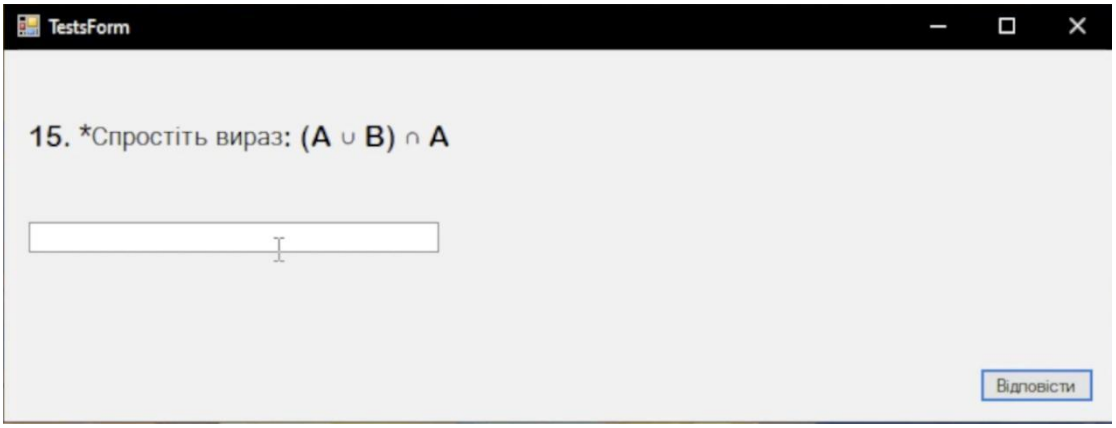
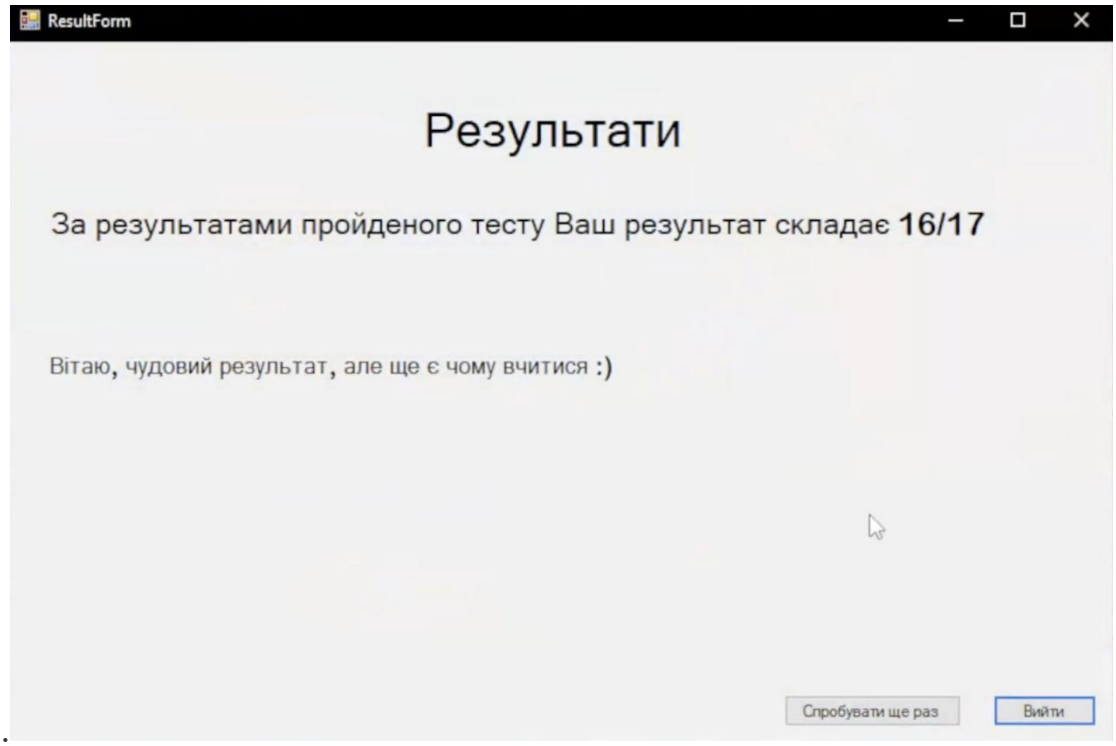


Рисунок 10 - питання з відкритою відповілю

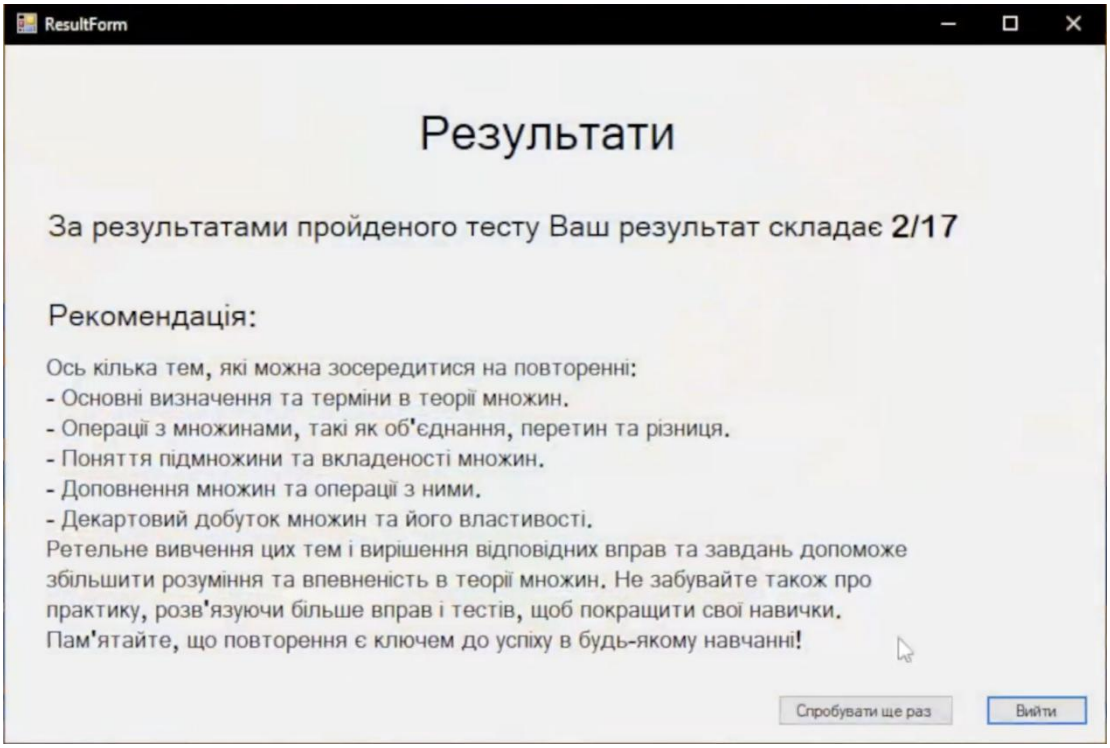
При успішній здачі тесту буде можливість пройти його ще раз та закінчити спробу. Там зроблені дві кнопкі <<спробувати ще раз>> та <<вийти >>.



.

Риссунок 11 - Результат

Якщо ви не набрали потрібну суму балів то вам сам додаток запропонує з якоі теми вам потрібно підняти знання. Рекомендацій автомотично підбираються спроводу ваших ошибок.



Риссунок 12 - Рекомендація літератури

Цей додаток спростить працю в вирішені оцінки для викладачів і також допоможе учнам швидше находити тему для виправленні свойх помилок і почати робити аналіз зі своїми работами.

3.5 Система оцінювання

Система оцінювання для розробки програмного забезпечення для оцінки знань студентів з дискретної математики може бути складною та включати кілька компонентів. Ось загальний опис системи оцінювання:

- Автентифікація та авторизація: Система повинна мати механізми для ідентифікації та аутентифікації користувачів, наприклад, студентів та викладачів. Це може бути здійснено за допомогою логіну та паролю або іншими методами автентифікації, такими як одноразові паролі або системи одноразових паролів.

- Управління студентами та курсами: Система повинна мати можливість реєструвати студентів на курси з дискретної математики та керувати їхніми

даними. Це включає створення профілю студента, назначення курсів та контроль доступу до матеріалів курсу.

- Навчальні матеріали: Система повинна надавати доступ до навчальних матеріалів з теорії множин, таких як підручники, лекції, вправи тощо. Це може включати структуроване представлення матеріалів та можливість навігації через них.

- Тестування та оцінювання: Система повинна мати модуль для проведення тестів та оцінювання знань студентів. Це може включати створення тестів з питаннями різного типу (наприклад, багатоваріантні, з вибором однієї правильної відповіді, або відкриті питання), автоматичну перевірку відповідей та підрахунок оцінок.

