МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
ПрАТ «ПРИВАТНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ
ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Предметно-циклова комісія інформаційних технологій

ДО ЗАХИСТУ ДОПУЩЕНА

Голова ПЦК _____

спеціаліст в/к Сабанов С.О.

ВИПУСКНА РОБОТА МОЛОДШОГО СПЕЦІАЛІСТА

РОЗРОБКА ІНТЕРАКТИВНОГО НАВЧАЛЬНОГО КОМПЛЕКСУ З ДИСКРЕТНОЇ МАТЕМАТИКИ

Виконав
ст. гр. ІПЗ-119К9 _____ М.А. Резніченко

Керівник
професор _____ С.О. Сабанов

Запоріжжя
2023

СТРУКТУРНИЙ ПІДРОЗДІЛ «ФАХОВИЙ КОЛЕДЖ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРАТ «ПВНЗ «ЗІЕІТ»

Предметно-циклова комісія інформаційних технологій

ЗАТВЕРДЖУЮ

Голова ПЦК спеціаліст в/к

Сабанов С.О._____

“21” лютого 2023 року

З А В Д А Н Н Я

НА ВИПУСКНУ РОБОТУ МОЛОДШОГО СПЕЦІАЛІСТА

студенту гр.ІПЗ-119К9

спеціальності 121 – «Інженерія програмного забезпечення»

Резніченку Максиму Андрійовичу

1. Тема: «*Розробка інтерактивного навчального комплексу з дискретної математики»*

затверджена наказом по інституту: № 09.2-14 від 20 лютого 2023 року

2. Термін здачі студентом закінченої роботи: 20 червня 2023 року

3. Перелік питань, що підлягають розробці:

1. Провести огляд літератури та інтернет-джерел, присвячених тематиці випускної роботи.

2.Виконати огляд найбільш популярних програмно-апаратних інтерактивних засобів навчання

3.Розглянути інструментальні засоби створення додатків з метою вибору оптимальних для створення проекту.

4. Визначитися з основними функціональними блоками проекту та структурою бази даних.

5. Реалізувати алгоритм роботи додатку.

6. Протестувати розробку та провести остаточне налагодження

7. Створити достатнє наповнення БД для нормального користування системою.

8. Оформити результати роботи у вигляді пояснювальної записки, що відповідає стандартам підприємства щодо оформлення випускних робіт.

4. Календарний графік

№ етапу	Зміст	Термін виконання	Готовність (%), підпис керівника	Підпис керівника про повну готовність етапу, дата
1	Збір практичного матеріалу за темою	16.01.23-17.02.23		
2	І атестація. І розділ випускної роботи	27.03.23-01.04.23		
3	ІІ атестація. ІІ розділ випускної роботи	01.05.23-06.05.23		
4	ІІІ атестація. ІІІ розділ випускної роботи, висновки та рекомендації, додатки, реферат.	29.05.23-03.06.23		
5	Перевірка випускної роботи на оригінальність	15.05.23-12.06.23		
6	Доопрацювання випускної роботи, підготовка презентації, отримання відгуку керівника та рецензії	05.06.23-10.06.23		
7	Попередній захист випускної роботи	12.06.23-18.06.23		
8	Подача випускної роботи на кафедру	за 3 дні до захисту		
9	Захист випускної роботи	19.06.23-24.06.23		

Дата видачі завдання: 22 лютого 2023 р.

Керівник випускної роботи_____ Сабанов С.О.
(підпис)

Завдання прийняв до виконання_____ Резніченко М.А.
(підпис студента)

РЕФЕРАТ

Випускна робота молодшого спеціаліста містить 69 сторінок, 13 рисунків, 2 додатки, 10 першоджерел.

Об’єкт розробки– інтерактивний програмний додаток.

Метою роботи є розробка програмного засобу інтерактивного навчання, що працює на локальному комп’ютері без необхідності використання мережевих ресурсів.

У випускній роботі розглядаються основні принципи та етапи побудови систем, що використовуються в процесі спеціального професійного навчання, а також інструментальні засоби створення подібних ресурсів.

Детально описано процес розробки тематичного програмного комплексу, що містить інформаційно-довідкові матеріали та тестові завдання для вивчення досить специфічної для сприйняття дисципліни – дискретної математики.

Додаток розроблено на з використання мови програмування C# та платформи проектування інтерфейсів користувача WPF.

БАЗАДАНИХ,ДИСКРЕТНА МАТЕМАТИКА, ІНТЕРАКТИВНЕ НАВЧАННЯ, СУБД, C#, MS SQL, WPF

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Слово / словосполучення	Скорочення
C	
C Sharp	CS
M	
Model View View-Model	MVVM
Microsoft	MS
S	
Structured Query Language	SQL
W	
Windows Presentstion Founfation	WPF
X	
eXtensible Application Markup Language	XAML
O	
Об’єктно-орієнтоване програмування	ООП
C	
Система управління базами даних	СУБД

ВСТУП

Сучасний світ суттєво відрізняється від того, що існував ще декілька поколінь назад. Зміни, що відбулись за останні п'ятдесят років перетворили конфігурацію людського життя й значно змінили правила успіху. За окреслений проміжок часу економіка, культура та технології перетерпіли потужних змін, поставивши нові вимоги до суспільства. Третя промислова революція в значній мірі дозріла й людське суспільство в притул підійшло до четвертої промислової революції. Наслідки переходу до нової технологічної конфігурації можна відчути вже зараз, як то нейромережі здатні до швидкого навчання.

Отже, ми живемо в світі швидких технологічних змін. За будь якими технологічними змінами приходять нові виклики та потреби, що постають перед суспільством. Для успішності, суспільству необхідно правильно та швидко адаптуватись до змін, що відбуваються. Для правильного адаптування, необхідно правильно розуміти нові потреби, обумовлені технологічними змінами.

Технологічні зміни не є одномоментними та раптовими. Процес переходу до нової технологічної конфігурації, як вже було вказано, відбувається прямо зараз. Наше суспільство знаходиться на етапі адаптації до вже існуючих змін. Так, до окремих суспільних інститутів висуваються запити на зміни. Зокрема, запит на зміни висувається до професійної підготовки фахівців.

Перед тим як обговорювати запит на зміни до інституцій освіти, необхідно описати загальну проблематику нової технологічної конфігурації та вимоги, що постають перед суспільством вже сьогодні. Далі будуть наведені вимоги що вже стоять та будуть стояти перед людьми в найближчому майбутньому:

- Вміння шукати, аналізувати та сприймати інформацію в різних джерелах.

- Вміння швидко та самостійно навчатись, засвоювати нові технології.
- Вміння відокремлювати необхідну інформацію від навколишнього інформаційного шуму.
- Вміння презентувати себе на ринку праці.
- Вміння оптимізувати свій робочий процес та концентруватися на виконанні поставлених задач.
- Здатність до креативної та оригінальної діяльності.

Як можна помітити, майже всі описані вимоги стосуються роботи з інформацією та навчання. Головним завданням інституту освіти є формування базових економічних навичок, здатності до навчання, уміння сприймати та оброблювати інформацію. Отже, можна зробити висновок, що зміни які відбуваються в економіці та технологічній сфері ставлять великий виклик саме перед інститутом освіти.

Нажаль, в нашій країні інститут освіти, в значній мірі, функціонує на принципах цехового навчання, які були актуальні років сто назад. Це сильніше всього відчувається в сфері молодшої та середньої освіти. З іншої сторони, позитивні зміни відбуваються, не дуже швидко, не повсемісно, але відбуваються.

Технологічні зміни приносять не лише вимоги та випробування, вони надають нові можливості, які зокрема можна використати для вирішення існуючих, або потенційних проблем. Так, нові технології надають можливість організовувати альтернативні та допоміжні освітні методології, як-то дистанційна освіта, інтерактивні освітні засоби, презентаційні засоби. Зокрема, завдяки новітнім технологіям, стало можливим створення інтерактивних навчальних комплексів – програмних засобів, спрямованих на організацію інтерактивного навчального процесу, стосовно певної галузі знань, без прямої участі живого викладача, але з наявністю зворотнього зв'язку, через взаємодію учня з програмою.

Інтерактивні навчальні комплекси – потужний допоміжний інструмент, з високим потенціалом до розвитку у майбутньому, здатний поліпшити функціонування інституту освіти в контексті технологічних змін що відбуваються прямо зараз та будуть відбуватись в найближчому майбутньому.

Ця випускна робота присвячена дослідженню інтерактивних навчальних комплексів та суміжних технологій. Розгляд існуючих рішень, як працюючих в онлайн-режимі, так і локальних, виявив досить обмежену кількість безкоштовних та простих рішень, присвячених, зокрема, саме дискретній математиці. В процесі виконання цієї випускної роботи було створено інтерактивний навчальний комплекс для базового курсу дискретної математики.

РОЗДІЛ 1.
ІНТЕРАКТИВНІ ЗАСОБИ НАВЧАННЯ В ПРОФЕСІЙНІЙ ПІДГОТОВЦІ

1.1. Загальні відомості про інтерактивні засоби навчання

Інтерактивні засоби навчання – технічні або програмні засоби, що створені для виконання доповнюючих функцій під час освітнього процесу, підвищення інтерактивності освітнього процесу, спрощення та покращення сприйняття освітньої інформації [1].

Інтерактивні засоби навчання виступають у допоміжній ролі відносно основного освітнього процесу. В повній мірі, інтерактивні навчальні засоби замінити живий освітній процес не здатні, від них цього й не потребується.

Класичний освітній процес, принаймні в загальноосвітніх навчальних закладах має свої складні для вирішення проблеми. Зокрема, зазвичай кількість учнів у звичайному навчальному класі не дозволяє викладачу приділити достатньо уваги кожному учню, навіть при наявному бажані. Поруч з цією проблемою стоїть пануючий підхід до навчання, який передбачає заучування навчального матеріалу, моделей, правил та методів їх застосування до рівня автоматичної функції організму. При цьому, зазвичай принцип роботи моделей залишається езотеричною загадкою. В результаті, через описану проблему, інститут освіти створює людей, які здатні заучувати певні принципи та використовувати їх у вузьких рамках, при виході за які, використання таких принципів та моделей стає справжнім випробуванням. Описана вище проблема стосується скоріше сфери молодшої та середньої освіти.

Наступною наявною проблемою, якій варто приділити увагу, в контексті дослідження що проводиться в рамках випускної роботи, є надання теоретичного матеріалу поза контекстом його практичного використання. Майже всі моделі, що вивчаються в закладах загальної середньої освіти мають своє практичне застосування, натомість, в тому числі через раніше

описані проблеми, вивчання моделей виступає як щось самоцінне. Про практичне використання вивчених моделей, учні, у кращому випадку, дізнаються сильно пізніше вивчення. Такий підхід не сприяє покращенню сприйняття матеріалу, створює ефект відірваності матеріалу від реальності та відсутності сенсу в навчальному процесі.

Наступною проблемою, якій варто приділити увагу, є недостатня презентаційність матеріалу. Занадто сухий матеріал, позбавлений візуального надання – не найкращий об'єкт для сприйняття молодими людьми. Велика кількість моделей, пов'язаних з алгоритмами, графіками, комбінаторикою, можуть бути представлені у зрозумілій, інтерактивній, геометричній формі, що може значно полегшити сприйняття таких моделей та принципів їх функціонування. Натомість, часто таке представлення відсутнє в силу технічної неоснащеності та непристосованості наявної програми освітнього процесу.

З попередньої проблеми можна вивести більш широку, загальну проблему – проблему сприйняття інформації [2]. Під час навчального процесу, головною задачею інституту освіти є донесення необхідної інформації до учнів, формування в них правильних уявлень про моделі, що вивчаються. Донесення інформації може відбуватися різними методами, які мають різну ефективність. Найпоширенішим таким методом є проведення лекції. Ефективність такого методу сильно залежить від особистих якостей лектора, максимальний рівень ефективності сильно обмежений, оскільки слухання самої лекції є пасивним та прямолінійним способом отримання інформації, що потребує великих вольових зусиль, витримки та уважності. Такий метод діє в основному тільки на слуховий канал сприйняття інформації.

Іншим методом отримання інформації є читання тексту. Такий спосіб має більший потенціал ефективності, оскільки цей процес не лінійний, як-то слухання лекції, а контрольований учнем. Цей спосіб задіює зоровий канал

сприйняття інформації, що для багатьох людей є куди більш потужним інструментом для сприйняття.

Третій метод – перегляд статичних, або динамічних візуальних матеріалів. Цей метод може виступати потужним інструментом для донесення інформації. Він сильно задіює зоровий канал сприйняття інформації та стимулює формування асоціацій, що вкрай позитивно впливає на запам'ятовування інформації. Окрім цього, цей метод дозволяє зображувати у геометричній формі різні моделі точних наук, як-то алгоритми, графіки функцій, множини, графи.

Останнім методом, можливо найефективнішим серед описаниих до цього, є споглядання за тим як хтось інший використовує отримані знання на практиці, ефективність цього методу можна підвищити пояснювальними коментарями з боку того, хто виконує завдання.

В додаток до описаних раніше методів, що є пасивними методами, можна добавити активні методи, що здатні підвищити ефективність донесення інформації. Серед активних методів донесення-сприймання інформації можна виділити писання, що задіює моторику та змушує додатково прикладати зусилля для зрозуміння отриманої інформації. Сильним активним методом донесення-сприйняття інформації є обговорення, що сильно сприяє розумовій діяльності, та спонукає саме до розуміння принципів роботи того, про що йде мова, а не просто повторювання завченого матеріалу, цей ефект можна посилити переведучи обговорення у формат дискусії. Наступний метод є одним з самих ефективних – це виконання практичних задач, з використанням вивчених моделей, ефект від цього методу можна посилити, даючи учням нетипові завдання, для вирішення яких буде необхідне розуміння принципів, за якими функціонує модель, що має бути використана. Багаторазове виконання більш-менш однакових за структурою завдань може позитивно сприяти на запам'ятовування.

Всі вище описані методи донесення-сприйняття інформації можливо комбінувати в різних пропорціях, це може позитивно сприяти на якість освітнього процесу. Але, використовуючи комбінації описаних методів, важливо не створити перегружений масив інформації. Задіювати різні канали сприйняття інформації учнів корисно, але важливо не створити несприймаємий інформаційний шум. Подача інформації має бути достатньо повною, але при цьому лаконічною. Силогічна структура подання вербальної інформації, мабуть є самою оптимальною, за умови того, що учні здатні прослідкувати логічну послідовність у тексті.

Проаналізувавши описані методи, можна прийти до висновку, що інтерактивність у навчальному процесі є вкрай важливим фактором, здатним зробити навчальний процес значно ефективнішим. Інтерактивність у контексті освіти – здатність учнів вчиняти продуктивні активні дії, що мають сприяти кращому засвоєнню та зрозумінню освітньої інформації. Інтерактивність може бути зорганізована завдяки можливості учня отримувати зворотній зв'язок, задавати питання, уточнювати правильність розуміння інформації та завдяки можливості виконувати практичні завдання подані в цікавому, неформальному вигляді. для посилення ефекту інтерактивності можна додавати елемент змагання, це буде чудовим стимулом для проявлення власних знань та використання навичок.

Отже, можна зробити висновок, що підвищити ефективність освітнього процесу можна завдяки наступним чинникам:

- інтерактивність;
- задієння різних каналів сприйняття інформації;
- спрямованість на практичне застосування теоретичних моделей;
- лаконічність подання інформації.

Посилення вказаних чинників у навчальному процесі безумовно є бажаним, й завдяки сучасним технологіям, цьому можна суттєво посприяти.

Інструментом, завдяки якому можна значно покращити якість освітнього процесу та адаптувати інститут освіти до потреб нової технологічної епохи, є інтерактивні навчальні засоби. Цей інструмент може бути чудовим додатковим засобом для процесу навчання. По перше, він здатен в певній мірі спростити роботу вчителів, оскільки інтерактивні навчальні засоби можуть мати функцію автоперевірки, автогенерації завдань, надання необхідних підказок, можуть містити в собі текстовий навчальний матеріал, презентації. Також інтерактивні навчальні засоби можуть сприяти розвитку навичок пошуку інформації та самонавчання. Загалом, інтерактивні навчальні засоби здатні позитивно вплинути на всі чотири чинники, описані раніше в цьому розділі.