- Статистика та звіти: Система повинна збирати та аналізувати дані про прогрес та результати студентів. Це може включати створення звітів про оцінки студентів, статистику успішності та прогресу, а також інші аналітичні звіти для викладачів та адміністраторів.

- Взаємодія з користувачами: Система повинна мати зручний інтерфейс для студентів та викладачів, щоб вони могли легко взаємодіяти зі всіма компонентами системи. Це може бути веб-інтерфейс, додаток для мобільних пристроїв або комбінація обох.

- Забезпечення безпеки: Система повинна мати заходи для забезпечення безпеки даних користувачів, зокрема персональних даних студентів та результатів їхнього навчання. Це може включати шифрування даних, контроль доступу, аудит дій користувачів та захист від вторгнень.

Ці компоненти системи можуть бути реалізовані з використанням мови програмування C# та підтримуваних баз даних, таких як SQL Server або SQLite. При розробці системи оцінювання, рекомендується дотримуватись найкращих практик програмування та безпеки, таких як застосування патернів проектування, перевірка введених даних та використання засобів криптографії для захисту конфіденційності даних.

При розробці модулю "Теорія множин" мовою C# для системи оцінювання з дискретної математики можна розглянути такі деталі:

- Моделювання даних: Розробка модулю починається з моделювання даних, пов'язаних з теорією множин. Це може включати створення класів або структур, що представляють множини, елементи множин, операції над множинами та будь-які інші відповідні концепції.

- Функціональні можливості: Модуль може містити реалізацію основних функцій теорії множин, таких як об'єднання, перетин, різниця, доповнення множин, декартовий добуток тощо. Ці функції можуть бути реалізовані як методи класу або окремі функції, які працюють зі створеними об'єктами множин.

- Перевірка правильності введених даних: Модуль може містити перевірку правильності введених даних користувача. Наприклад, перевірка на наявність дублікатів елементів у множині, перевірка на коректність операцій над множинами (наприклад, перетин двох множин має бути виконаний над однаковими типами множин).

- Графічний інтерфейс користувача: Модуль може мати графічний інтерфейс користувача, який дозволяє вводити дані про множини, виконувати операції над множинами та отримувати результати. Графічний інтерфейс може включати елементи, такі як кнопки, поля введення, віджети для відображення результатів тощо.

- Збереження та завантаження даних: Модуль може мати функціонал збереження та завантаження даних про множини.

Це може включати можливість зберігати множини у файлі або базі даних, а також завантажувати їх для подальшої обробки.

- Інтеграція з іншими модулями: Модуль "Теорія множин" може бути інтегрований з іншими модулями системи оцінювання, наприклад, з модулем тестування та оцінювання. Це дозволить використовувати теорію множин у завданнях тестування та оцінювання студентів.

ВИСНОВКИ

Розробка програмного забезпечення для оцінки знань студентів з дискретної математики є важливим завданням, а розробка спеціалізованого модулю "Теорія множин" мовою C# може значно полегшити цей процес. Мова програмування C# має багато переваг, зокрема широку підтримку від Microsoft і потужну інтегровану середовище розробки (IDE) Visual Studio.

Розробка модулю "Теорія множин" мовою C# може включати такі компоненти, як моделювання даних, функціональні можливості для операцій над множинами, перевірку правильності введених даних, графічний інтерфейс користувача та збереження/завантаження даних.

При розробці такого програмного забезпечення необхідно приділяти увагу безпеці. Забезпечення безпеки включає заходи для захисту конфіденційності та цілісності даних користувачів, контроль доступу до системи та перевірку введених даних. Для досягнення цих цілей можуть використовуватись різні методи, такі як шифрування даних та аудит дій користувачів.

Офіційні джерела, такі як документація мови C#, документація Visual Studio, книги та статті з теми дискретної математики та розробки програмного забезпечення, можуть бути використані для збору додаткової інформації та деталей розробки.

Узагальнюючи, розробка програмного забезпечення для оцінки знань студентів з дискретної математики з використанням модулю "Теорія множин" мовою C# може покращити ефективність та точність процесу оцінювання, а також забезпечити зручний інтерфейс для користувачів. Розробка такого модулю вимагає уважного планування, розробки та тестування, а також забезпечення безпеки даних користувачів.

Для успішної розробки програмного забезпечення для оцінки знань студентів з дискретної математики та модулю "Теорія множин" мовою C# важливо дотримуватись кількох кроків:

1. Аналіз вимог: Перш ніж приступати до розробки, необхідно чітко визначити вимоги до програмного забезпечення. Це включає визначення функціональних та нефункціональних вимог, вимог щодо безпеки, швидкодії, зручного інтерфейсу користувача та інших аспектів.

2. Проектування архітектури: На основі вимог до системи потрібно розробити архітектуру програмного забезпечення. Вона включає розподіл функціональності на компоненти, взаємодію між компонентами та вибір відповідних технологій та патернів проектування.

3. Розробка модулю "Теорія множин": На основі проектованої архітектури розпочинається розробка самого модулю "Теорія множин" мовою C#. Це включає реалізацію класів та методів для роботи з множинами, операцій над ними, перевірки правильності введених даних та інших функціональних можливостей, які були визначені на етапі аналізу вимог.

4. Тестування та відлагодження: Після розробки модулю необхідно провести тестування, щоб переконатись у його правильній роботі та відповідності вимогам. Виявлені помилки та проблеми слід виправити на етапі відлагодження.

5. Забезпечення безпеки: У процесі розробки важливо приділяти увагу забезпеченню безпеки системи. Це може включати захист від несанкціонованого доступу до даних, шифрування конфіденційної інформації, перевірку введених даних на наявність шкідливих кодів та інші заходи.

6. Документація: Необхідно створити документацію, яка описує функціональність модулю, його використання та взаємодію з іншими компонентами системи. Це допоможе іншим розробникам або користувачам зрозуміти, як правильно використовувати модуль.

Узагаління розробки програмного забезпечення для оцінки знань студентів з дискретної математики та модулю "Теорія множин" мовою C# включає в себе ретельний аналіз вимог, проектування архітектури, розробку, тестування та

забезпечення безпеки. Дотримання цих кроків допоможе забезпечити стабільну та функціональну систему оцінювання з дискретної математики.

СПИСОК ЛІТЕРАТУРИ

1. Курси та матеріали з дискретної математики:

- Курси MIT OpenCourseWare: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/

- Курси Coursera: https://www.coursera.org/

- Книги з дискретної математики, такі як "Дискретна математика та її застосування" Кеннета Розенбаума (Kenneth Rosen)

2. Офіційна документація та ресурси для мови програмування C#:

- Документація Microsoft C#: https://docs.microsoft.com/uk-ua/dotnet/csharp/

- Офіційний сайт C#: https://docs.microsoft.com/uk-ua/dotnet/csharp/

3. Джерела для розробки програмного забезпечення:

- Книги та ресурси про розробку програмного забезпечення, такі як "Чистий код. Рефакторинг, проектирование и паттерны на платформе .NET" Роберта Мартіна (Robert C. Martin)

- Ресурси для розробки програмного забезпечення, такі як Stack Overflow (https://stackoverflow.com/) та GitHub (https://github.com/)

4. Академічні ресурси:

- Електронні бібліотеки та бази даних наукових статей, такі як IEEE Xplore (https://ieeexplore.ieee.org/) та ACM Digital Library (https://dl.acm.org/), можуть містити статті та дослідження, пов'язані з дискретною математикою та розробкою програмного забезпечення.

- Академічні журнали з комп'ютерних наук та математики, такі як Journal of Discrete Mathematics (https://www.journals.elsevier.com/journal-of-discrete-mathematics) та Journal of Computer Science and Technology (https://www.springer.com/journal/11390), можуть містити статті з тематики дискретної математики та програмування.

5. Онлайн-курси та платформи:

- Платформи для онлайн-навчання, такі як Udemy (https://www.udemy.com/), Coursera (https://www.coursera.org/), Pluralsight (https://www.pluralsight.com/), можуть містити курси, присвячені дискретній математиці та програмуванню.

- Веб-сайти з безкоштовними онлайн-курсами, такі як Khan Academy (https://www.khanacademy.org/) та edX (https://www.edx.org/), можуть містити матеріали з дискретної математики та програмування.

6. Форуми та спільноти:

- Форуми програмістів та спільноти, такі як Reddit (https://www.reddit.com/r/learnprogramming/) та Stack Exchange (https://stackexchange.com/), можуть мати спеціалізовані підфоруми, де можна задавати запитання щодо дискретної математики, мови програмування C# та розробки програмного забезпечення.

7. "Discrete Mathematics and Its Applications" by Kenneth H. Rosen - ця книга надає вступ до основних концепцій та методів дискретної математики, що можуть бути застосовані в розробці модулю "Теорія множин".

8. "C# 9.0 in a Nutshell: The Definitive Reference" by Joseph Albahari and Ben Albahari - ця книга є вичерпним джерелом інформації про мову C# та його основні функції, що можуть бути корисними під час розробки модулю.

9. "Building Secure Software: How to Avoid Security Problems the Right Way" by John Viega and Gary McGraw - ця книга надає розуміння основних принципів безпеки програмного забезпечення та методів для запобігання вразливостям. Це може бути корисним при розробці безпечного програмного забезпечення для оцінки знань студентів.

10. "Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design" by Michael J. Hernandez - ця книга надає розуміння основ проектування баз даних та допоможе вам створити ефективну діаграму бази даних для вашого програмного забезпечення.

11. Офіційна документація Microsoft для мови C# та інструментів розробки Visual Studio - офіційна документація надає вичерпну інформацію про мову C#, фреймворк .NET та середовище розробки Visual Studio. Вона може бути

використана для отримання детальної інформації про функціональність та можливості мови C# та інструментів розробки.

12. Офіційна документація Microsoft для .NET та C#: На офіційному веб-сайті Microsoft ви знайдете документацію, приклади коду, уроки та різноманітні ресурси, що стосуються розробки програмного забезпечення з використанням мови C# та фреймворка .NET.

13. Stack Overflow: Це популярний веб-сайт, де програмісти можуть задавати питання та отримувати відповіді від спільноти розробників. Ви можете знайти вже відповіді на подібні питання або задати власні, які вам цікаві.

14. GitHub та інші веб-сайти з відкритим кодом: Ви можете шукати відкритий код проектів, які стосуються дискретної математики, мови C# та розробки програмного забезпечення загалом. Це може надати вам приклади реалізації та ідеї для вашого власного проекту.

15. Академічні статті та журнали: Дослідницькі статті та журнали, пов'язані з дискретною математикою, розробкою програмного забезпечення та освітою, можуть надати вам глибше розуміння теми та новітніх розробок у цій галузі.

16. Онлайн-курси та навчальні ресурси: Існує безліч онлайн-курсів та навчальних ресурсів, які надають матеріали, відеоуроки та завдання, спрямовані на вивчення дискретної математики, мови C# та розробки програмного забезпечення.

ДОДАТОК А
ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка програмного за безпечення для оцінки знань с тудентів з дискретної математики. Спец частина. Розробка модулю «Теорія множин» мовою C#

Виконав студент 4 курсу
групи ПД-44
Гуцан Данило Вячеславович
Керівник роботи Садовенко В.С.
К.т.н, доц, доцент кафедри ІПЗ Золотухіна Оксана Анатоліївна

Київ – 2023

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. ***Аналіз вимог***
2. ***Проектування архітектури***
3. ***Розробка модулю***
4. ***Тестування та налагодження***
5. ***Інтеграція з іншими компонентами***
6. ***Оцінка ефективності***
7. ***Написання дипломної роботи***
8. ***Захист дипломної роботи***

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги:

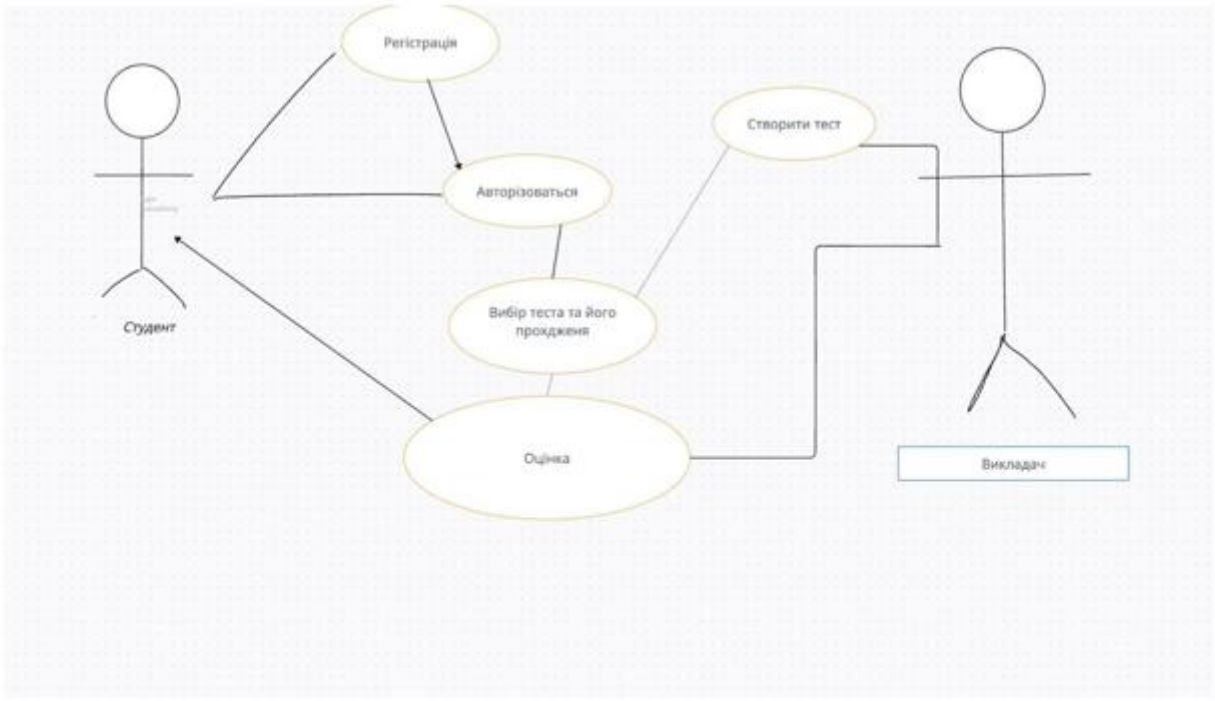
1. *Реєстрація та авторизація користувачів: Можливість реєстрації нових користувачів і авторизації існуючих.*
2. *Навчальний матеріал: Забезпечення доступу до відповідного навчального матеріалу з теорії множин.*
3. *Тестування: Реалізація тестів, які дозволяють оцінювати рівень знань студентів з теорії множин.*
4. *Оцінювання результатів: Автоматична оцінка відповідей студентів та надання зворотного зв'язку.*

5

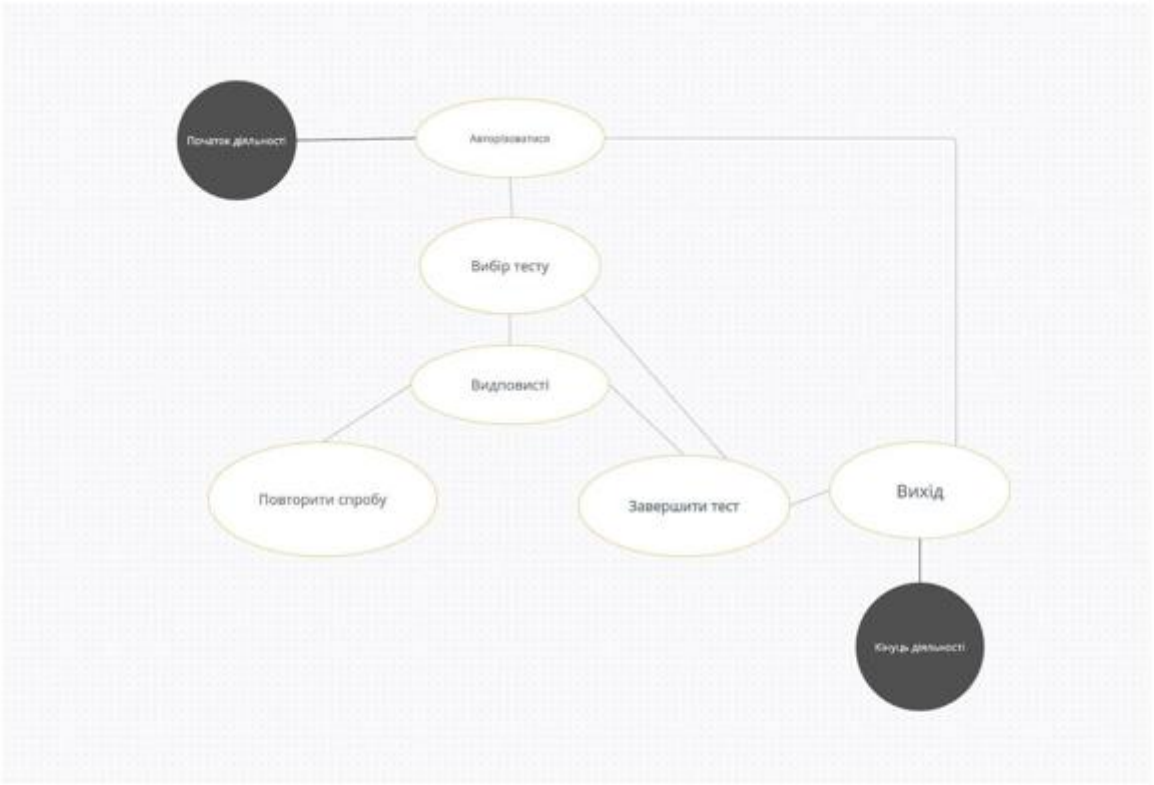
ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Microsoft .NET Framework або .NET Core: Це основний фреймворк для розробки програмного забезпечення на мові C#. Він надає широкий набір бібліотек і засобів для розробки, тестування і виконання додатків.
Entity Framework: Це об'єктно-орієнтований мапер об'єктно-реляційного відображення (ORM), який дозволяє зручно взаємодіяти з базою даних у вашому додатку. Ви можете використовувати Entity Framework для створення моделей даних і виконання операцій з базою даних, таких як зчитування, запис, оновлення тощо.