1.2. Програмно-апаратні засоби створення інтерактивного середовища

Оскільки потреба в інтерактивних навчальних засобах вже встановлена та підкріплена аргументами в попередньому підрозділі, варто більш конкретно описати, чим саме є інтерактивні навчальні засоби та якими ресурсами необхідно володіти для створення інтерактивного середовища.

В контексті освіти, інтерактивне середовище – це таке середовище, в якому відбувається навчальний процес та яке має значну складову у вигляді інтерактивності.

Інтерактивне середовище, організоване завдяки інтегруванню інтерактивних навчальних засобів є технологічним середовищем. Для його функціонування використовуються комп'ютерні технології, а це передбачає наявність апаратної та програмної складової, що в сукупності й утворюють інтерактивні навчальні засоби.

1.2.1. Технічні засоби інтерактивного навчання

Організація інтерактивного навчального середовища неможлива без використання технічних засобів [3]. В якості технічних засобів інтерактивного навчання можуть бути використані такі.

Персональний комп'ютер – найбанальніший технічний засіб, що може бути використаний для створення інтерактивного навчального середовища. Більшість учнів у розвинених країнах мають базові навички використання комп'ютера, що робить цей засіб простим для засвоєння у контексті інтерактивного навчання. На персональний комп'ютер може бути встановлене різноманітне програмне забезпечення для інтерактивного навчання. Оскільки персональний комп'ютер частіше всього надає можливість для приєднання додаткової апаратури, використання його як технічного засобу інтерактивного навчання може бути вкрай гнучким.

Ноутбук – технічний засіб, що може мати деякі переваги перед персональним комп'ютером. Однією з переваг є відносна мобільність такого технічного засобу.

Планшет або телефони з сенсорним екраном – технічний засіб, яким в порівнянні з попередніми ще більше молодих учнів вміє користуватися. Спектр можливих способів використання у такого технічного засобу є меншим, але такий технічний засіб надає певні можливості, які частіше всього не передбачені двома попередніми технічними засобами. А саме, спрощене та більш інтуїтивне керування, користувацького інтерфейсу на робочу ділянку.

Інтерактивна дошка – сенсорний екран великого розміру. Такий технічний засіб є досить розповсюдженим й можна сказати що це спеціалізований під організацію інтерактивного навчального середовища пристрій. Така дошка може виступати альтернативою класичній твердій дошці, на якій наносять певні символи за допомогою крейди або фламастера. З цим засобом, учні можуть взаємодіяти використовуючи свої руки або

спеціальні сенсорні пера. Цей засіб є зручним, через можливість гнучкого масштабування контенту, розміщеного на екрані.

Засоби доповненої або альтернативної реальності – технічні засоби, що мають перспективу стати головними технічними засобами для створення інтерактивного навчального середовища, але наразі ці засоби мають певні вади, як-то дискомфорт при використанні засобів альтернативної реальності, відносна дороговизна, не найкраща інтуїтивність. Тим не менш, ці засоби вже зараз активно використовуються в освітньому процесі, але здебільш у професійних напрямках, як то військова та медична сфери.

Засоби, що використовують нейроінтерфейси – засоби з надзвичайно високим потенціалом, але наразі вони не є повноцінною іновацією, їх розробка активно ведеться. Такі засоби могли б стати найбільш інтуїтивними серед описаної множини. Їх впровадження могло б стати проривом в сфері навчання. Наразі це технологія нової епохи – епохи четвертої промислової революції, говорити про їх реальну ефективність та супровідні ефекти від їх впровадження ще зарано.

1.2.2. Програмні засоби інтерактивного навчання

Технічних засобів для створення інтерактивних навчальних середовищ замало, другою обов'язковою складовою, необхідною для їх організації є програмні засоби інтерактивного навчання [4]. Вони є спеціальними програмними продуктами, що мають навчальні функції, зокрема можливість автоматичної генерації завдань або тестів, можливість проходження завдань або тестів та автоматичну перевірку результатів виконання завдань або тестів. Крім цього програмні засоби інтерактивного навчання забезпечують зворотній зв'язок для учня, містять в собі необхідну теоретичну інформацію, підказки та детальні пояснення, що можуть подаватися у різних форматах, як-то текст, відео, аудіо, статичні та динамічні зображення, гіперпосилання.

Програмні засоби інтерактивного навчання є досить різноманітними. Зокрема можна виділити такі типи:

1) Пояснювальні – такі що містять в собі теоретичний матеріал в різних поданнях, мають інтерактивний функціонал спрямований на різностороннє пояснення теоретичного матеріалу, демонстрацію можливостей використання такого матеріалу.
2) Практичного застосування – такі, що надають функціонал для практичного використання знань та навичок, перевірки отриманих результатів та певної кількості підказок та пояснень, щодо виконуваних задач або тестів.
3) Змішані – такі, що мають в собі ярко виражені складові обох раніше описаних типів. Робота в таких може бути зорганізована у алгоритмі "проходження теоретичного матеріалу – практична перевірка знань", або за більш складними алгоритмами, як то "спіральний алгоритм" або "ітераційний алгоритм".

Також, програмні засоби інтерактивного навчання можна поділити на вузькоспрямовані, загальнокурсові, загальнодисциплінарні та широкого спрямування. Вузькоспрямовані засоби надають можливість роботи з окремими невеликими підрозділами певної дисципліни. Загальнокурсові – це такі, що надають можливість роботи з цілим навчальним курсом, стосовно певної дисципліни. Загальнодисциплінарні засоби що надають можливість роботи з всеохоплюючою множиною курсів, з якоїсь дисципліни. Засоби широкого спрямування надають можливість роботи з елементами різних дисциплін.

Програмні засоби інтерактивного навчання також можна поділити на такі що працюють в off-line й такі що працюють в on-line.

1.3.Огляд інтерактивних програмних комплексів з математики

1.3.1. Рішення, що працюють off-line

Рішення, що здатні працювати без Інтернет-підключення в окремих випадках можуть містити в собі функціонал, пов'язаний з взаємодією з мережею Інтернет. Тим не менш відсутність Інтернет-підключення не є критичної для використання таких програмних засобів інтерактивного навчання. Частіше всього, такі програмні засоби мають власну локальну базу даних.

Під час виконання випускної роботи було проведено аналіз таких рішень: MATLAB, GNU Octave та Scilab.

MATLAB – великий програмний комплекс, чий функціонал є значно ширшим ніж у звичайного інтерактивного навчального комплекса [5]. Хоча основною задачею цього комплексу є прикладне використання для проведення різних обчислень та моделювання процесів, комплекс має в собі широкий функціонал, який має бути в навчальному інтерактивному комплексі, зокрема є можливість ручного створення моделей з різних математичних напрямків та можливість отримання пояснень щодо таких моделей. Також програма має розвинену систему візуалізації даних. Інтерфейс програми є досить простим та інтуїтивним, але використовувати цей програмний комплекс, принаймні для навчального процесу у загальноосвітніх закладах середньої освіти буде не найкращою ідеєю, оскільки програмний комплекс є занадто всеохоплюючим. Тим не менш, MATLAB можна сміливо використовувати для професіональних цілей не дивлячись на те, скільки років цьому програмному забезпеченню. MATLAB був розроблений у 1984 році. Варто відмітити, що в програмному комплексі використовується спеціалізована однойменна мова програмування, завдяки якій можна досить зручно передавати математичні задачі комп'ютеру.

GNU Octave – програмний комплекс для роботи з різними розділами математики. Комплекс є сумісним з MATLAB, для роботи в ньому

використовується мова програмування схожа на мову MATLAB. Інтерфейс комплексу є трохи складнішим ніж в MATLAB, але область застосування є в певній мірі вужчою. Елементів інтерактивного навчального комплексу у цьому програмному забезпеченні куди менше, але програма може досить успішно використовуватися для виконання складних математичних задач, з подальшою візуалізацією даних. Програмний комплекс був розроблений у 1988 році.

Scilab – мабуть найзручніший аналог комплексу MATLAB. За своїм функціоналом схожий на MATLAB спеціалізований для роботи саме з математичними моделями, але куди більш легковісний. Інтерфейс схожий на MATLAB, мова програмування, що використовується для роботи в комплексі також схожа на мову MATLAB. Плюсом цього комплексу є можливість простого створювання власних типів даних та їх інтегрування у програмну систему.

Як можна побачити, ці програми мають один спільний недолік – їх навряд чи можна назвати саме інтерактивними навчальними комплексами. Безсумнівно, вони можуть бути корисними для освітнього процесу, але скоріше у ролі платформи, завдяки якій, вже обізнаний учень міг би використати на практиці свої знання з математики. Саме для навчання математиці такі програмні засоби навряд чи згодяться.

1.3.1. Рішення, що працюють on-line

Рішення що працюють on-line, можливо використовувати тільки при наявності Інтернет-підключення. Частіше всього це обумовлено тим, що такі застосунки є веб-ресурсами, для роботи з якими критично важливо мати підключення до віддаленої серверної інфраструктури.

Під час виконання випускної роботи було проведено аналіз таких рішень: DiscreteMathTutor та Mathigon.

DiscreteMathTutor – інтерактивний навчальний комплекс, що працює в режимі on-line. Цей комплекс має відносно зручний інтерфейс, велику бібліотеку, що містить в собі велику кількість теоретичного матеріалу, але нажаль має не дуже багато прикладів використання моделей та досить мало пояснень. Крім цього, програмний комплекс надає функціонал для контролю знань, з автоматичною перевіркою результатів виконання завдань. Нажаль цей програмний комплекс не занадто інтерактивний, він скоріше схожий на електронний підручник з дискретної математики, в якому міститься трохи більше пояснень. Окремо варто сказати про відсутність нормальної візуалізації даних та відсутності редактору моделей.

Mathigon – інтерактивний навчальний комплекс, що в певних місцях вигідно виділяється на фоні DiscreteMathTutor. Комплекс виглядає як досить гарно оформлений інтерактивний підручник, з можливістю взаємодії з представленими моделями. Також в комплексі є редактор в якому власноруч можна створити свою модель з різних розділів дискретної математики. Крім цього, комплекс містить в собі функцію уроків, в рамках яких можна отримати та виконати завдання. Ще одним плюсом є наявність розділу з інтерактивними іграми, що спираються на різні моделі дискретної математики. На останок – комплекс має форум, на якому можна поспілкуватися з іншими людьми, що цікавляться дискретною математикою. Єдиним суттєвим мінусом є відсутність пророблено функціоналу для отримання повноцінних завдань.

З вище описаного можна зробити висновок, що інтерактивні навчальні комплекси з дискретної математики, що працюють в режимі on-line представлені куди ширше ніж комплекси що працюють off-line.

1.4. Висновки за першим розділом

Під час виконання початкового етапу роботи було проведено дослідження предметної галузі, яке показало, що успішність нашого суспільства в нову добу четвертої промислової революції зможе забезпечити, наряду з іншими факторами, правильна та своєчасна адаптація інституту освіти. Існуючи методики навчання мають бути вдосконалені, деякі іноваційні методики мають бути впроваджені. Однією з методик, важливих для освіти в майбутньому є використання інтерактивних навчальних комплексів. Ця методика має високий потенціал розвитку, технології, що скоро можливо стануть іноваціями можна буде адаптувати для організації інтерактивного навчального простору. Інтерактивні навчальні комплекси є надзвичайно корисними як доповнюючий освітній інструмент та як інструмент для самоосвіти. Таким чином, можна стверджувати, що діяльність пов'язана з розробкою засобів інтерактивного навчання є перспективною галуззю, що обумовлено попитом на такий продукт, недосконалістю існуючих екземплярів та перспективою інтеграції нових технологій у цю галузь. Особливо це стосується інтерактивних навчальних комплексів з дискретної математики, яка є досить складним розділом математики.

Був проаналізований ринок інтерактивних навчальних комплексів та зокрема інтерактивних навчальних комплексів з дискретної математики. Аналіз показав, що on-line інтерактивні навчальні комплекси з дискретної математики представлені набагато ширше, ніж комплекси що працюють off-line.

Крім того значна частка програм має один спільний недолік – вони не є повноцінними інтерактивними навчальними комплексами. Хоча розглянути додатки можуть бути корисними для освітнього процесу, але акцент в них робиться не на функцію навчання, а на функціонал довідково-прикладного плану, тому саме для навчання математиці такі програмні засоби навряд чи згодяться.

РОЗДІЛ 2.
ІНСТРУМЕНТАЛЬНІ ЗАСОБИ СТВОРЕННЯ ДОДАТКУ

Головним завданням випускної роботи було створення інтерактивного навчального засобу, що невідворотньо передбачало використання певних інструментальних засобів для проектування, програмування та тестування створюваного програмного засобу.

Серед інструментальних засобів, використаних для створення додатку були:

1) Мова програмування.
2) Технологія проектування користувацьких інтерфейсів.
3) Мова розмітки інтерфейсів.
4) Патерн проектування програми.
5) Система управління базами даних.
6) Інтерактивне середовище програмування.

Кожен з цих інструментів був використаний за призначенням, що дозволило створити бажаний програмний засіб. Кожен з використаних інструментальних засобів розглянутий далі у цьому розділі.

2.1. Мова програмування C#

Мова програмування C# була розроблена Microsoft в 2000 році як мова програмування спеціалізована для створення програмних засобів на платформі .NET [6]. Важливою особливістю мови C# є зорієнтованість на програмування в об'єктно-орієнтованій парадигмі. Наразі мова програмування C# є однією з найпопулярніших мов програмування для створення програмних засобів для операційної системи Windows.

Синтаксис мови програмування C# має багато спільностей з синтаксисом мови програмування Java та синтаксисом мови програмування C++. Хоча мова програмування C# не надає стільки ж інструментів для тонкої роботи з системою, керуванням пам'яттю, як C++, але код на мові програмування C# вигідно вирізняється більшою простотою та безпечністю.

Далі будуть надані ключові особливості мови програмування C#, на які варто звернути увагу, приймаючи рішення щодо використання саме цієї мови програмування для створення програмних засобів:

1) Статична типізація даних.
2) Зорієнтованість на ООП.
3) Можливість перегрузки операторів.
4) Автоматичне керування пам'ятю.
5) Опрацювання виключень.
6) Підтримка багатопотоковості.
7) Підтримка асинхронного програмування.
8) Інтеграція мови LINQ.
9) Інтеграція мови XML.
10) Можливість сереалізації даних.
11) Можливість надавати аннотації конструкціям та типам данних.
12) Підтримка великої кількості бібліотек та фреймворків.
13) Можливість створення багатоплатформових програмних засобів.

Варто окремо та більш детально розглянути деякі особливості мови програмування, через наявність яких, саме ця мова була обрана для створення програмного засобу, розроблюваного в рамках виконання випускної роботи. А саме, варто приділити увагу таким особливостям:

– Статична типізація даних - явне оголошення типу даних при написанні програмного коду. Для створення коду розроблюваного програмного засобу, не було жодної потреби у динамічній типізації даних. А отже, використання мови, яка б

базово використовувала динамічну типізацію даних, було б непотрібною надлишковістю.

- Зорієнтованість на ООП – першочергово, мова програмування C# створювалась як спеціалізований інструмент для програмування у об'єктно-орієнтованій парадигмі. Оскільки для створення програмного засобу планувалося використання ООП, вибір мови програмування був зроблений на користь мови C#, як спеціалізованого та зручного інструменту для роботи з ООП, що містить в собі безліч синтактичних інструментів, що значно полегшують роботу в означеній парадигмі програмування.
- Автоматичне керування пам'яттю – програми створені на платформі .NET за допомогою мови програмування C# автоматично виконують виділення та очистку пам'яті, що значно полегшує роботу програміста, оскільки дозволяє не займатися напряму керуванням пам'яттю.
- Опрацювання виключень – ця особливість дозволяє робити програму більш стійкою до непередбачуваних збоїв у роботі. Це може сильно знадобитися, наприклад, для поліпшення валідації даних, які програма може приймати від користувача.
- Інтеграція мови LINQ – мова програмування C# дозволяє використовувати конструкції LINQ для створення запитів до різних джерел даних, при цьому працюючи з ними, як з будь яким внутрішнім джерелом.

Крім вище названих властивостей мови C# варто згадати інтегровані інструменти для розширення існуючих конструкцій. Це значно полегшує роботу над вже створеним програмним кодом, дає можливість не змінюючи існуючий код, розширювати функціонал програми. Для підтримки програмного засобу це однозначно позитивною особливістю.

2.1.1. Використання ООП

ООП – підхід до програмування, що передбачає використання об'єктів (комплексних інформаційних одиниць, що зберігаються в окремому відділі пам'яті що називається купою або хіпом та що можуть зберігати в собі інші дані та інструкції).