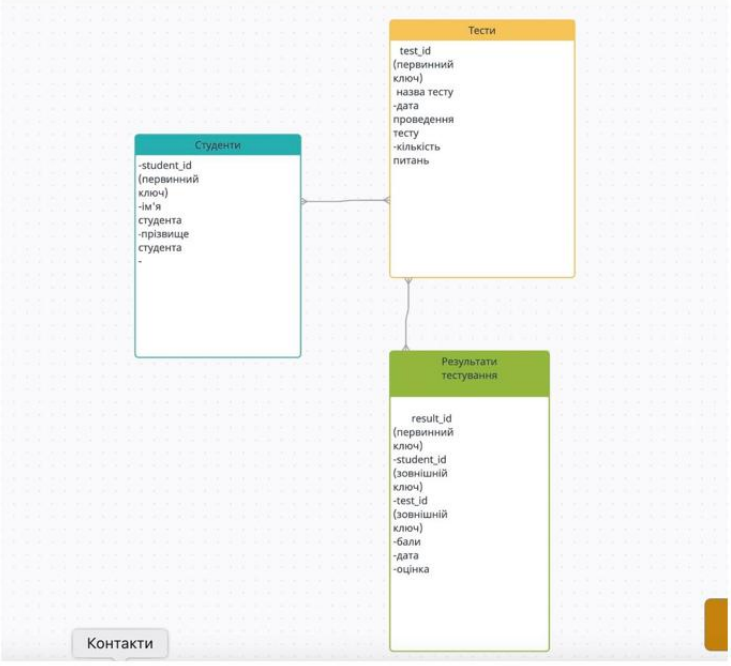
6



Діаграма прецедентів

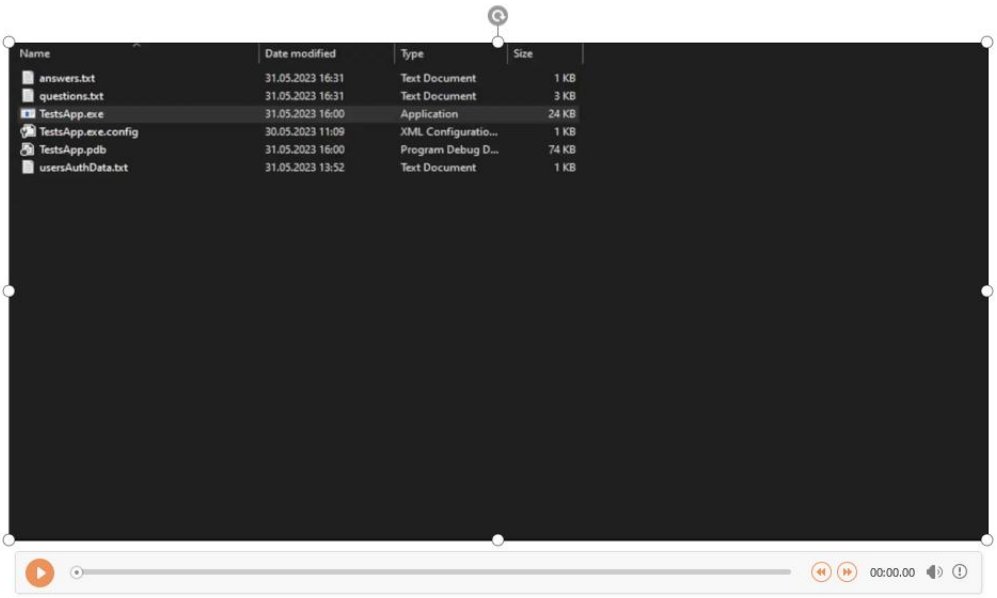


Дінрама діяльності

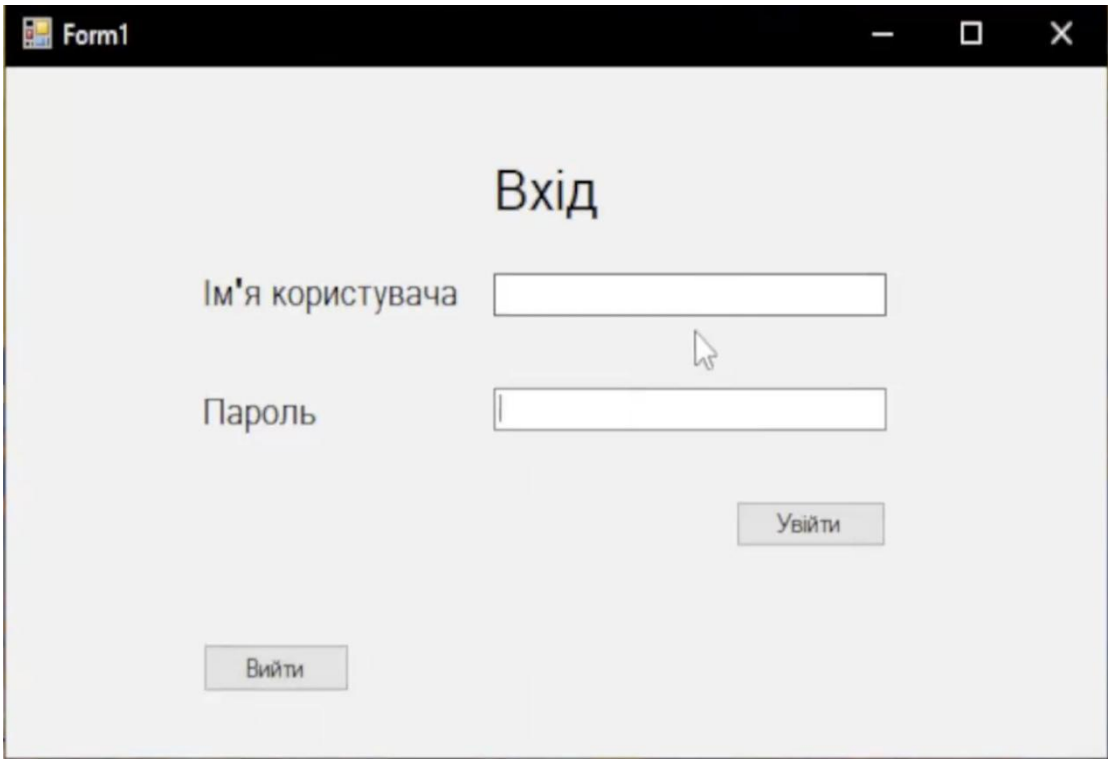


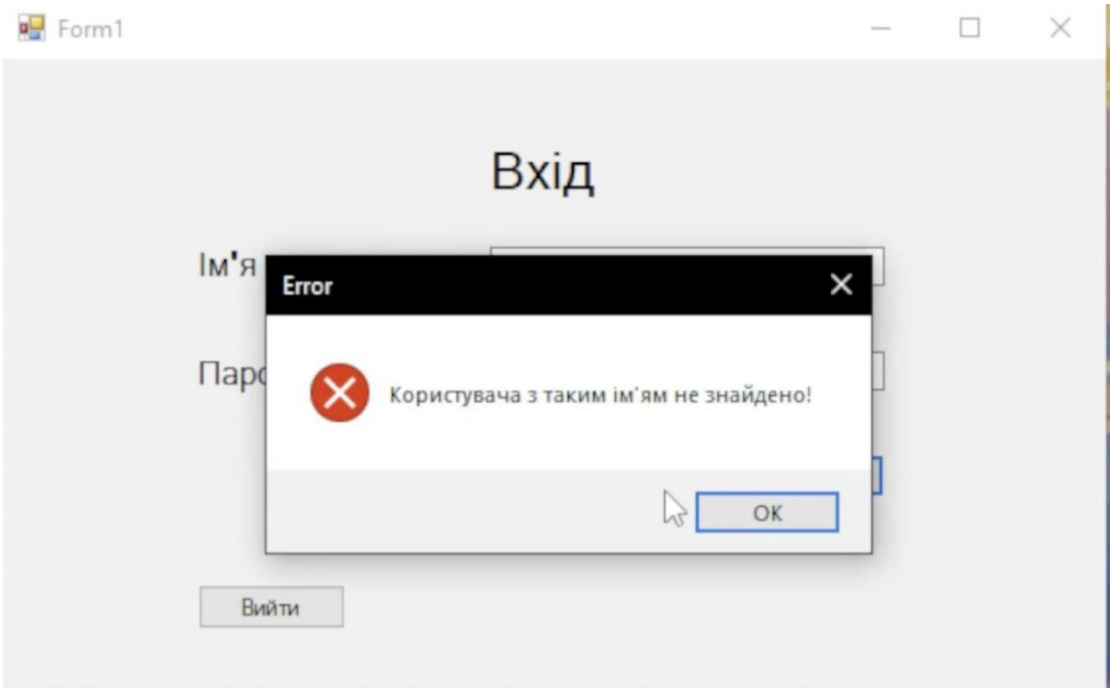
Діаграма бази даних

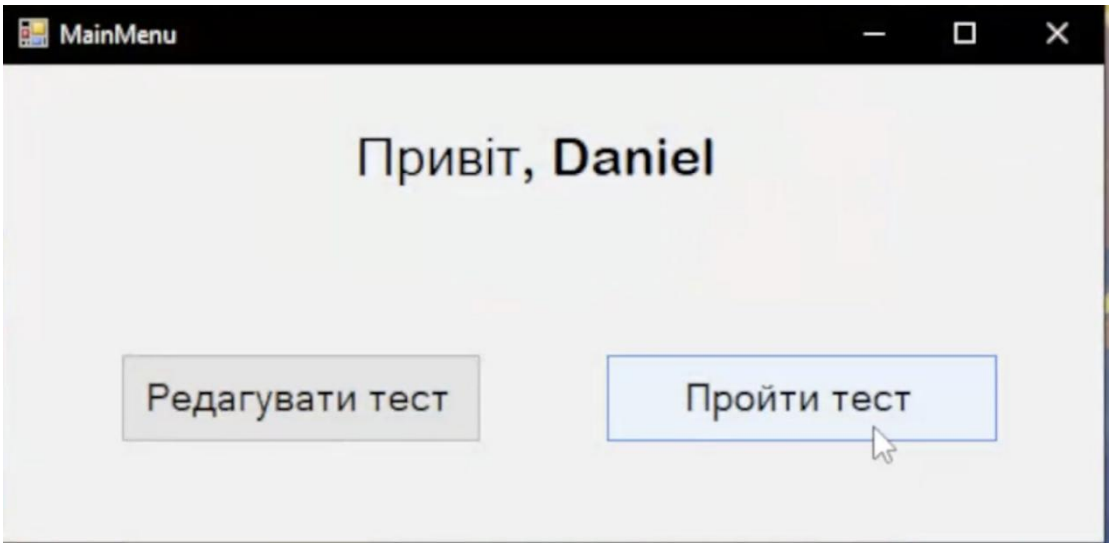
ЕКРАННІ ФОРМИ

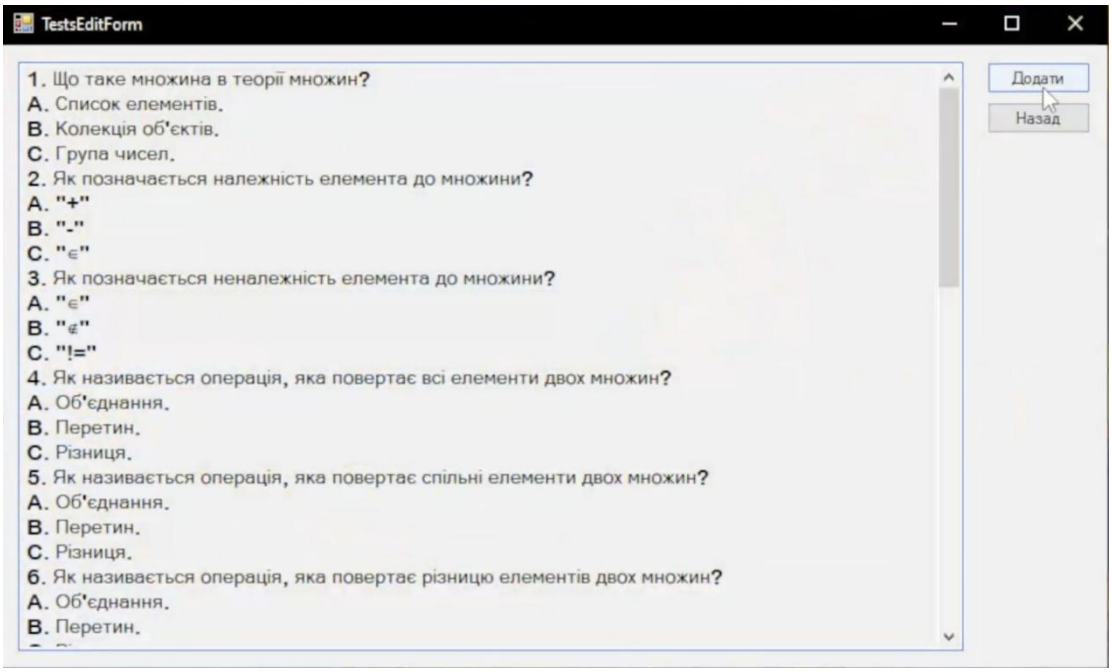


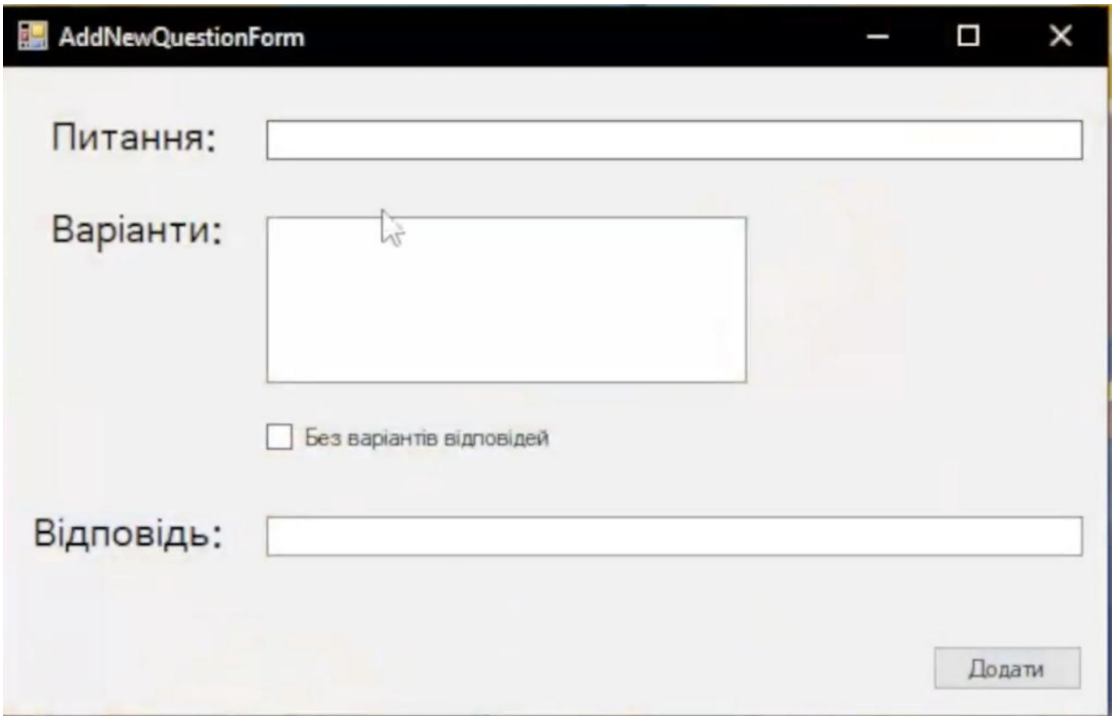
9

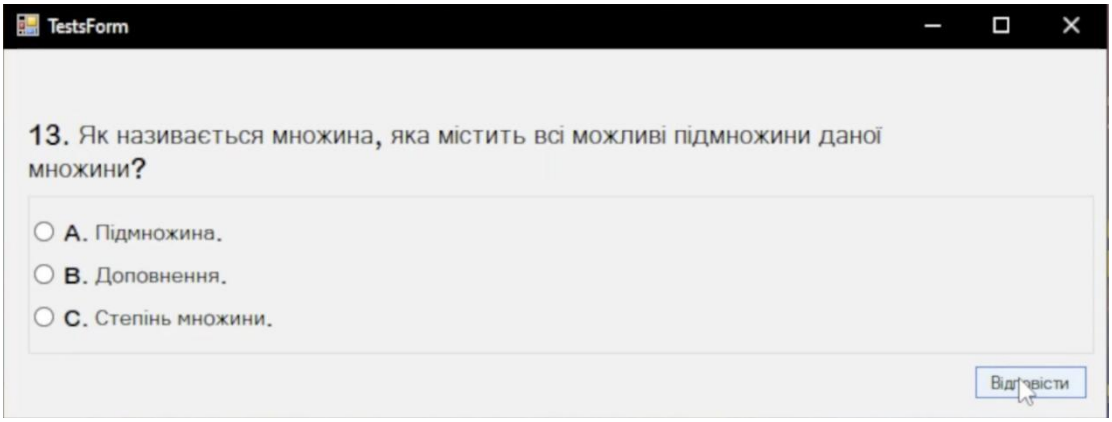


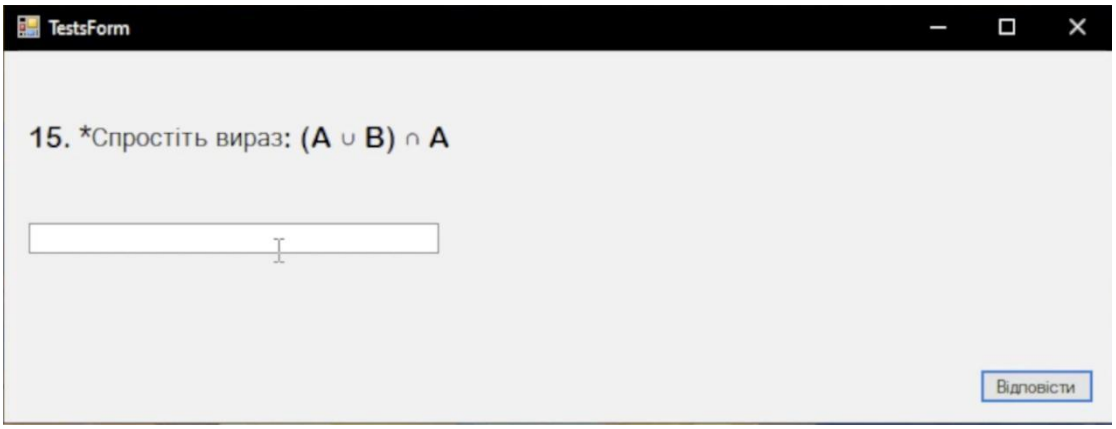

Form1
Вхід
Ім'я
Error
Користувача з таким ім'ям не знайдено!
OK
Вийти

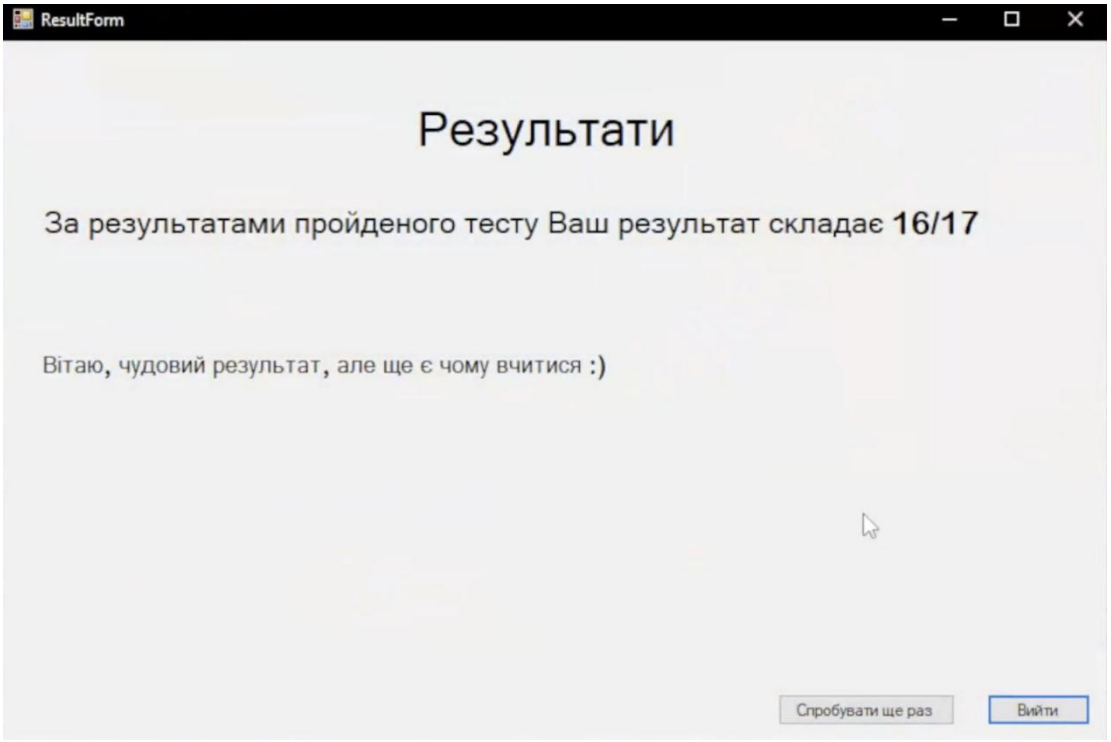

MainMenu
Привіт, Daniel
Редагувати тест
Пройти тест

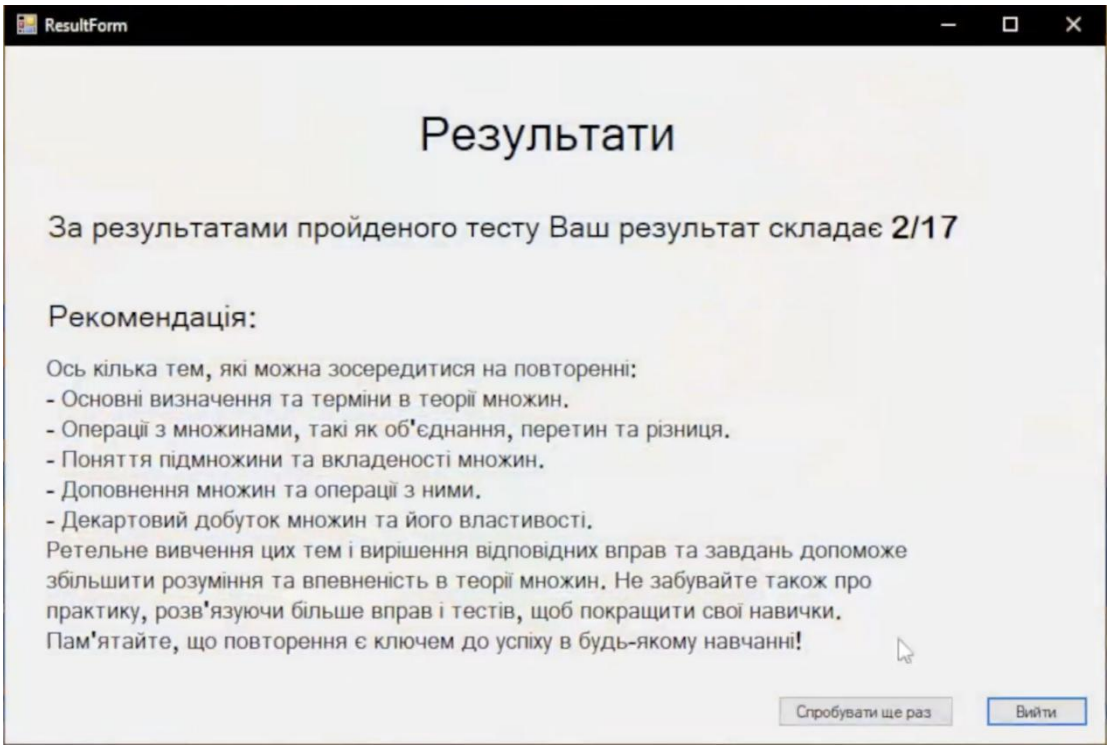

TestsEditForm
1. Що таке множина в теорії множин?
A. Список елементів.
B. Колекція об'єктів.
C. Група чисел.
2. Як позначається належність елемента до множини?
A. "+"
B. "-"
C. "∈"
3. Як позначається неналежність елемента до множини?
A. "∈"
B. "∉"
C. "!="
4. Як називається операція, яка повертає всі елементи двох множин?
A. Об'єднання.
B. Перетин.
C. Різниця.
5. Як називається операція, яка повертає спільні елементи двох множин?
A. Об'єднання.
B. Перетин.
C. Різниця.
6. Як називається операція, яка повертає різницю елементів двох множин?
A. Об'єднання.
B. Перетин.
Додати
Назад