Раніше в цьому розділі вже вказувалось що для створення інтерактивного навчального засобу було використано ООП. Варто окремо приділити увагу поясненню рішення про використання ООП для розробки. Вибір в користь ООП був зроблений через наявні позитивні особливості об'єктно-орієнтованої парадигми. Далі буде надано перелік таких особливостей:

1) Можливість інтуїтивного розбиття програми на модулі.
2) Інкапсуляція даних.
3) Генералізація типів даних.
4) Поліморфізм.
5) Висока інтуїтивність коду.

Далі буде наданий більш детальний розбір вище вказаних особливостей:

1) Розробка інтерактивного навчального комплексу з дискретної математики передбачала створення великої кількості різним чином пов'язаних між собою програмних елементів. Це підштовхувало до розбиття розроблюваної програми на окремі модулі з чітко вираженими інтерфейсами, завдяки яким, з модулями можна було б працювати глибоко не вникаючи в їх внутрішній устрій. ООП надає чудові можливості для подібного розбиття програми на модулі й що важливо, завдяки ООП це можливо робити з високим рівнем інтуїтивності.

2) ООП надає можливість налаштовувати внутрішній устрій об'єктів, завдяки чому, розробник має можливість чітко визначати інтерфейси, через які відбувається взаємодія з зовнішніми акторами, відділяючи їх від внутрішніх елементів об'єктів. Інкапсуляція значно розширює можливості для підвищення безпечності програмного коду та підвищує ступінь контролю розробника над розроблюваною системою. Ця особливість тісно пов'язана з попередньою.
3) ООП надає можливість вибудовувати ієрархію спадкоємства типів даних. Це значним чином розширює можливості налаштування поведінки та внутрішньої побудови загалом різних елементів розроблюваної системи. Також, генералізація може відчутно знизити повторюваність коду.
4) ООП надає можливість робити неоднозначною поведінку формально однакових елементів програми. Це надає широкі можливості з налаштування поведінки елементів системи. Також, поліморфізм сильно пов'язаний з генералізацією.
5) Код написаний за принципами ООП має високий рівень інтуїтивності та читаємості, оскільки окремі елементи програми легко асоціюються з окремими речами, що мають певні особливості поведінки, способи використання, окремі складові, властивості.

ООП для розробки інтерактивного навчального комплексу було обрано через загальну зручність у розробці та підтримці програми та зокрема через вище описані корисні особливості.

2.1.2. Використання інструментів для взаємодії з базами даних

Мова програмування C# містить в собі декілька інструментів для зручної взаємодії з базами даних. Оскільки в розроблюваній програмі використовуються бази даних, цей аспект є важливим до розгляду. Серед інструментів для взаємодії з базами даних мова C# підтримує технології ADO.NET, Entity Framework, LINQ. Завдяки останній технології, в код написаний на мові програмування C# можна вписати SQL-подібний код, завдяки якому можна налаштувати взаємодію з базами даних.

Мова C# надає інструменти для забезпечення безпеки даних. Так, на приклад, використовуючи інструменти мови C# можливо створювати параметризовані запити, перевіряти права доступу.

2.1.3. Взаємодія з інтерфейсом користувача

Мова програмування C# має інтегровані інструменти для роботи з інтерфейсами користувача, зокрема маються потужні інструменти для роботи з інтерфейсами, створеними на платформі WPF. Мова C# дозволяє інтегрувати XAML-код в основний код програми, завдяки чому зорганізовується взаємодія між програмною логікою та інтерфейсом користувача. На етапі організації взаємодії між моделлю програми та інтерфейсом відчутно проявляють плюси ООП, зокрема модульність.

2.2. Платформа проектування інтерфейсів користувача WPF

WPF – технологічна платформа для створення інтерфейсів користувача, розроблена компанією Microsoft [7]. WPF використовує мову розмітки XAML, для створення вікон користувацького інтерфейсу, використовуючи декларативний підхід. XAML дозволяє розробнику детально описувати

інтерфейс, представляючи його у вигляді дерева елементів інтерфейсу. Таким чином, розробник створює інтерфейс, як ієрархічну структуру, з якою досить легко працювати через програмний код.

Платформа WPF була обрана для створення інтерактивного навчального комплексу через позитивні особливості цієї технології. Далі буде наданий список позитивних особливостей:

1) Розширюваність.
2) Широкі можливості з налаштування візуалізації.
3) Можливість розділення інтерфейсу та програмної логіки.
4) Підтримка багатозадачності.
5) Інтегрованість з обраним середовищем програмування.
6) Адаптивність

Детальніше про кожен позитивний аспект:

1) Розширюваність – розробник має можливість доповнювати інтерфейс новими елементами, доповнювати поведінку різних елементів інтерфейсу, суттєво не перероблюючи при цьому існуючі елементи. Це значно поліпшує процес підтримки програми, отже є важливим фактором для розробки масштабного проекту.
2) WPF надає можливість детально налаштовувати елементи інтерфейсу та створювати пророблене та приємне візуальне оформлення.
3) WPF надає можливість повністю відокремити інтерфейс від моделі програми, що позитивно впливає на можливість розбиття задачі створення програмного засобу з інтерфейсом на менші підзадачі. Цей фактор описується патерном проектування MVVM.
4) WPF дозволяє створювати програми з інтерфейсом, в яких використовується багатопотоковість.

5) Обране середовище програмування має інтегровані інструменти для зручного створення інтерфейсів користувача на основі платформі WPF, що поліпшує процес створення програмного застосунку.
6) WPF дозволяє створювати інтерфейси, що автоматично підлаштовуються під конфігурації пристроїв виводу, що позитивно сказується на придатність та зручність використання розроблюваної програми.

Отже WPF є зручним інструментом для створення інтерфейсів з великою кількістю можливостей для налаштування візуально відображення та логіки взаємодії з програмною логікою.

2.2.1. MVVM

Раніше було згадано про патерн MVVM. Варто детальніше розглянути цю технологію.

MVVM – патерн архітектури для створення програмних застосунків, що мають інтерфейс користувача. Патерн заточений для використання з платформою створення користувацьких інтерфейсів WPF.

Сутність патерну полягає в розбитті програми на три шари:

1) Модель – шар, що складається з даних, логіки їх обробки та забезпечує взаємодію з зовнішніми джерелами даних. Цей шар повністю відокремлений від шару представлення.
2) Представлення – шар, що являється користувацьким інтерфейсом, повністю відокремленим від шару моделі.
3) Модель-представлення – шар, що зорганізовує двосторонню взаємодію моделі з представленням. На цей шар покладається

задача відображення даних з моделі та обробка взаємодій користувача з інтерфейсом, з подальшим задієнням моделі.

Вибір цього патерну для створення програми є гарним рішенням. Розділення програми на шари значно полегшує процес розробки, через можливість відокремлено працювати над програмною логікою та інтерфейсом.

2.3. СУБД MS SQL

Microsoft SQL Server – система управління реляційними базами даних, розроблена компанією Microsoft [8]. MS SQL використовується для зберігання та обробки даних, а також для підтримки програм, що працюють з базами даних.

MS SQL Server – одна з найпопулярніших систем управління базами даних.

MS SQL Server має певні переваги, ось деякі з них:

1) Надійність та безпека.
2) Масштабуємість.
3) Зручність адміністрування.
4) Багатоплатформовість.
5) Велика та розвинена спільнота.
6) Інтегрованість з іншими інструментами.

При розробці інтерактивного навчального комплексу, важливими факторами були:

– Надійність та безпека – оскільки навчальний комплекс буде використовуватися реальними користувачами, важливо передбачити захист бази даних від некоректного створення та

використання записів у базі даних. MS SQL Server надає достатньо інструментів для забезпечення високої надійності системи.

- Велика та розвинена спільнота – при роботі з MS SQL Server можуть виникати певні незрозумілості та проблеми, допомогти з вирішенням цих проблем може інформація від спільноти.
- Для роботи з MS SQL Server мова програмування C# має достатньо інтегрованих інструментів для зручної взаємодії.

Отже, MS SQL Server є достатнім інструментом, для реалізації необхідного для розроблюваної системи функціоналу з зберігання та отримання даних.

2.4. Середовище розробки

Microsoft Visual Studio – інтегроване середовище розробки, створена компанією Microsoft. Середовище дозволяє розробляти програми для різних платформ.

Microsoft Visual Studio надає велику кількість інструментів для створення програмних засобів з використанням різних мов програмування та інших технологій, як-то платформи для створення інтерфейсів.

Microsoft Visual Studio має зручні інструменти для редагування коду, відладження, аналізу коду, що дозволяє пришвидчувати процес створення програмних застосунків.

Отже, Microsoft Visual Studio є зручним та достатнім за функціоналом середовищем для розробки. Він містить в собі необхідні інструменти для роботи з усіма іншими інструментальними засобами, що задіяні для розробки інтерактивного навчального комплексу з дискретної математики.

2.5. Висновки за другим розділом

Перед початком виконання практичного завдання випускної роботи, важливою задачею був підбір необхідних інструментальних засобів. Обрані технології мали б відповідати потребам, щодо наявного функціоналу та зручності використання.

В першу чергу були визначені дві базові засади, щодо створення інтерактивного навчального комплексу:

1) Використання ООП.
2) Створення гнучкого графічного інтерфейсу.

Отже, підбір інструментальних засобів для створення програми, на першому етапі, полягав у пошуку мови програмування, зручної для об'єкто-орієнтованого програмування та платформи для створення інтерфейсів користувача, з широкими можливостями налаштування.

В якості мови програмування була обрана мова C#, оскільки вона є спеціалізованою мовою для ООП з досить простим та зручним синтаксисом. Крім цього, в користь цієї мови зіграли інші переваги, що вона здатна надати.

В якості платформи для створення інтерфейсів користувача була обрана платформа WPF, оскільки вона надає велику кількість інструментів для детального налаштування окремих елементів інтерфейсу. Крім цього, важливим фактором є інтегрованість цієї платформи з мовою програмування C#.

Наступним етапом став підбір системи управління базами даних. Вибір був зроблений у користь СУБД MS SQL Server, оскільки ця технологія має достатній функціонал та є відносно легкою у використанні. Ще одним фактором , важливим для вибору було те, що обрана мова програмування має зручні інтегровані інструменти для роботи зокрема і з цієї СУБД.

Після визначення основних технологічних засобів етап визначення споміжних. Так, в якості патерну проектування програми був обраний

шаблон MVVM. Такий вибір був зроблений через зручність та точність патерну, його інтуїтивність, високу придатність для використання з вже обраними технологіями.

В останню чергу, було обрано середовище розробки. Через наявність інструментів для роботи з усіма раніше обраними технологіями, загальну зручність та доступність, вибір був зроблений у користь середовища Microsoft Visual Studio.

Як виявилось, при створенні інтерактивного навчального комплексу, вибір такого стеку інструментальних засобів був правильний. Робота з описаними інструментами була зручною, вони були здатні надати необхідний для створення програми функціонал.

РОЗДІЛ 3.
РЕАЛІЗАЦІЯ ІНТЕРАКТИВНОГО НАВЧАЛЬНОГО ДОДАТКУ "DisMath Assistant"

Після виконання таких етапів як вивчення теоретичного матеріалу та підбору необхідних інструментальних засобів, настав час приступити до практичного виконання головного завдання цієї випускної роботи – створення інтерактивного навчального комплексу з дискретної математики [9].

В першу чергу, був проведений аналіз вимог стосовно розроблюваної програми. Так було визначено, що програма в кінцевому вигляді має бути якомога більш оптимізованою, легковаговою та зрозумілою для використання непідготовленим користувачем.

Оскільки тема випускної роботи стосується такої дисципліни як дискретна математика та оскільки дискретна математика є розгалуженою дисципліною, що стосується великої кількості формальних систем, з великою кількістю матеріалу, було прийнято рішення чітко окреслити підрозділи та формальні системи дискретної математики, з якими розроблена система буде надавати можливість працювати. Було прийняте рішення, в рамках виконання випускної роботи, обмежитись такими підрозділами дискретної математики:

- Булева алгебра.
- Пропозиційна логіка.
- Теорія множин.
- Теорія відношень.
- Теорія графів.

Вибір саме такої комбінації підрозділів дискретної математики був не випадковим. Конкретні пояснення, щодо вибору підрозділів дискретної

математики для використання у програмі, в якості модельної бази, надані в підрозділі записки до випускної роботи №3.1.

Після визначення підрозділів дискретної математики, функціонал для роботи з якими буде мати розроблювана програма, був проведений аналіз можливих функцій, що мають корисний навчальний потенціал, задля подальшого визначення конкретних методів виконання програми. За результатами проведеного аналізу, було вирішено обрати наступну комбінацію функціоналу:

- Створення та перевірка завдань.
- Створення та перевірка тестів.
- Ручне створення моделей з обраних формальних систем.
- Обчислення на моделях з обраних формальних систем.

Наступними етапами, після вибору функціоналу, стали вибір патернів проектування класів та методів оптимізації алгоритмів роботи з даними а також вибір підходу до проектування бази даних та визначення способів взаємодії між програмою та базою даних. На цьому етапі було визначено, що основною задачею бази даних буде зберігання створених моделей з формальних систем, зберігання створених задач та тестів. На останньому етапі підготовки до процесу створення інтерактивного навчального комплексу з дискретної математики було проведено встановлення конкретних вимог до інтерфейсу користувача. Зокрема, були встановлені такі вимоги:

- Достатність при максимальній простоті.
- Наявність підказок щодо використання програми.
- Наявність підказок, щодо формальних систем дискретної математики.
- Інтуїтивність.

– Обмеженість, що здатна убезпечити програму від непередбачених дій користувача, що могли б призвести до виникнення помилок в роботі програми.

3.1. Призначення та функціональні можливості додатку

Проаналізувавши назву класу програми, що має бути створена в рамках виконання цієї випускної роботи, приходимо до висновку, що програма повинна мати такі властивості:

1) Інтерактивність.
2) Навчальну цінність.
3) Комплексність.
4) Причетність до дискретної математики.

Окремо розберемо кожну названу властивість, за рахунок чого програма буде наділена цими властивостями.

1) Інтерактивність – програма повинна давати користувачу зрозумілу інформацію, як-то пояснення щодо способів експлуатації, пояснення щодо теоретичного матеріалу, стосовно якого користувач веде діяльність в програмі й саме головне, програма має давати широкий та корисний відгук щодо дій користувача. Важливо забезпечити користувача зворотньою інформацією, щодо його діяльності.
2) Навчальна цінність – програма має функціонувати таким чином, щоб забезпечувати покращення знань та навичок користувача, стосовно того навчального матеріалу, якому присвячена програма, або окремий її модуль. Для забезпечення навчальної цінності програми, необхідно використовувати перевірені методики навчання, враховуючи навантаження на різні канали сприйняття інформації та різну ефективність методів навчання.

3) Комплексність – програма має бути розділеною на окремі елементи, за ознакою функціонального призначення. Користувач повинен мати можливість обирати з яким функціоналом він буде працювати, весь інший надлишковий функціонал не повинен заважати користувачу.
4) Причетність до дискретної математики – головним функціональним призначенням розроблюваної програми є покращення процесу вивчення дискретної математики, як розділу математики в цілому та як дисципліни, що вивчається в вищих навчальних закладах України зокрема.

За допомогою чого програма буде наділена описаними властивостями?

1) Інтерактивність– для забезпечення наявності цієї властивості, програма буде мати особливі елементи користувацького інтерфейсу, такі як підказки щодо актуальних моделей формальних систем, підказки щодо окремих елементів управління інтерфейсу користувача. Крім цього, інтерфейс буде відразу відображати реакцію на всі дії користувача, щодо редагування моделей, в зрозумілій формі, підкріплюючи відображення підказками, за необхдності.
2) Навчальна цінність – програма буде містити в собі набір функціоналу, що забезпечить можливість застосування певних навчальних методів, таких як: виконання готових завдань, проходження тестів, власноручне створення моделей формальних систем та використання останніх, наприклад, для проведення розрахунків, виконання задач. Враховуючи те, що програма не міститиме в собі відео, аудіо матеріалу та анімованих зображень, важливою складовою, необхідною для забезпечення програми високим рівнем навчальної корисності, буде доцільне, достатньо інформативне та легке для сприйняття відображення інформації на інтерфейсі користувача. Окремо, необхідно забезпечити

можливість приховування та розвернення окремих масивів інформації, це необхідно для позбавлення інтерфейсу перевантаженості. Позбавлення інтерфейсу перевантаженості важливо, через те, що перевантаженість поскладнює сприймаємість інформації, що йде в розріз з головним функціональним призначенням розроблюваної системи.