AddNewQuestionForm
Питання:
Варіанти:
Без варіантів відповідей
Відповідь:
Додати


TestsForm
13. Як називається множина, яка містить всі можливі підмножини даної множини?
A. Підмножина.
B. Доповнення.
C. Степінь множини.
Відповісти


TestsForm
15. *Спростіть вираз: (A ∪ B) ∩ A
Відповісти


ResultForm
Результати
За результатами пройденого тесту Ваш результат складає 16/17
Вітаю, чудовий результат, але ще є чому вчитися :)
Спробувати ще раз
Вийти


ResultForm
Результати
За результатами пройденого тесту Ваш результат складає 2/17
Рекомендація:
Ось кілька тем, які можна зосередитися на повторенні:
- Основні визначення та терміни в теорії множин.
- Операції з множинами, такі як об'єднання, перетин та різниця.
- Поняття підмножини та вкладеності множин.
- Доповнення множин та операції з ними.
- Декартовий добуток множин та його властивості.
Ретельне вивчення цих тем і вирішення відповідних вправ та завдань допоможе збільшити розуміння та впевненість в теорії множин. Не забувайте також про практику, розв'язуючи більше вправ і тестів, щоб покращити свої навички.
Пам'ятайте, що повторення є ключем до успіху в будь-якому навчанні!
Спробувати ще раз
Вийти

ВИСНОВКИ

Розробка програмного забезпечення для оцінки знань студентів з дискретної математики є важливим завданням, а розробка спеціалізованого модулю "Теорія множин" мовою C# може значно полегшити цей процес. Мова програмування C# має багато переваг, зокрема широку підтримку від Microsoft і потужну інтегровану середовище розробки (IDE) Visual Studio.

Розробка модулю "Теорія множин" мовою C# може включати такі компоненти, як моделювання даних, функціональні можливості для операцій над множинами, перевірку правильності введених даних, графічний інтерфейс користувача та збереження/завантаження даних.

При розробці такого програмного забезпечення необхідно приділяти увагу безпеці. Забезпечення безпеки включає заходи для захисту конфіденційності та цілісності даних користувачів, контроль доступу до системи та перевірку введених даних. Для досягнення цих цілей можуть використовуватись різні методи, такі як шифрування даних та аудит дій користувачів.

Узагальнюючи, розробка програмного забезпечення для оцінки знань студентів з дискретної математики з використанням модулю "Теорія множин" мовою C# може покращити ефективність та точність процесу оцінювання, а також забезпечити зручний інтерфейс для користувачів. Розробка такого модулю вимагає уважного планування, розробки та тестування, а також забезпечення безпеки даних користувачів.

ДОДАТОК Б

ЛИСТИНГИ ПРОГРАМНИХ МОДУЛІВ

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

namespace TestsApp
{
    public partial class LoginForm : Form
    {
        private readonly MainMenu _mainMenu;
        private const string usersAuthDataFilePath = "usersAuthData.txt";
        private const string UndefinedUsernameErrorMsg = "Користувача з таким ім'ям не
знайдено!";
        private const string WrongUsernameOrPasswordErrorMsg = "Неправильне ім'я
користувача або пароль";

        private Dictionary<string, string> usersAuthData;

        public LoginForm()
        {
            InitializeComponent();

            _mainMenu = new MainMenu();

            this.CenterToScreen();

            usersAuthData = new Dictionary<string, string>();
            if (File.Exists(usersAuthDataFilePath)) {
                using (StreamReader sr = File.OpenText(usersAuthDataFilePath)) {
                    string data;
                    while ((data = sr.ReadLine()) != null) {
                        string[] usernameAndPassword = data.Split(' ');
                        usersAuthData[usernameAndPassword[0]] = usernameAndPassword[1];
                    }
                }
            }
        }

        private void signInBtn_Click(object sender, EventArgs e)
        {
            string username = usernameTxt.Text;
            string password = passwordTxt.Text;

            if (!usersAuthData.ContainsKey(username))
            {
                MessageBox.Show(UndefinedUsernameErrorMsg, "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                usernameTxt.Clear();
                passwordTxt.Clear();
                return;
            }

            if (usersAuthData[username] != password)
            {
```

```
                MessageBox.Show(WrongUsernameOrPasswordErrorMsg, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                usernameTxt.Clear();
                passwordTxt.Clear();
                return;
            }

            _mainMenu.Show();
            this.Hide();
        }

        private void exitBtn_Click(object sender, EventArgs e) {
            Application.Exit();
        }
```

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

namespace TestsApp
{
    public partial class MainMenu : Form
    {
        private readonly TestsForm _testsForm;
        private readonly TestsEditForm _editForm;

        public MainMenu() {
            InitializeComponent();

            this.CenterToScreen();
            _testsForm = new TestsForm();
            _editForm = new TestsEditForm();
        }

        private void takeTestBtn_Click(object sender, EventArgs e)
        {
            _testsForm.Show();
            this.Hide();
        }

        private void editTestsBtn_Click(object sender, EventArgs e)
        {
            _editForm.Show();
            this.Hide();
        }
    }
}
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace TestsApp
```

```
{
    public partial class TestsEditForm : Form
    {
        public TestsEditForm() {
            InitializeComponent();

            this.CenterToScreen();

            questions.ScrollBars = ScrollBars.Both;
            questions.ReadOnly = true;

            string questionsFilePath = "questions.txt";

            using (StreamReader sr = new StreamReader(questionsFilePath)) {
                string line;
                while ((line = sr.ReadLine()) != null) {
                    questions.AppendText(line + Environment.NewLine);
                }
            }
        }

        private void addQuestionBtn_Click(object sender, EventArgs e)
        {
            var addNewQuestionForm = new AddNewQuestionForm();
            addNewQuestionForm.Show();
            this.Hide();
        }

        private void backToMenuBtn_Click(object sender, EventArgs e)
        {
            var menu = new MainMenu();
            menu.Show();
            this.Hide();
        }
    }
}
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Reflection.Emit;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TestsApp
{
    public partial class TestsForm : Form
    {
        private ResultForm resultForm;
        private List<Question> questions;
        private int currentQuestionIndex;
        private int score;

        public TestsForm() {
            InitializeComponent();

            resultForm = new ResultForm();

            this.CenterToScreen();

            // Ініціалізуємо список питань та результат
```

```
            questions = QuestionLoader.LoadQuestionsFromFile("questions.txt",
"answers.txt");
            score = 0;

            // Відображаємо перше питання
            currentQuestionIndex = 0;
            ShowQuestion();

            questionLabel.MaximumSize = new Size(this.Size.Width - 100, 0);
            questionLabel.AutoSize = true;


        }

        public void Reload()
        {
            currentQuestionIndex = 0;
            score = 0;
        }

        private void ShowQuestion()
        {
            var radioBtns = answersGroup.Controls.OfType<RadioButton>();

            foreach (var radioButton in radioBtns)
            {
                radioButton.Checked = false;
            }

            answerTxt.Text = string.Empty;

            // Перевіряємо, чи всі питання пройдені
            if (currentQuestionIndex >= questions.Count) {
                // Відображаємо результати
                resultForm.LoadResult(questions.Count, score);
                resultForm.Show();
                this.Hide();
                return;
            }

            // Отримуємо поточне питання
            Question currentQuestion = questions[currentQuestionIndex];

            // Відображаємо питання
            questionLabel.Text = currentQuestion.QuestionText;

            // Відображаємо варіанти відповідей
            if (currentQuestion.Answers.Count > 0)
            {
                answerRadioButton1.Text = currentQuestion.Answers[0];
                answerRadioButton2.Text = currentQuestion.Answers[1];
                answerRadioButton3.Text = currentQuestion.Answers[2];
                answersGroup.Visible = true;
                answerTxt.Visible = false;
            }
            else
            {
                answersGroup.Visible = false;
                answerTxt.Visible = true;
            }
        }


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