3) Комплексність – наявність цієї якості буде забезпечена завдяки розбиттю програми на окремі модулі такі як: редактори моделей, калькулятори, генератори задач, генератори тестів, модулі для проходження задач та модулі для проходження тестів. Всі ці модулі будуть пов’язані між собою головним модулем програми. Завдяки такому підходу до конструювання програми, користувач зможе використовувати функціонал програми, необхідний йому для вирішення тих задач, які йому необхідно вирішити.
4) Причетність до дискретної математики – ця якість програми буде забезпечена завдяки наявності функціоналу для різноманітної взаємодії з різними моделями формальних систем, що входять до складу дискретної математики. Серед формальних систем будуть доступними для роботи: булева алгебра, пропозиційна логіка, теорія множин, теорія відносин, теорія графів. Як було вказано на початку цього розділу записки, вибір саме такої комбінації формальних систем не є випадковим, він буде пояснений одразу після того, як будуть розглянуті ці системи по окремості, в цьому підрозділі.

Тепер розглянемо обрану комбінацію формальних систем.

3.1.1. Булева алгебра в контексті навчального комплексу

Булева алгебра – формальна система, що входить до складу дискретної математики, в межах якої вивчають різні типи взаємодії між булевими значеннями.

В контексті цієї формальної системи відбувається оперування такими математичними об'єктами як:

- Булеве значення.
- Булева функція.

В контексті розроблюваної програми, моделлю булевої алгебри називається булева функція.

Взаємозв'язок між вказаними математичними об'єктами забезпечується за допомогою логічних операцій, різної арності, як-то:

- Кон'юнкція.
- Диз'юнкція.
- Матеріальна імплікація.
- Заперечення.

В цілому, логічних операцій є величезна кількість. В розроблюємій програмі будуть присутні тільки унарні та бінарні операції, список останніх буде трохи ширшим за той, що надається в рамках звичайного курсу дискретної математики в Українських вищих навчальних закладах.

Стосовно математичних об'єктів, що використовуються в цій формальній системі необхідно враховувати дві аксіоми:

1) Будь яке булеве значення має істино-хибну цінність, булеве значення в певний момент є або хибним, або ж істиним, ніколи не одночасно й обов’язково одним з них. Істинність або хибність часто відображається за допомогою цифр 1 та 0 відповідно.
2) Будь яке булеве значення може бути представлене у вигляді булевої функції й навпаки.
3) Булеві функції можуть мати арність (кількість аргументів), довжину (кількість операцій, що проводяться в межах цієї

функції) та можуть містити в собі константні булеві значення. Булеві функції відображаються у вигляді формул, складених з обмеженого різномаїття символів, які іноді різняться. Так, наприклад, дуже часто оператори логічних операцій відображаються за допомогою різних символів.

4) Булеві функції з однаковою арністю та за однакових умов (за однакових значень аргументів), при однаковому істинно-хибну цінність є еквівалентними, взаємозамінними.

Важливим інструментом для роботи з булевою алгеброю є таблиця істинності. Вона широко використовується в розроблюємій програмі як інструмент представлення моделей булевої алгебри та як керувальний інструмент користувача. Корисність таблиці істинності булевої функції полягає у представлені всіх кроків виконання відповідної булевої функції при всіх можливих унікальних комбінаціях значень аргументів функції, якщо такі наявні. У випадку з нульарними функціями, таблиця істинності такої функції буде мати один рядок.

Булеві функції можуть бути представлені у різних формах, як-то:

- Проста форма – в такій формі можуть бути надмірні кроки, вона може містити будь які логічні операції.
- Досконала кон'юнктивна нормальна форма – в такій формі формула є максимально оптимізованою при тому що в ній використовуються тільки операції заперечення, кон'юнкції та диз'юнкції. ЇЇ особливістю є те, що формула функції є кон'юнкцією диз'юнкцій аргументів у прямому або запереченому вигляді.
- Досконала диз'юнктивна нормальна форма – в такій формі формула є максимально оптимізованою при тому що в ній використовуються тільки операції заперечення, кон'юнкції та диз'юнкції. ЇЇ особливістю, на відміну від СКНФ є те, що

формула функції є диз'юнкцією кон'юнкцій аргументів у прямому або запереченому вигляді.

– Алгебраїчна нормальна форма або поліном Жегалкіна – в такій формі формула функції є максимально оптимізованою при тому, що в ній використовуються тільки операції кон'юнкції та сильної диз'юнкції, крім цього, може використовуватися константна істина.

Будь яка логічна операція може бути представлена через іншу операцію або послідовність операцій. Формула функції може бути конвертована з однієї форми до іншої. Це надає можливість для оптимізації функцій та представлення формули функції через послідовність бажаний операцій. Останнє може бути корисним при створені електросхем та в багатьох інших галузях.

Розроблювана програма моє в собі функціонал для створення булевих функцій, їх обчислення конвертації до нормальних форм та порівняльного аналізу різних булевих функцій. Також програма ефективно використовує можливість представлення булевої функції через булеве значення для оптимізації вирахувань та загалом алгоритмів роботи з булевими функціями.

Важливим елементом розроблюваної програми є транслятор формул булевої алгебри. Завдяки цьому елементу, програма може транслювати дані строкового формату у булеву функцію, зрозумілу для комп'ютера.

3.1.2. Пропозиційна логіка в контексті навчального комплексу

Пропозиційна логіка – формальна система, що входить до складу дискретно математики, в межах якої вивчаються взаємозв'язки між пропозиціями (висловленнями, що можуть мати істино-хибну цінність, як це можуть булеві значення). В контексті пропозиційної логіки, пропозиції

можуть бути представленими як у вигляді вимовлень на природній мові так й у вигляді вимовлень на формальних значень.

В контексті розроблюваної програми, трансляція вимовлень у неформальній формі та зворотній процес не розглядається.

Моделлю пропозиційної логіки у контексті розроблюваної програми вважається пропозиція.

В контексті пропозиційної логіки відбувається оперування такими математичними об'єктами:

- Пропозиція.
- Пропозиційна функція (складна пропозиція).

Кожна пропозиція може бути представлена у вигляді пропозиційної функції та навпаки.

Пропозиційні функції формально не відрізняються від функцій булевої алгебри, тож їх можна взаємозамінювати, що продуктивно використовується в програмі.

Вцілому, пропозиційна алгебра оперує тими ж самими логічними операціями, що й булева алгебра, але зазвичай в пропозиційній алгебрі використовуються не всі логічні операції.

Вивчання пропозиційної логіки може бути корисним для покращення навичок аналізу інформації, комунікації вцілому та риторичного мистецтва в часності.

3.1.3. Теорія множин в контексті навчального комплексу

Теорія множин – формальна система, що належить до дискретної математики, в межах якої вивчаються множини, як об'єкти що сприймаються цілісними.

В контексті навчального комплексу, моделлю теорії множин вважається система множин або як її ще можуть називати – простір задачі теорії множин.

В контексті теорії множин, відбувається оперування такими математичними об'єктами:

- Множина – об'єкт мислимий одним цілим, що має свою властивість інклюзії та який може мати в собі жодного, один, скінчену або нескінчену кількість елементів, кожен з яких обов'язково має властивість інклюзії відповідної множини. В системі множин може існувати множина, яка є пустою. Так відбувається тоді, коли в межах простору задачі міститься властивість, яка є властивістю інклюзії для відповідної множини, але простір задачі не містить в собі жодного елементу, який володів би відповідною властивістю.
- Елемент множини – дискретний об'єкт, що має жодну, одну або більше однієї властивості. Приналежність елементу до властивості залежить від наявності відповідної властивості.
- Властивість – певна характеристика об'єкту.

Над множинами можуть проводитись операції різної арності. Операції одиничної арності зазвичай називають операціями редагування, вони призводять до отримання нової множини, що є відредагованою множиною-аргументом. Бінарні функції призводять до отримання нової третьої множини.

Множини є еквівалентними, якщо їх властивість інклюзії є однаковою, в наслідок чого набір їх елементів буде однаковим. Якщо в системі множин є еквівалентні множини, їх спрощення через видалення надмірних множин не призведе до небажаних змін.

Для представлення системи множин можуть використовуватися різні методи, наприклад:

- Строкове перераховування елементів.
- Діаграма Ейлера.

В контексті розроблюваного навчального комплексу використовується метод строкового перерахування множин, елементів та властивостей.

Вивчення теорії множин є корисним для подальшого вивчення дискретної математики та для роботи з інформацією, статистикою. Зокрема, теорія множин несе корисну пояснювальну функцію для широкого загалу наукових дисциплін.

3.1.4. Теорія відношень в контексті навчального комплексу

Теорія відношень– формальна система, що належить до дискретної математики, в межах якої вивчаються відносини як дискретні об'єкти, що містять властивості.

В контексті навчального комплексу, моделлю вважаються системи відносин або ж простори задач теорії множин.

В контексті теорії відношень, відбувається оперування такими математичними об'єктами:

- Відношення – дискретний об'єкт, за допомогою якого описується взаємодія між певною кількістю об'єктів. Відношення мають арність, вони можуть бути нульарними, унарними, бінарними та з арністю більше двох. Відношення можуть бути накладені на окрему комбінацію елементів або на цілі множини елементів.
- Множина.
- Елемент.
- Властивість відношення.
- Властивість елемента множини.

В контексті навчального комплексу, використовуються тільки бінарні та унарні відношення. Бінарні відношення описують взаємодію між двома конкретними об'єктами. В той час, унарні відношення, по факту, описують наявність або відсутність певної властивості у об'єкта.

Для роботи з бінарними відношеннями частіше всього використовуються такі засоби як таблиці сміжності, на яких відображають декартове накладення двох еквівалентних або різних множин, на яких в залежності від вступу пари елементів у відношення у відповідному порядку вказується бінарне значення (вступає / не вступає). Цей інструмент продуктивно використовується в програмі для відображення інформації про актуальну систему відношень та для керування користувачем системою відношень.

В контексті навчального комплексу, відношення накладені на множини задаються у вигляді таблиць сміжностей, списків елементів повязаних з елементом та у вигляді сміжних пар.

Самі ж відношення в навчальному комплексі можуть створюватись через керовану генерацію відношень або через індуктивний аналіз заданої таблиці суміжності.

Будь яке бінарне відношення має набір обов'язкових властивостей, що описують:

- Рефлексивність відношення.
- Симетричність відношення.
- Транзитивність відношення

Крім вище названих, відношення можуть мати додаткові властивості, певні з яких можливо отримати при індуктивному аналізі таблиці суміжності, інші ж можна отримати через аналіз базових властивостей.

В рамках теорії відношень, над відношеннями можуть виконуватись унарні та бінарні операції, через виконання яких отримуються нові відношення.

Вивчення теорії відношень є важливим для розуміння реляційної алгебри, створення баз даних, подальшого вивчення дискретної математики, розвитку аналітичних навичок.

3.1.5. Теорія графів в контексті навчального комплексу

Теорія графів – формальна система, що є важливою та складною частиною дискретної математики, в рамках якої вивчаються графи, методи їх аналізу та методи їх перетворень.

В контексті навчального комплексу, моделлю теорії графів вважається система графів або ж простір задачі теорії графів. Модель теорії графів містить в собі множину верхівок та множину графів, в межах яких існують ребра, що певним чином сполучають верхівки. При цьому, модель теорії графів може працювати з графами тільки одного типу з наступних:

- Неорієнтовані незважені графи.
- Неорієнтовані зважені графи.
- Орієнтовані незважені графи.
- Орієнтовані зважені графи.

В контексті теорії графів використовуються наступні математичні об'єкти:

- Граф – множина верхівок, певним чином сполучених множиною ребер.
- Верхівка графу – дискретний математичний об'єкт, що може бути сполученим з іншими верхівками за допомогою ребер.
- Ребро графу – дискретний математичний об'єкт, завдяки якому сполучується пара різних або еквівалентних верхівок графа.

Для представлення графів у навчальному комплексі використовуються таблиці суміжності, таблиці інцидентності, списки верхівок, списки ребер,

списки інцидентних ребер та списки інцидентних верхівок. Ці конструкції також використовуються як елементи керування.

Програма містить в собі функціонал для створення моделей теорії графів, створення графів, обробки графів та аналізу графів.

Над графами можуть бути виконані унарні та бінарні операції, що призводять до отримання нового графа. Крім цього, над графами мужуть бути проведені складні операції, як-то пошук шляху між верхівками.

Вивчення теорії графів є складною але важливою задачею. Розуміння цієї формальної системи може бути корисним для діяльності у галузі логістики, створенні електросхем для опису моделей попередніх розглянутих формальних систем.

Отже, як можна було помітити, описані підрозділи дискретної математики сильно пов'язані між собою, їх вивчення за вказаною послідовністю може бути більш продуктивним, ніж інші підходи. Вивчення кожного нового підрозділу здатно покращити знання про попередні вивчені підрозділи.

Як було вказано на початку цього розділу, програма, розроблювана в межах виконання випускної роботи є модульною, тобто розділеною на модулі, згідно їх функціональним призначенням. Далі буде наданий опис окремих модулей та їх функціоналу.

3.1.6. Редактори моделей

Ці модулі призначені для ручного створення користувачем моделей конкретних формальних систем. У випадку з булевою алгеброю та пропозиційною логікою, редактори будуть тісно суміжними з калькуляторами.

Перелік функцій цих модулів:

- Створення нових моделей.

- Запис моделей до базі даних
- Редагування існуючих моделей.
- Видалення існуючих моделей з базі даних.
- Редагування записів бази даних, стосовно конкретних моделей.
- Використання моделей інших формальних систем, там де це доцільно.

Для створення моделей булевої алгебри та пропозиційної логіки будуть використовуватися функції суміжні з калькуляторами для цих формальних систем. Задавати їх можна буде створюючи формулу функцій, яка завдяки влаштованому підмодулю транслятору формул буде перетворюватися на моделі відповідних формальних систем.

Всі дії щодо редагування будуть супроводжуватися динамічним відображенням моделей.

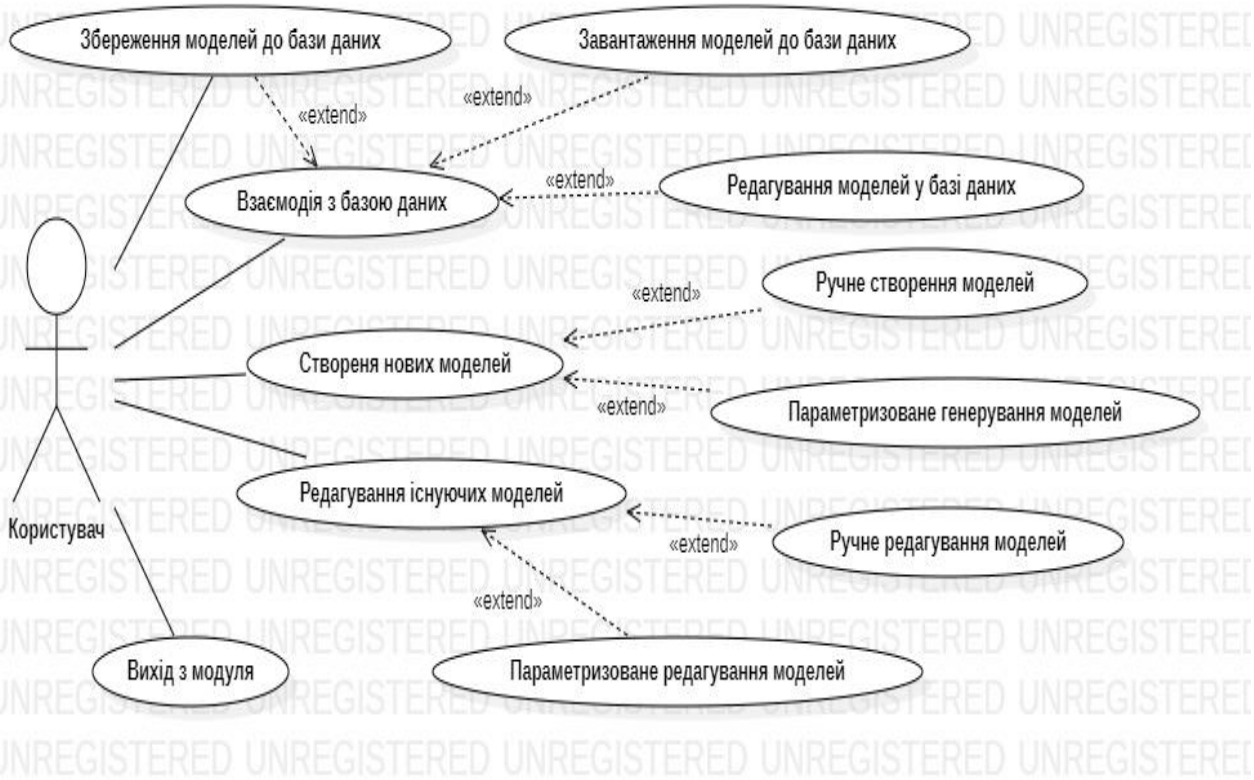

Рисунок 3.1 – Діаграма варіантів використання редакторів

3.1.7. Калькулятори

Ці модулі призначені для виконання обчислень та вирішення задач за допомогою конкретних існуючих моделей.

Ці модулі мають свої влаштовані підмодулі для задання формул, що описують операції над моделями та подальшої трансляції формул у функції.

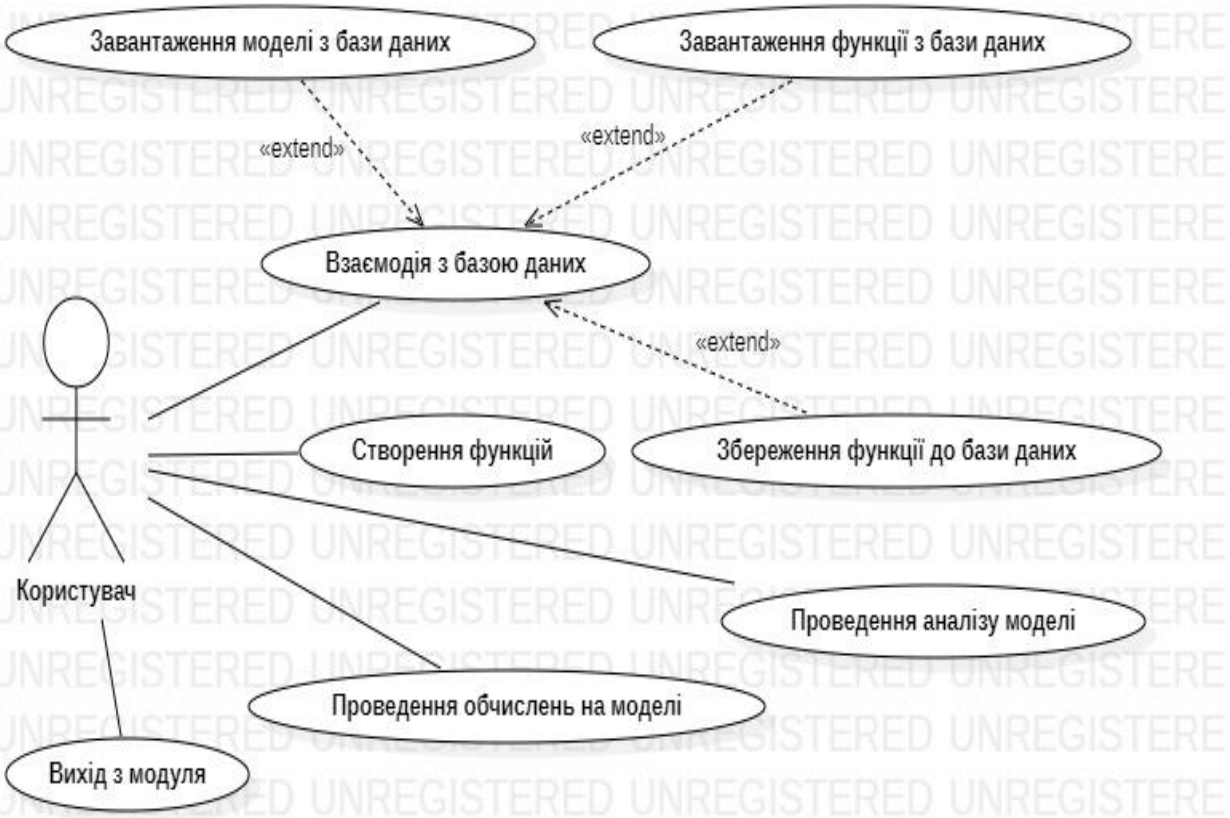

Рисунок 3.2 – Діаграма варіантів використання калькуляторів

3.1.8. Генератори завдань та генератори тестів

Ці модулі дозволяють генерувати завдання та тести з відповідних підрозділів дискретної математики, з можливістю ручного налаштування генератора.

Згенеровані задачі, тести та налаштування генераторів можна зберігати до бази даних. Або завантажувати налаштування генераторів з бази даних.

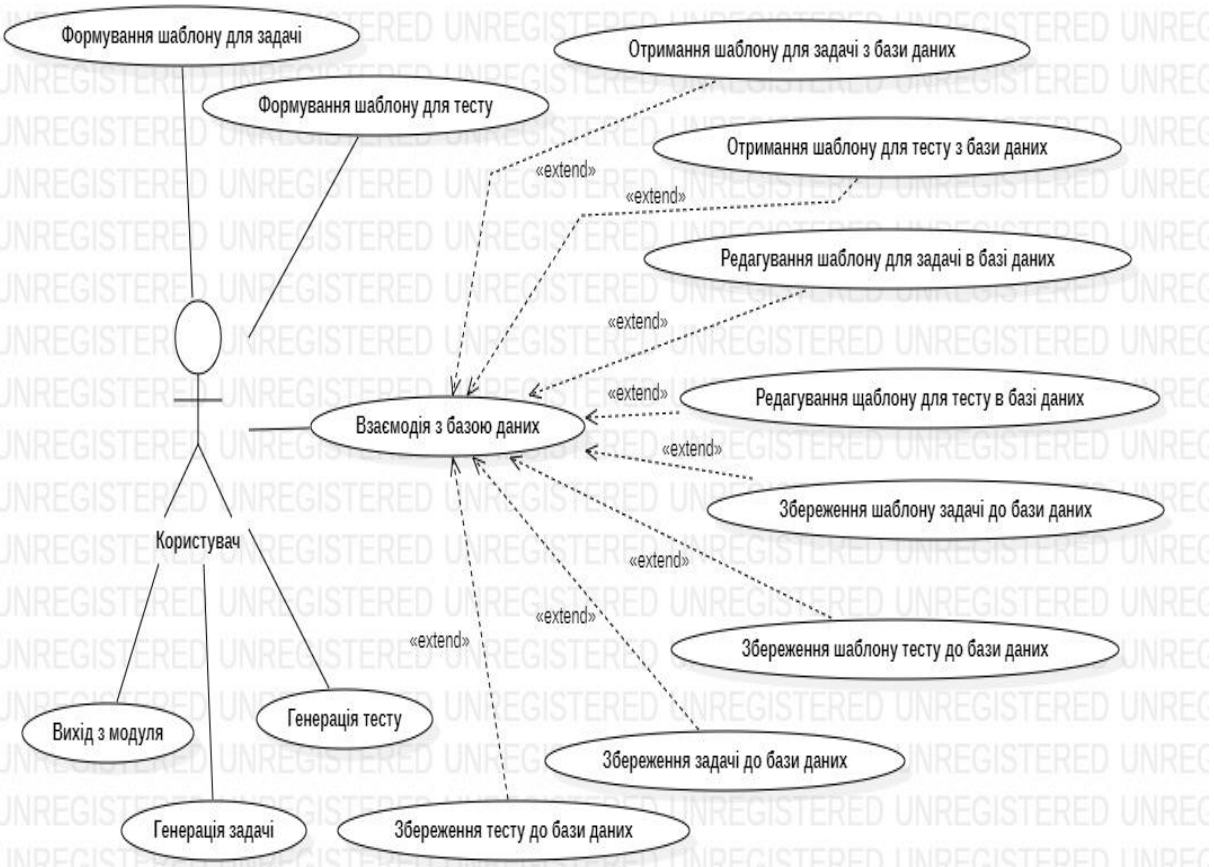

Рисунок 3.3 – Діаграма варіантів використання генераторів задач та тестів

3.1.9. Модулі для перевірки виконання завдань та тестів

Ці модулі дозволяють виконувати згенеровані завдання та тести, з можливістю завантаження завдань або тестів з бази даних, з подальшою перевіркою результатів та отриманням пояснень та підказок.

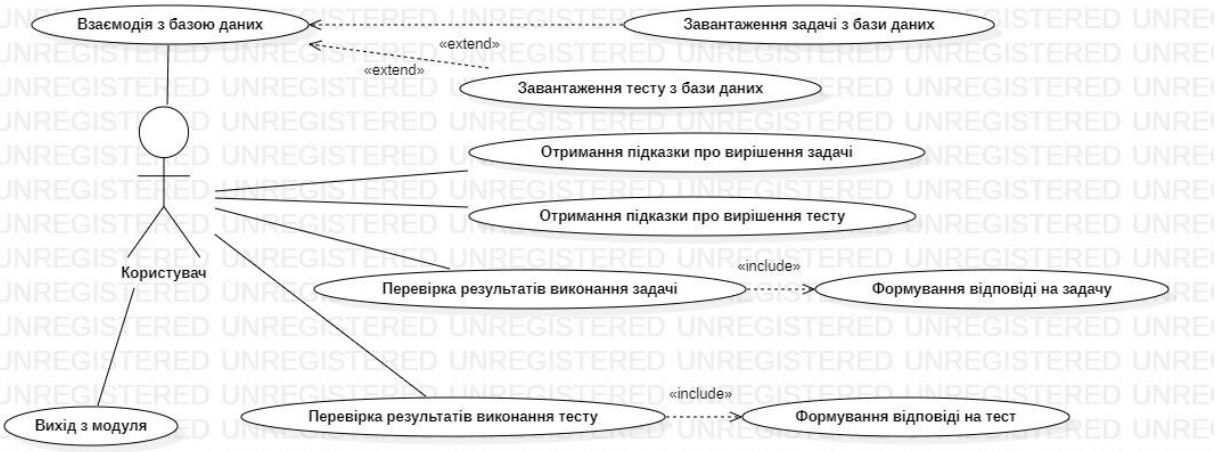

Рисунок 3.4 – Діаграма варіантів використання модуля для виконання задач та тестів

3.1.10. Головний модуль навчального комплексу

Цей модуль призначений для отримання доступу до всіх інших модулів комплексу та для взаємодії з базами даних. Зокрема, завдяки цьому модулю можна переглянути існуючі записи бази даних та видалити існуючі за потреби.

Рисунок 3.5 – Діаграма варіантів використання головного модуля навчального комплексу

3.2. Блок-схема та алгоритм роботи

Для організації роботи програми з моделями та об'єктами дискретної математики, були створені спеціальні типи даних. Головними вимогами до таких, під час розробки, були достатність та інтуїтивність. Так наприклад, для роботи з формальною системою теорії графів, були створені типи:

- Система графів.
- Граф.
- Верхівка графу.
- Ребро графу.

Такий набір типів, дозволив зручно працювати з різними графами, виконувати на них необхідні алгоритми та виконувати над графами утворювальні операції.

Приклад класової структури для представлення моделей деяких підрозділів дискретної математики наданий в рисунках 3.6, 3.7, 3.8.

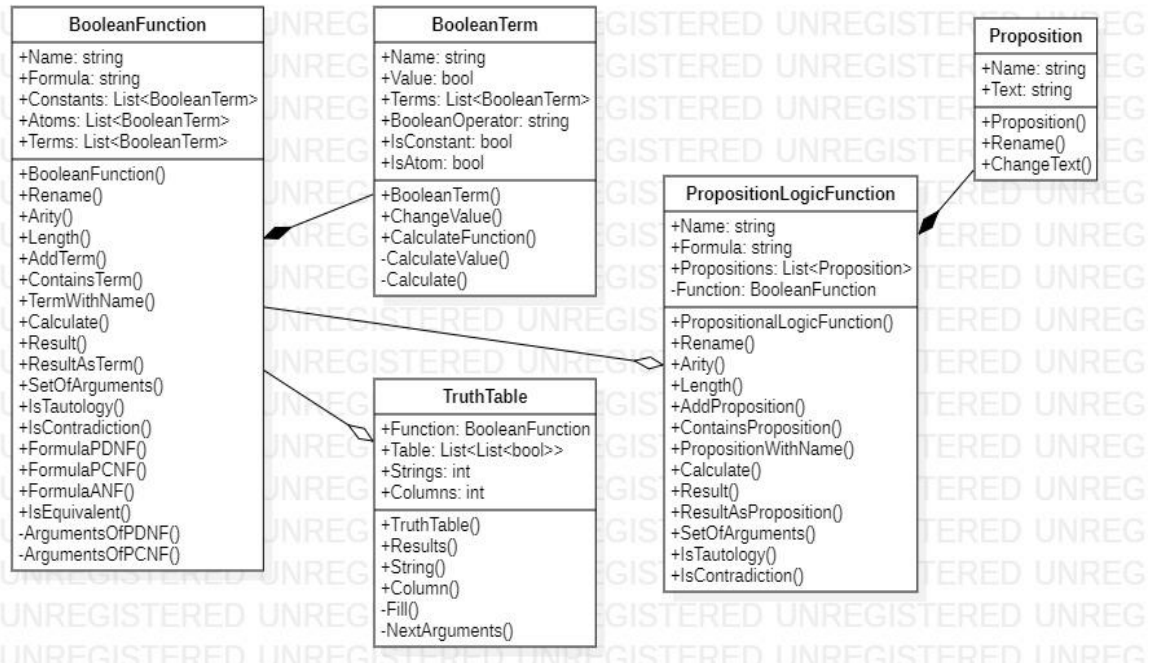

Рисунок 3.6 – Діаграма класів для представлення моделей булевої алгебри та пропозиційної логіки.

Оскільки пропозиційна логіка в формальній площині нічим не відрізняється від булевої алгебри, для роботи з нею більшою мірою використовуються типи даних створені для роботи з булевою алгеброю.

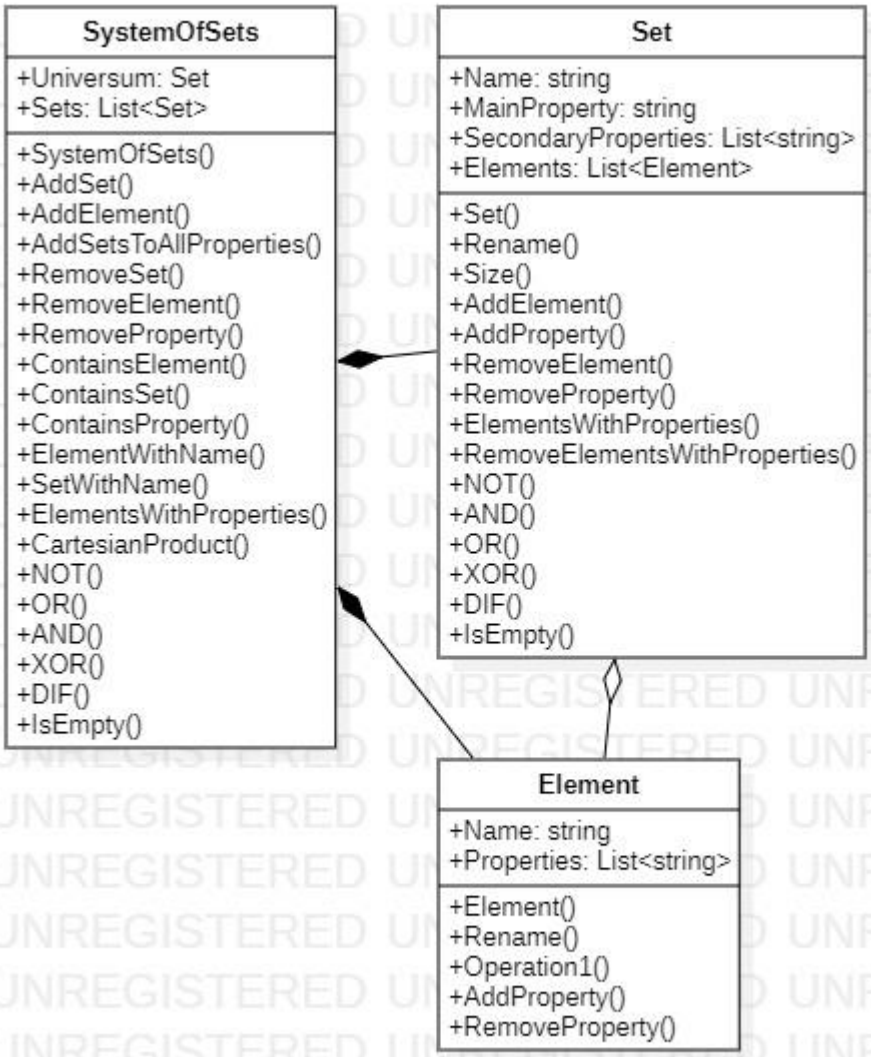

Рисунок 3.7–Діаграма класів для представлення моделей теорії множин

Робота з множинами зорганізована через наповнення системи множин елементами та властивостями, на базі яких створюються конкретні множини пов'язані з наявними властивостями. В системі множин, над множинами можуть виконуватися різноманітні операції, що волічуть за собою створення нових множин.