```
            var checkedBtn = answersGroup.Controls.OfType<RadioButton>().FirstOrDefault(r
=> r.Checked);

            if (answerTxt.Visible && answerTxt.Text ==
questions[currentQuestionIndex].CorrectAnswer
                || checkedBtn != null && checkedBtn.Text[0] ==
questions[currentQuestionIndex].CorrectAnswer[0])
            {
                score++;
            }

            currentQuestionIndex++;
            ShowQuestion();
        }

    }
}
namespace TestsApp
{
    partial class TestsForm
    {
        /// <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() {
            this.questionLabel = new System.Windows.Forms.Label();
            this.answersGroup = new System.Windows.Forms.GroupBox();
            this.answerRadioButton3 = new System.Windows.Forms.RadioButton();
            this.answerRadioButton2 = new System.Windows.Forms.RadioButton();
            this.answerRadioButton1 = new System.Windows.Forms.RadioButton();
            this.submitAnswer = new System.Windows.Forms.Button();
            this.answerTxt = new System.Windows.Forms.TextBox();
            this.answersGroup.SuspendLayout();
            this.SuspendLayout();
            //
            // questionLabel
            //
            this.questionLabel.AutoSize = true;
            this.questionLabel.Font = new System.Drawing.Font("Arial Rounded MT Bold",
14.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.questionLabel.Location = new System.Drawing.Point(12, 45);
            this.questionLabel.Name = "questionLabel";
            this.questionLabel.Size = new System.Drawing.Size(65, 22);
            this.questionLabel.TabIndex = 0;
            this.questionLabel.Text = "label1";
```

```
            //
            // answersGroup
            //
            this.answersGroup.Controls.Add(this.answerRadioButton3);
            this.answersGroup.Controls.Add(this.answerRadioButton2);
            this.answersGroup.Controls.Add(this.answerRadioButton1);
            this.answersGroup.Location = new System.Drawing.Point(16, 90);
            this.answersGroup.Name = "answersGroup";
            this.answersGroup.Size = new System.Drawing.Size(700, 111);
            this.answersGroup.TabIndex = 1;
            this.answersGroup.TabStop = false;
            //
            // answerRadioButton3
            //
            this.answerRadioButton3.AutoSize = true;
            this.answerRadioButton3.Font = new System.Drawing.Font("Arial Rounded MT
Bold", 12F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.answerRadioButton3.Location = new System.Drawing.Point(4, 75);
            this.answerRadioButton3.Name = "answerRadioButton3";
            this.answerRadioButton3.Size = new System.Drawing.Size(131, 22);
            this.answerRadioButton3.TabIndex = 0;
            this.answerRadioButton3.TabStop = true;
            this.answerRadioButton3.Text = "radioButton1";
            this.answerRadioButton3.UseVisualStyleBackColor = true;
            //
            // answerRadioButton2
            //
            this.answerRadioButton2.AutoSize = true;
            this.answerRadioButton2.Font = new System.Drawing.Font("Arial Rounded MT
Bold", 12F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.answerRadioButton2.Location = new System.Drawing.Point(4, 47);
            this.answerRadioButton2.Name = "answerRadioButton2";
            this.answerRadioButton2.Size = new System.Drawing.Size(131, 22);
            this.answerRadioButton2.TabIndex = 0;
            this.answerRadioButton2.TabStop = true;
            this.answerRadioButton2.Text = "radioButton1";
            this.answerRadioButton2.UseVisualStyleBackColor = true;
            //
            // answerRadioButton1
            //
            this.answerRadioButton1.AutoSize = true;
            this.answerRadioButton1.Font = new System.Drawing.Font("Arial Rounded MT
Bold", 12F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.answerRadioButton1.Location = new System.Drawing.Point(4, 19);
            this.answerRadioButton1.Name = "answerRadioButton1";
            this.answerRadioButton1.Size = new System.Drawing.Size(131, 22);
            this.answerRadioButton1.TabIndex = 0;
            this.answerRadioButton1.TabStop = true;
            this.answerRadioButton1.Text = "radioButton1";
            this.answerRadioButton1.UseVisualStyleBackColor = true;
            //
            // submitAnswer
            //
            this.submitAnswer.Location = new System.Drawing.Point(637, 207);
            this.submitAnswer.Name = "submitAnswer";
            this.submitAnswer.Size = new System.Drawing.Size(75, 23);
            this.submitAnswer.TabIndex = 2;
            this.submitAnswer.Text = "Відповісти";
            this.submitAnswer.UseVisualStyleBackColor = true;
            this.submitAnswer.Click += new System.EventHandler(this.submitAnswer_Click);
            //
            // answerTxt
```

```
            //
            this.answerTxt.Location = new System.Drawing.Point(16, 112);
            this.answerTxt.Name = "answerTxt";
            this.answerTxt.Size = new System.Drawing.Size(268, 20);
            this.answerTxt.TabIndex = 1;
            this.answerTxt.Visible = false;
            //
            // TestsForm
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(724, 242);
            this.Controls.Add(this.answerTxt);
            this.Controls.Add(this.submitAnswer);
            this.Controls.Add(this.answersGroup);
            this.Controls.Add(this.questionLabel);
            this.Name = "TestsForm";
            this.Text = "TestsForm";
            this.answersGroup.ResumeLayout(false);
            this.answersGroup.PerformLayout();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.Label questionLabel;
        private System.Windows.Forms.GroupBox answersGroup;
        private System.Windows.Forms.RadioButton answerRadioButton3;
        private System.Windows.Forms.RadioButton answerRadioButton2;
        private System.Windows.Forms.RadioButton answerRadioButton1;
        private System.Windows.Forms.Button submitAnswer;
        private System.Windows.Forms.TextBox answerTxt;
    }
}namespace TestsApp
{
    partial class ResultForm
    {
        /// <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() {
            this.title = new System.Windows.Forms.Label();
            this.resultInfo = new System.Windows.Forms.Label();
```

```
            this.recommendationLabel = new System.Windows.Forms.Label();
            this.recommendation = new System.Windows.Forms.Label();
            this.exitBtn = new System.Windows.Forms.Button();
            this.button1 = new System.Windows.Forms.Button();
            this.SuspendLayout();
            //
            // title
            //
            this.title.AutoSize = true;
            this.title.Font = new System.Drawing.Font("Arial Rounded MT Bold", 26.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.title.Location = new System.Drawing.Point(294, 44);
            this.title.Name = "title";
            this.title.Size = new System.Drawing.Size(204, 40);
            this.title.TabIndex = 0;
            this.title.Text = "Результати";
            //
            // resultInfo
            //
            this.resultInfo.AutoSize = true;
            this.resultInfo.Font = new System.Drawing.Font("Arial Rounded MT Bold", 18F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.resultInfo.Location = new System.Drawing.Point(26, 117);
            this.resultInfo.Name = "resultInfo";
            this.resultInfo.Size = new System.Drawing.Size(79, 28);
            this.resultInfo.TabIndex = 1;
            this.resultInfo.Text = "result";
            //
            // recommendationLabel
            //
            this.recommendationLabel.AutoSize = true;
            this.recommendationLabel.Font = new System.Drawing.Font("Arial Rounded MT
Bold", 18F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.recommendationLabel.Location = new System.Drawing.Point(26, 182);
            this.recommendationLabel.Name = "recommendationLabel";
            this.recommendationLabel.Size = new System.Drawing.Size(165, 28);
            this.recommendationLabel.TabIndex = 2;
            this.recommendationLabel.Text = "Рекомендація:";
            //
            // recommendation
            //
            this.recommendation.AutoSize = true;
            this.recommendation.Font = new System.Drawing.Font("Arial Rounded MT Bold",
14.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.recommendation.Location = new System.Drawing.Point(26, 223);
            this.recommendation.Name = "recommendation";
            this.recommendation.Size = new System.Drawing.Size(65, 22);
            this.recommendation.TabIndex = 3;
            this.recommendation.Text = "label2";
            //
            // exitBtn
            //
            this.exitBtn.Location = new System.Drawing.Point(713, 472);
            this.exitBtn.Name = "exitBtn";
            this.exitBtn.Size = new System.Drawing.Size(75, 23);
            this.exitBtn.TabIndex = 4;
            this.exitBtn.Text = "Вийти";
            this.exitBtn.UseVisualStyleBackColor = true;
            this.exitBtn.Click += new System.EventHandler(this.exitBtn_Click);
            //
            // button1
            //
            this.button1.Location = new System.Drawing.Point(562, 472);
            this.button1.Name = "button1";
```

```
            this.button1.Size = new System.Drawing.Size(128, 23);
            this.button1.TabIndex = 5;
            this.button1.Text = "Спробувати ще раз";
            this.button1.UseVisualStyleBackColor = true;
            this.button1.Click += new System.EventHandler(this.button1_Click);
            //
            // ResultForm
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(800, 507);
            this.Controls.Add(this.button1);
            this.Controls.Add(this.exitBtn);
            this.Controls.Add(this.recommendation);
            this.Controls.Add(this.recommendationLabel);
            this.Controls.Add(this.resultInfo);
            this.Controls.Add(this.title);
            this.Name = "ResultForm";
            this.Text = "ResultForm";
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.Label title;
        private System.Windows.Forms.Label resultInfo;
        private System.Windows.Forms.Label recommendationLabel;
        private System.Windows.Forms.Label recommendation;
        private System.Windows.Forms.Button exitBtn;
        private System.Windows.Forms.Button button1;
    }
}
```