Для роботи з теорією відношень використовуються не тільки типи, створені виключно для роботи з цим підрозділом дискретної математики. Так, наприклад, для роботи з відношеннями накладеними на множини, використовуються типи, створені для роботи з теорією множин.

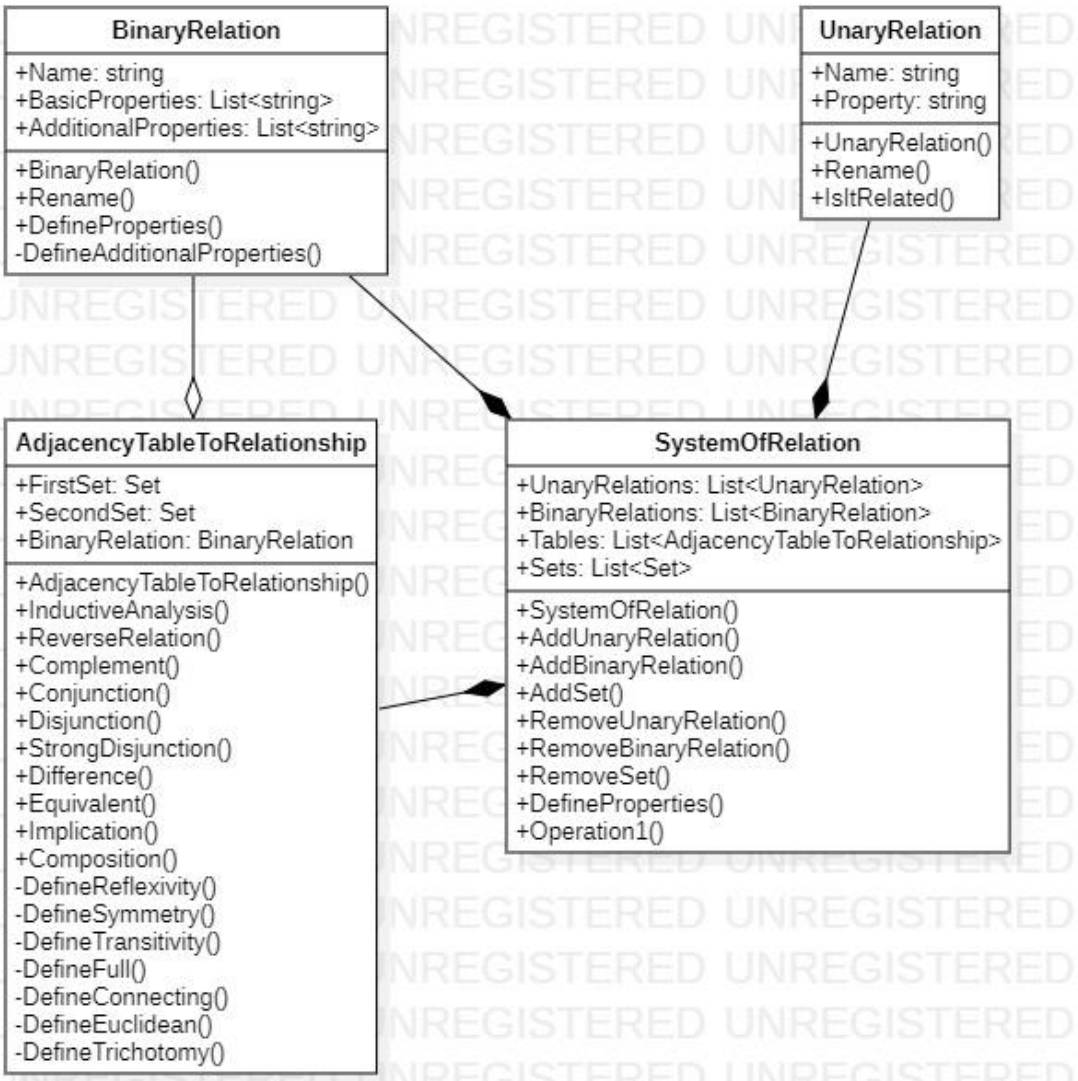

Рисунок 3.8 – Діаграма класів для представлення моделей теорії відношень

Під час роботи програми, об'єкти класів, вказаних вище, не з'являються незвідки. Отримати їх можливо завантаживши з бази даних, або створивши власноруч. Так ,наприклад, для того щоб отримати булеву функцію, необхідно ввести її формулу використовуючи функціонал спеціального

редактора булевих функцій. Отримана формула є звичайною строкою символів, яку програма зможе перетворити на робочу булеву функцію, за допомогою спеціального підмодуля, транслятора булевих функцій. Пропоную детальніше розглянути побудову класу, завдяки якому транслятор булевих функцій виконує свою роботу. Спрошене представлення класу транслятора булевих функцій:

```
public static TranslatorOfBooleanFunction()
{
private static string _formula;
private static BooleanFunction _function;

public static BooleanFunction Translate(string formula);
public static bool CheckValidity();

private static void FormatFormula();
private static void PrioritizeStaps();
private static void WriteTerms();
private static void WriteStep();
}
```

Функціональним призначенням транслятора булевих функцій є створення об'єктів (булевих функцій) на основі наданих строкових даних. Детальніше розглянемо функціональне призначення методів класу транслятора.

- Метод Translate – приймає строкове значення, в якості результату повертає зібрану булеву функцію , за умовою того, що формула має коректне наповнення. В процесі виконання цього метода, послідовно виконуються етапи роботи над формулою та створюваною функцією. Цей метод викликає методи, що будуть описані нижче.
- Метод CheckValidity – виконує перевірку введеної формули на коректність вмісту, правильність використаних у формулі символів та правильність семантики формули. У тому випадку, що формула має валідний зміст, клас продовжує виконувати трансляцію формули у функцію.

- Метод FormatFormula – цей метод приводить формулу до вигляду, позбавленого надлишковими символами, що не є невалідними. Після виконання цього методу, транслятор приступає до виконання наступного методу.
- Метод PrioritizeStaps – цей метод виконує покрокову пріоритезацію операцій майбутньої функції, завдяки розстановленню пріоретизуючих дужок там де вони відсутні, згідно з приоритетністю логічних операцій. Цей метод є рекурсивним, він виконується до тих пір, поки кожній операції майбутньої функції не буде наданий пріоритет.
- Метод WriteTerms – Цей метод виконує створення об'єкту булевої функції та вписуваня в неї всіх атомарних термів, що містяться в формулі.
- Метод WriteStep – цей метод виконує покрокове рекурсивне записування операцій до функції, згідно з заданим порядком приоритетності виконання операцій. Цей метод виконується до тих пір, поки кожна операція не буде записана до функції в означеному порядку.

3.3. Структура бази даних

Головним функціональним призначенням бази даних, розроблюваної програми, є зберігання моделей задіяних формальних систем. Зв’язок таблиць бази представлено на рисунку 3.9.

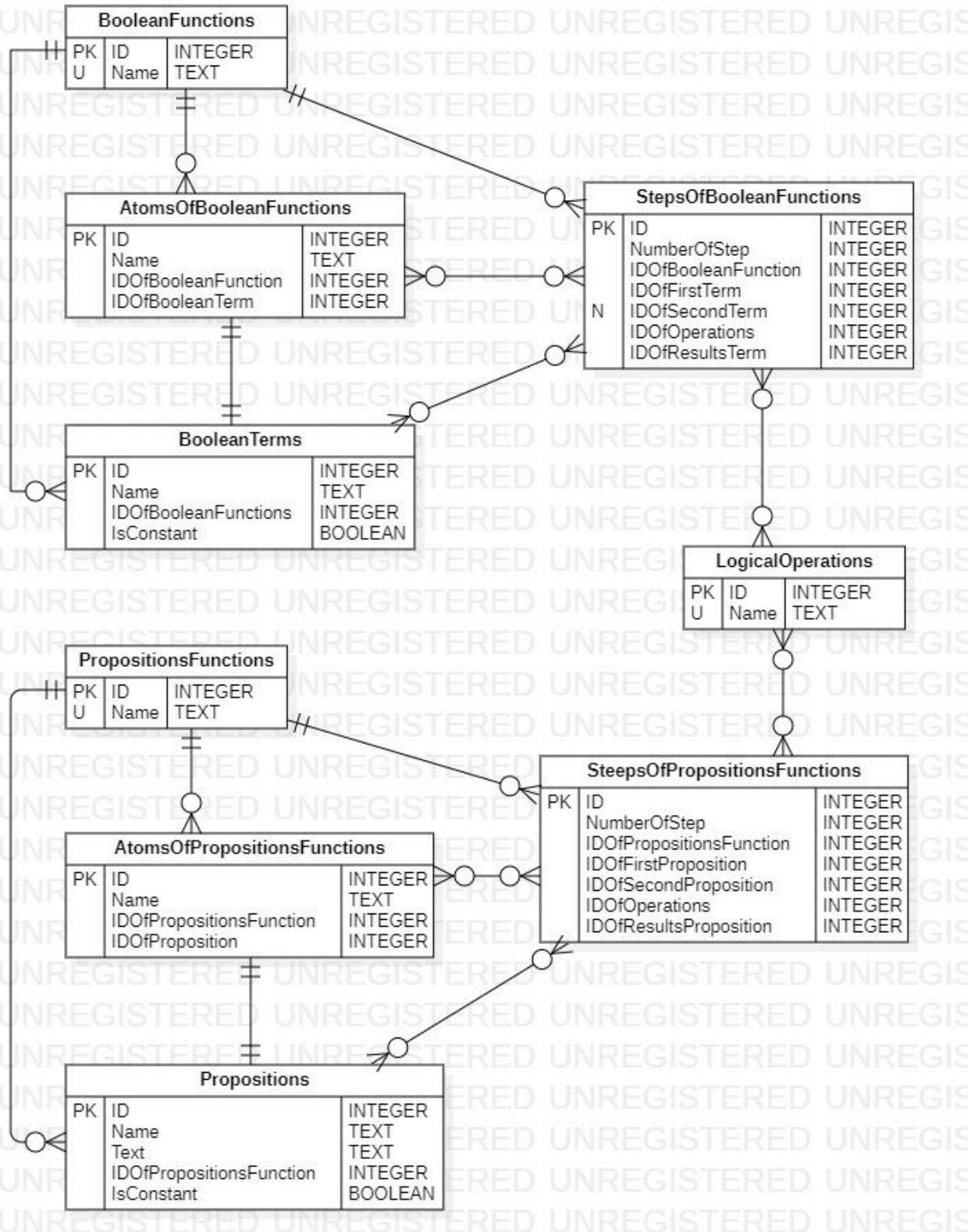

Рисунок 3.9 – Таблиці бази даних, що забезпечують роботу з моделями булевої алгебри та пропозиційної логіки

Цей сегмент бази даних відповідає за зберігання моделей булевої алгебри та пропозиційної логіки. При загрузці певної моделі, програма інтерпретує отримані записи бази даних та утворює в задіяній оперативній пам'яті об'єкти необхідні для роботи з відповідними моделями.

Як можна побачити, окремі унікальні елементи моделей, одночасно належать одним моделям.

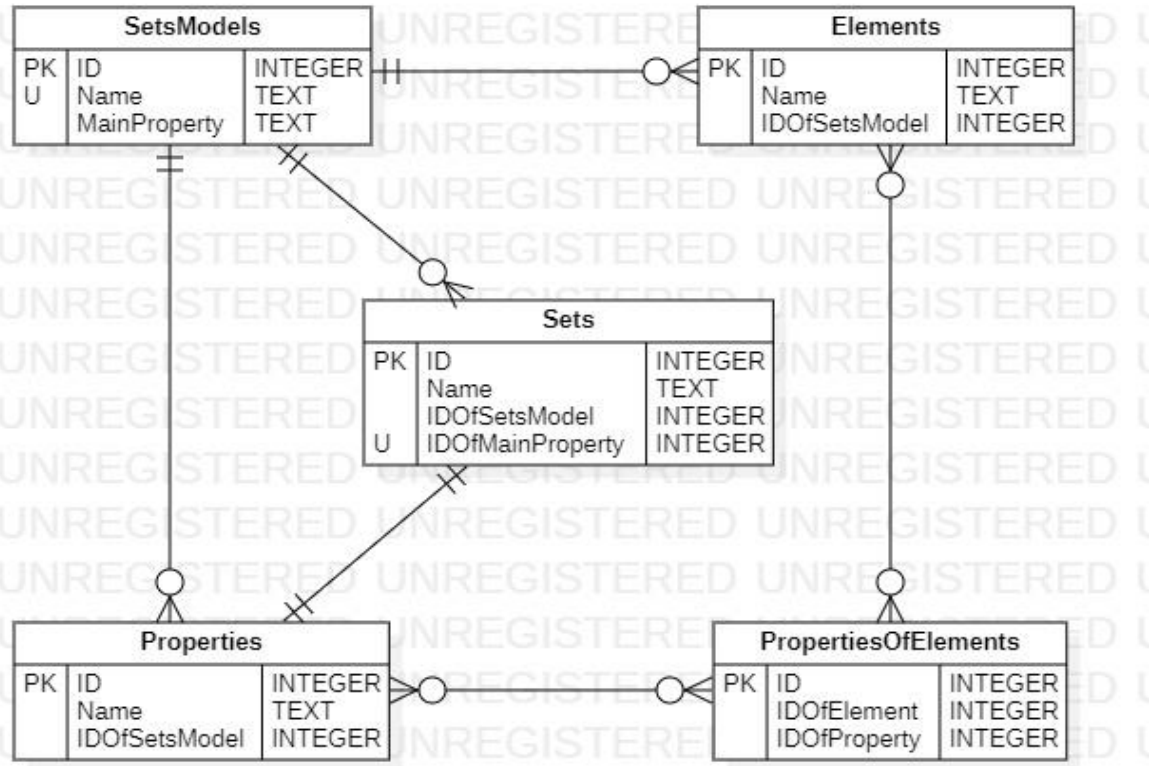

Рисунок 3.10 – Таблиці бази даних, що забезпечують роботу з моделями теорії множин

В контексті зберігання в базі даних, моделі теорії множин пов’язані з унікальними наборами елементів, унікальними наборами множин та унікальними наборами властивостей, жоден з цих унікальних наборів, як й окремі їх елементи, не пов'язані з іншими моделями теорії множин.

Як можна побачити, таблиці множин та елементів не мають прямого зв'язку. Формування об'єктів множин відбувається під час інтерпретаціїданих

програмою в набір об'єктів формальної системи теорії множин. Формування об'єктів множин, в системі множин, спирається на наявні властивості, притаманні елементам, пристнім в системі множин.

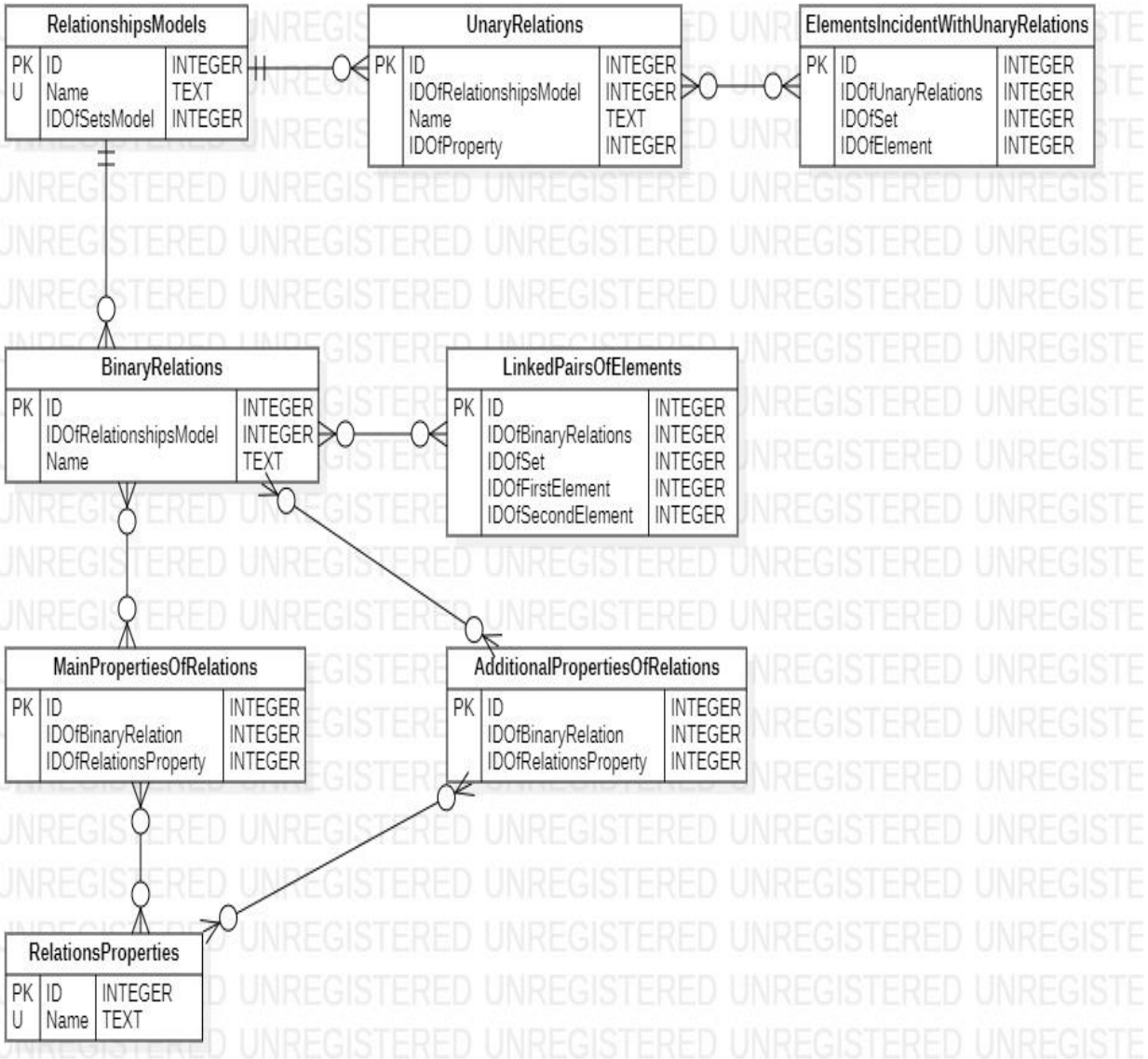

Рисунок 3.11 – Таблиці бази даних, що забезпечують роботу з моделями теорії відношень

Сегмент, що відповідає за зберігання моделей теорії відношень, тісно пов'язаний з сегментів, відповідальним за зберігання моделей теорії множин.

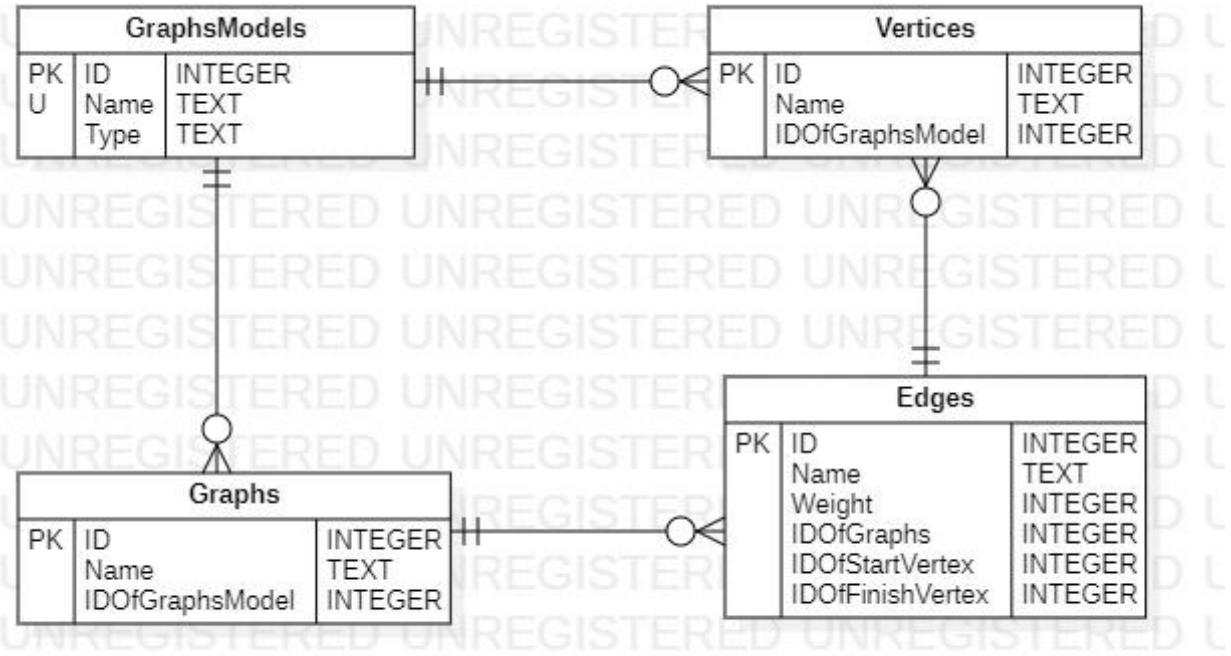

Рисунок 3.12 – Таблиці бази даних, що забезпечують роботу з моделями теорії графів

Як було вказано на початку розділу записки №3, програма надає можливість роботи з моделями теорії графів, в яких містяться графи лише одного типу. В базі даних, моделі графів зберігаються уніфіковано. В контексті роботи програми, при загрузці моделі теорії графів з бази даних, графи конкретних типів отримуються завдяки спеціальному процесу інтерпретації уніфікованих даних, отриманих з бази даних, що спирається на маркер типу, який містить в собі запис моделі графу.

Робота програми з базами даних зорганізована через групу класів, що містяться в шарі моделі програми. Ці класи виконують перевірку існуючих записів бази даних, можливості створення нових записів для зберігання визначених даних програми. Також вони відповідають за процеси зберігання даних до бази даних та отримання моделей з бази даних, з подальшою їх трансляцією в об'єкти відповідних типів.

3.4. Інтерфейс користувача

Процес створення інтерфейсу користувача був важливою та місткою роботою. По-перше, він полягав у створенні шару відображення для чого були виконані наступні кроки:

1) Верстка вікон інтерфейсу користувача.
2) Створення стилів для оформлення елементів інтерфейсу користувача.
3) Дизайн вікон інтерфейсу користувача.
4) Налаштування ідентифікаторів керувальних елементів інтерфейсу користувача.
5) Створення логіки поведінки керувальних елементів інтерфейсу користувача.
6) Створення інтерфейсів доступу до керувальних елементів для з'єднання з моделлю представлення.
7) Налаштування ідентифікаторів відображальних елементів інтерфейсу користувача.
8) Створення логіки поведінки відображальних елементів інтерфейсу користувача.
9) Створення інтерфейсів доступу до відображальних елементів інтерфейсу користувача.
10) Налаштування зв'язків між вікнами інтерфейсу користувача.

При створенні інтерфейсу користувача, акцент був зроблений на функціональну простоту інтерфейсу та уніфікованість дизайну всіх вікон інтерфейсу. При створенні дизайну, використовувалася легка палітра кольорів, що мало б полегшити сприйняття візуальної інформації. Приклад застосування дизайну надано на рисунку №3.13.

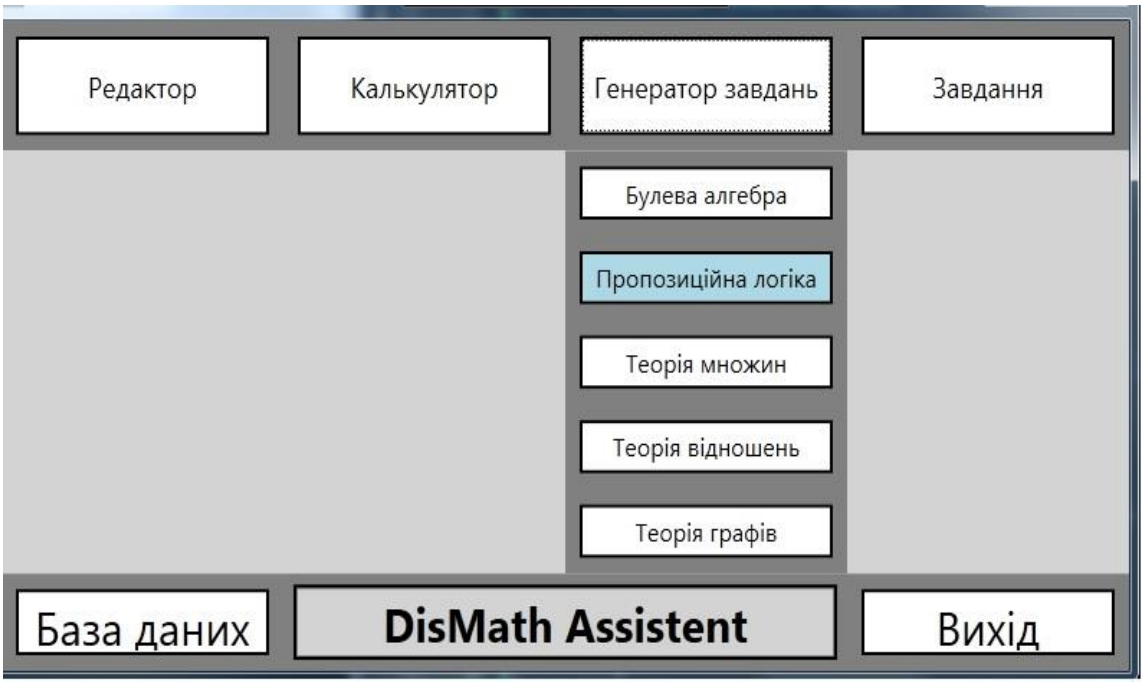

Рисунок 3.13 – Інтерфейс головного меню програми

Після створення шару представлення, та часткового створення шару моделі-представлення, була проведено створення моделі-представлення, що полягало у виконанні наступних кроків:

1) Створення класів для взаємодії між базою даних, через шар моделі, та інтерфейсом користувача.
2) Створення класів для керування модулями програми.
3) Створення класів для взаємодії з даними моделі через інтерфейс користувача.

3.5. Створення завдань та моделей

3.5.1. Створення моделей

Для створення моделей булевої алгебри та моделей пропозиційної логіки використовуються спеціальні модулі. Вони мають чітко обмежений функціонал, що полегшує процес створення моделей. А саме, ці модулі

надають можливість добавляти та видаляти з простору актуальної задачі аргументи функції, для цього використовується калькуляторoподібна панель. У випадку з пропозиційною логікою, додається можливість запису конкретного змісту пропозицій у вільному режимі, з клавіатури. Наступним етапом створення моделей є запис формули функції за допомогою складання такої з елементів розміщених на панелі доступних елементів формули та панелі з списком операторів. Крім цього, функції можна надати унікальне ім'я. При записі формули, відбувається підсвічування помилок та відображаються відповідні підказки.

Якщо створена формула є коректною, розблокується можливість створення та збереження функції до бази даних. Окрім збереження моделі до бази даних, користувач має можливість завантажити модуль з бази даних та створити на його основі нову, або редагувати й перезаписати.

У випадку з теорією множин, використовується набір інструментів, за допомогою яких можна добавляти та видаляти до простору задачі елементів, множин та властивостей. При створенні множини є обов'язковим вказання властивості інклюзії. Кожен елемент має як мінімум одну властивість притаманну множині-універсуму відповідної моделі. При видалені властивості з простору задачі, відбувається видалення цієї властивості з всіх елементів у просторі задачі та видаляється множина для якої, видаляєма властивість є властивістю інклюзії. При додаванні властивості до елементу, при наявності множини у просторі задачі, чиєю властивістю інклюзії є властивість, додаваєма до елементу, відповідний елемент приписується до відповідної множини. При наявності в просторі задачі множини, властивістю інклюзії якої не володіє жоден елемент, присутній у просторі задачі, відповідна множина є порожньою множиною.

У випадку з теорією відносин, для створення бінарних відношень можуть бути використані такі методи як пряме створення через задання властивостей відношення або через індуктивний аналіз декартового накладання множень, належних до підключеної моделі множин.

Для створення моделей теорії графів використовується спеціальний інструмент, що надає функціонал для додавання та видалення верхівок до простору задачі. Графи в просторі задачі існують як унікальні об'єкти. Ребра, що поєднують між собою верхівки існують тільки в межах конкретного графу. Тобто, в контексті моделі теорії графів, граф виступає інструментом для зберігання ребер, існування верхівок, що містяться в просторі задачі, напряму не залежить від графів, моделі теорії графів.

3.5.2. Створення завдань

Створення завдань полягає у випадковій генерації програмою моделей, відповідних формальних систем, відштовхуючись від шаблону завдання.

Шаблони завдань є окремими типами, об'єкти яких містять в собі інформацію про майбутньо згенероване завдання. В шаблоні міститься інформація стосовно того, якими параметрами будуть володіти моделі, згенеровані для формування завдань.

Шаблони завдань створюються в модулях генерації завдань.

Шаблони завдань можуть бути добавлені або видалені або відредаговані в базі даних.

Завдання, згенеровані програмою можуть бути добавлені або видалені або відредаговані в базі даних.

3.5.3. Створення тестів

Створення тестів включає в себе етап створення завдання. Але, створення тестів відбувається з використанням шаблонів тестів. Головною відмінністю процесу створення тестів від процесу створення завдань є додатковий, кінцевий етап, що полягає в генеруванні варіантів відповідей на

завдання, серед яких є як мінімум одна вірна відповідь та певна кількість неправильних відповідей, що вже залежить від конкретного шаблону для генерування тестів.

ВИСНОВКИ

Під час виконання випускної роботи молодшого спеціаліста, була проведена комплексна робота з дослідження теоретичного матеріалу, що стосується теми випускної роботи, а конкретно: дискретної математики, навчальних методів, методів вивчення дискретної математики, інтерактивних навчальних комплексів в цілому та інтерактивних навчальних засобів для вивчення дискретної математики зокрема.

Після проведення пошуку та дослідження матеріалу, були обрані підрозділи дискретної математики, для роботи з якими, розроблювана програма матиме функціонал. А саме, були обрані такі підрозділи дискретної математики: булева алгебра, пропозиційна логіка, теорія множин, теорія відношень, теорія графів.

Наступник кроком у виконанні випускної роботи був вибір навчальних методів, які будуть реалізовані в розроблюваній програмі. Були обрані такі методи: інтерактивне створення моделей, надання завдань, надання тестів, надання інтерактивних засобів для проведення обчислень, з використанням моделей.

Другим важливим етапом виконання випускної роботи був вибір інструментальних засобів для створення навчального комплексу. З міркувань легкості використання та високої взаємної інтегрованості, був обраний наступний стек технологічних засобів: мова програмування C#, платформа для створення інтерфейсів WPF, патерн проектування MVVM, система управління реляційними базами даних MS SQL Sercer та інтегроване середовище розробки Microsoft Visual Studio.

Кінцевим кроком, перед початком створення програми стало визначення головних вимог до програми. Серед останніх було обрано: легкість для засвоєння користувачем, навчальна корисність.

Першим кроком в створенні програми стала побудова діаграм та схем алгоритмів роботи майбутньої програми.

Після проведення проектування, був створений шар моделі програми, що складався з класів для представлення об'єктів формальних систем дискретної математики, класів для організації логіки роботи модулів калькуляторів, генераторів задач, редакторів й інший, класів для роботи з базою даних.

Після закінчення роботи над шаром моделі, була проведена робота зі створення бази даних. Були створені сутності бази даних, налаштовані зв'язки між сутностями та наприкінці, були дороблені класи програми для роботи з базами даних.

Наступним етапом в розробці програми було створення шару представлення. На цьому етапі була проведена робота з верстки вікон користувацького інтерфейсу, налаштований дизайн, створені класи для керування інтерфейсом.

Передостаннім етапом розробки програмного зсатосунку стало створення шару моделі представлення, завдяки якому відбувається вся взаємодія між шаром представлення та шаром моделі. Цей етап був найважчим, оскільки саме на цьому етапі вирішувалось, якою програма буде в результаті.

Останнім етапом створення інтерактивного навчального комплексу з дискретної математики стало тестування та виправлення помилок.

В результаті виконання випускної роботи було створено інтерактивний навчальний комплекс з дискретної математики "DisMath Assistent", здатний бути корисним для вивчення дискретної математики. Були поліпшені знання з дискретної математики та методів навчання. Були отримані корисні навички з проектування програмних систем, програмування, роботи з системами управління реляційними базами даних.

ПЕРЕЛІК ПОСИЛАНЬ

1. Що таке інтерактивні засоби навчання? [Електронний ресурс] / - Режим доступу: www. URL: https://ivo.kneu.edu.ua/ua/dosl_glot/proj_soit/interaktiv/.
2. Про сенсорний аспект навчального процесу. [Електронний ресурс] / - Режим доступу: www. URL: https://nus.org.ua/view/multysensorne-navchannya-zrozumity-uchniv-i-navchaty-po-novomu/.
3. Про технічні засоби, необхідні для організації інтерактивного навчального процесу. . [Електронний ресурс] / - Режим доступу: www. URL: https://naurok.com.ua/stattya-misce-tehnichnih-zasobiv-navchannya-u-pedagogichnomu-procesi-79123.html.
4. Про програмні засоби, необхідні для організації інтерактивного навчального процесу. [Електронний ресурс] / - Режим доступу: www. URL: https://naurok.com.ua/interaktivni-ta-proektni-metodi-navchannya-na-urokah-informatiki-156828.html.
5. Про програмний комплекс MATLAB. [Електронний ресурс] / - Режим доступу: www. URL: https://www.mathworks.com/products/matlab.html.
6. Стаття про мову програмування C#/ . [Електронний ресурс] / - Режим доступу: www. URL: https://edu.cbsystematics.com/ua/blog/learn-csharp-become-dnet/
7. Керівництво з використання WPF. [Електронний ресурс] / - Режим доступу: www. URL: https://metanit.com/sharp/wpf/.
8. Керівництво з використання MS SQL Server. [Електронний ресурс] / - Режим доступу: www. URL: https://metanit.com/sql/sqlserver/1.1.php.
9. Навчально-методичний посібник з дискретної математики. [Електронний ресурс] / - Режим доступу: www. URL: https://probability.knu.ua/userfiles/yamnenko/manual_DM.pdf

Додаток А

Частина вихідного коду бібліотеки DSModels

```
public class SystemOfSets
{
    public string Name { get; private set;
}
    public Set Universum { get; private
set; }
    public List<Set> Sets { get; private
set; }

    public SystemOfSets()
    {
        Name = "New model of Set
Theory";

        Universum = new("Universum",
"P0");

        Sets = new();
    }

    public void Rename(string name)
    {
        if(CheckName(name))
        {
            Name = name;
        }
    }

    public void AddSet()
    {
        if(Sets.Count < 10)
        {
            Sets.Add(new(NameToSet(),
NameToProperty()));
        }
    }

    public void AddProperty()
    {
        if (Properties().Count < 10)
        {
            Sets.Add(new(NameToSet(),
NameToProperty()));
        }
    }

    public void AddElement()
    {
        if(Universum.Elements.Count <
30)
        {

Universum.AddElement(new(NameToElement(),
Universum.MainProperty));
        }
    }

    public void
AddElementToSet(Element element, Set set)
    {
        if(set != Universum)
        {
            set.AddElement(element);
        }
    }

    public void
AddPropertyToElement(string property, Element
element)
    {
        if
(SetWithMainProperty(property) != null)
        {

SetWithMainProperty(property).AddElement(elemen
t);
        }
    }

    public void RemoveSet(Set set)
    {
        Sets.Remove(set);


Universum.RemoveElements(Universum.ElementsW
ithProperty(set.MainProperty));
    }

    public void RemoveProperty(string
property)
    {
        if(SetWithMainProperty(property)
!= null)
        {

Sets.Remove(SetWithMainProperty(property));
        }
    }

    public void
RemoveElement(Element element)
    {

Universum.RemoveElement(element);

        foreach(Set set in Sets)
        {
            set.RemoveElement(element);
```

```
            }
        }

        public void
RemoveElementFromSet(Element element, Set set)
        {
            set.RemoveElement(element);
        }

        public void
RemovePropertyFromElement(string property,
Element element)
        {

element.RemoveProperty(property);

            if(SetWithMainProperty(property)
!= null)
            {

SetWithMainProperty(property).RemoveElement(ele
ment);
            }
        }

        public Set Negation(Set set)
        {
            Set result = new($"Negation of
{set.Name}", $"Not {set.MainProperty}");


result.AddElements(Universum.ElementsWithoutPro
perty(set.MainProperty));

            return result;
        }

        public Set AND(Set set1, Set set2)
        {
            Set result = new($"Conjunction of
{set1.Name} and {set2.Name}",
$"{set1.MainProperty} AND {set2.MainProperty}");

            foreach(Element element in
set1.Elements)
            {

if(element.ContainsProperty(set2.MainProperty))
                {
                    result.AddElement(element);
                }
            }

            return result;
        }

        public Set OR(Set set1, Set set2)
        {
            Set result = new($"Disjunction of
{set1.Name} and {set2.Name}",
$"{set1.MainProperty} OR {set2.MainProperty}");

result.AddElements(set1.Elements);

result.AddElements(set2.Elements);

            return result;
        }

        public Set XOR(Set set1, Set set2)
        {
            Set result = new($"Strong
disjunction of {set1.Name} and {set2.Name}",
$"{set1.MainProperty} XOR {set2.MainProperty}");


result.AddElements(set1.ElementsWithoutProperty(s
et2.MainProperty));

result.AddElements(set2.ElementsWithoutProperty(s
et1.MainProperty));

            return result;
        }

        public Set Difference(Set set1, Set
set2)
        {
            Set result = new($"Difference of
{set1.Name} and {set2.Name}",
$"{set1.MainProperty} without
{set2.MainProperty}");


result.AddElements(set1.ElementsWithoutProperty(s
et2.MainProperty));

            return result;
        }

        public bool HaveIntersection(Set
set1, Set set2)
        {
            foreach(Element element in
set1.Elements)
            {

if(set2.ContainsElement(element))
                {
                    return true;
                }
            }

            return false;
        }

        public bool HasDifferenceBy(Set
set1, Set set2)
        {
            foreach (Element element in
set1.Elements)
            {
```

```
                    if
(!set2.ContainsElement(element))
                    {
                        return true;
                    }
                }

                return false;
            }

            public bool IsSubsetOf(Set set1, Set
set2)
            {
                if(HaveIntersection(set1, set2) &&
HasDifferenceBy(set2, set1) &&
!HasDifferenceBy(set1, set2))
                {
                    return true;
                }

                return false;
            }

            public bool IsSupersetOf(Set set1,
Set set2)
            {
                if (HaveIntersection(set1, set2)
&& HasDifferenceBy(set1, set2) &&
!HasDifferenceBy(set2, set1))
                {
                    return true;
                }

                return false;
            }

            public bool AreEquivalent(Set set1,
Set set2)
            {
                if (HaveIntersection(set1, set2)
&& !HasDifferenceBy(set1, set2) &&
!HasDifferenceBy(set2, set1))
                {
                    return true;
                }

                return false;
            }

            public bool IsEmpty(Set set)
            {
                return set.Capacity == 0;
            }

            public bool IsFull(Set set)
            {
                return set.Capacity ==
Universum.Capacity;
            }

            public List<string> Properties ()
            {
                List<string> properties = new();

                foreach(Set set in Sets)
                {

properties.Add(set.MainProperty);
                }

                return properties;
            }

            public List<Element> Elements()
            {
                return Universum.Elements;
            }

            public Set
SetWithMainProperty(string property)
            {
                foreach(Set set in Sets)
                {
                    if(set.MainProperty ==
property)
                    {
                        return set;
                    }
                }

                return null;
            }

            private bool CheckName(string
name)
            {
                string validValues =
"qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDF
GHJKLZXCVBNM 1234567890";

                foreach(char symb in name)
                {
                    if(!validValues.Contains(symb))
                    {
                        return false;
                    }
                }

                return true;
            }

            private string NameToSet()
            {
                foreach(string name in
Names.Sets)
                {

if(!ContainsSetWithName(name))
                    {
                        return name;
                    }
                }

                return "Error";
```

```
        }

        private bool
ContainsSetWithName(string name)
        {
            foreach(Set set in Sets)
            {
                if(set.Name == name)
                {
                    return true;
                }
            }

            return false;
        }

        private string NameToProperty()
        {
            foreach (string name in
Names.Properties)
            {
                if
(!ContainsPropertyWithName(name))
                {
                    return name;
                }
            }

            return "Error";
        }

        private bool
ContainsPropertyWithName(string name)
        {
            foreach (string nameOfProperty in
Properties())
            {
                if (nameOfProperty == name)
                {
                    return true;
                }
            }

            return false;
        }

        private string NameToElement()
        {
            foreach (string name in
Names.Elements)
            {
                if
(!ContainsElementWithName(name))
                {
                    return name;
                }
            }

            return "Error";
        }

        private bool
ContainsElementWithName(string name)
        {
            foreach (Element element in
Elements())
            {
                if (element.Name == name)
                {
                    return true;
                }
            }

            return false;
        }
    }

public class Set
    {
        public string Name { get; private set;
}
        public string MainProperty { get;
private set; }
        public List<Element> Elements {
get; private set; }
        public int Capacity { get =>
Elements.Count; }

        public Set(string name, string
mainProperty)
        {
            Name = name;
            MainProperty = mainProperty;
            Elements = new();
        }

        public void AddElement(Element
element)
        {
            if (!Elements.Contains(element))
            {
                Elements.Add(element);

element.AddProperty(MainProperty);
            }
        }

        public void
AddElements(List<Element> elements)
        {
            foreach (Element element in
elements)
            {
                if
(!Elements.Contains(element))
                {
                    Elements.Add(element);

element.AddProperty(MainProperty);
                }
            }
        }
```

```
        public void
RemoveElement(Element element)
        {
            Elements.Remove(element);

element.RemoveProperty(MainProperty);
        }

        public void
RemoveElements(List<Element> elements)
        {
            foreach(Element element in
elements)
            {
                RemoveElement(element);
            }
        }

        public void Clear()
        {
            foreach(Element element in
Elements)
            {
                Elements.Remove(element);

element.RemoveProperty(MainProperty);
            }
        }

        public bool
ContainsElement(Element element)
        {
            return
Elements.Contains(element);
        }

        public bool
ContainsElementWithProperty(string property)
        {
            foreach(Element element in
Elements)
            {

if(element.ContainsProperty(property))
                {
                    return true;
                }
            }

            return false;
        }

        public List<Element>
ElementsWithProperty(string property)
        {
            List<Element> elements = new();

            foreach(Element element in
Elements)
            {

if(element.ContainsProperty(property))
                {
                    elements.Add(element);
                }
            }

            return elements;
        }

        public List<Element>
ElementsWithoutProperty(string property)
        {
            List<Element> elements = new();

            foreach (Element element in
Elements)
            {
                if
(!element.ContainsProperty(property))
                {
                    elements.Add(element);
                }
            }

            return elements;
        }

        public List<string>
SecondaryProperties()
        {
            List<string> properties = new();

            foreach(Element element in
Elements)
            {
                foreach(string property in
element.Properties)
                {
                    if(property != MainProperty
&& !properties.Contains(property))
                    {
                        properties.Add(property);
                    }
                }
            }

            return properties;
        }
    }

    public class Element
    {
        public string Name { get; private set;
}
        public List<string> Properties { get;
private set; }

        public Element(string name, string
propertyOfUniversum)
        {
            Name = name;
            Properties = new();
```

```
Properties.Add(propertyOfUniversum);
                }

                public     void     AddProperty(string
property)
                {
                   if(!Properties.Contains(property))
                   {
                      Properties.Add(property);
                   }
                }

                public   void   RemoveProperty(string
property)
                {
                   Properties.Remove(property);
                }

                public void Clear()
                {
                   for(int i = 1; i < Properties.Count;
i++)
                   {

Properties.Remove(Properties[i]);
                   }
                }

                public   bool   ContainsProperty(string
property)
                {
                   return
Properties.Contains(property);
                }

             }
```

Додаток В
Частина вихідного коду бібліотеки DSGenerators

```
public static class SetsModelGenerator
{
    public static SystemOfSets Generate(TemplateToSetsModelGenerator template)
    {
        SystemOfSets model = new();

        for (int i = 0; i < Rand.Generate(template.SetsNumber); i++)
        {
            model.AddSet();
        }

        for(int i = 0; i < Rand.Generate(template.ElementsNumber); i++)
        {
            model.AddElement();
        }

        foreach(Set set in model.Sets)
        {

set.AddElements(Elements(template.ElementsInSetNumber, model.Elements()));
        }

        return model;
    }

    private static List<Element> Elements((int min, int max) interval, List<Element> elements)
    {
        List<Element> buff = new();
        List<Element> results = new();
        int randomValue;

        foreach(Element element in elements)
        {
            if(!buff.Contains(element))
            {
                buff.Add(element);
            }
        }

        for(int i = 0; i < interval.min; i++)
        {
            randomValue = Rand.Generate(0, buff.Count);


results.Add(buff[randomValue]);


buff.Remove(buff[randomValue]);
        }

        if(interval.min < interval.max)
        {
            int number = Rand.Generate(0, interval.max - interval.min);

            for (int i = 0; i < number; i++)
            {
                randomValue = Rand.Generate(0, buff.Count);


results.Add(buff[randomValue]);


buff.Remove(buff[randomValue]);
            }
        }

        return results;
    }
}

public class TemplateToSetsModelGenerator
    {
        public string Name { get; private set; }
        public (int min, int max) SetsNumber { get; private set; }
        public (int min, int max) ElementsNumber { get; private set; }
        public (int min, int max) ElementsInSetNumber { get; private set; }

        public TemplateToSetsModelGenerator(string name)
        {
            Name = name;
            SetsNumber = (3, 3);
            ElementsNumber = (10, 10);
            ElementsInSetNumber = (0, 10);
        }

        public void Rename(string name)
        {
            if(CheckName(name))
            {
                Name = name;
            }
        }

        public void ChangeSetsNumber(int min, int max)
        {
            if(min == 0)
            {
                min = 1;
            }

            if(max == 0)
            {
                max = 1;
```

```
            }

            if(min > max)
            {
                int b = min;
                min = max;
                max = b;
            }

            SetsNumber = (min, max);
        }

        public void
ChangeElementsNumber(int min, int max)
        {
            if (min > max)
            {
                int b = min;
                min = max;
                max = b;
            }

            if(ElementsInSetNumber.min >
min)
            {
                ElementsInSetNumber = (min,
ElementsInSetNumber.max);
            }

            if (ElementsInSetNumber.min >
max)
            {
                ElementsInSetNumber = (max,
ElementsInSetNumber.max);
            }

            if (ElementsInSetNumber.max >
max)
            {
                ElementsInSetNumber =
(ElementsInSetNumber.min, max);
            }

            SetsNumber = (min, max);
        }

        public void
ChangeElementsInSetNumber(int min, int max)
        {
            if (min > max)
            {
                int b = min;
                min = max;
                max = b;
            }

            if(min > ElementsNumber.min)
            {
                ElementsNumber = (min,
ElementsNumber.max);
            }

            if(min > ElementsNumber.max)
            {
                ElementsNumber =
(ElementsNumber.min, min);
            }

            if(max > ElementsNumber.max)
            {
                ElementsNumber =
(ElementsNumber.min, max);
            }

            SetsNumber = (min, max);
        }

        private bool CheckName(string
name)
        {
            string validValues =
"qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDF
GHJKLZXCVBNM 1234567890";

            foreach (char symb in name)
            {
                if
(!validValues.Contains(symb))
                {
                    return false;
                }
            }

            return true;
        }
    }
```