МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП'ЮТЕРНОЇ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ
КАФЕДРА ПРИКЛАДНОЇ ІНФОРМАТИКИ

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач кафедри

_____ __Гамаюн В.П.
(підпис) (ПІБ)

"_____"_____2021р.

ДИПЛОМНИЙ ПРОЕКТ
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ "БАКАЛАВР"

Тема: _____Конструктор тестів_____

Виконавець: _____ Малацківський Андрій Андрійович
(підпис) **(ПІБ)**

Керівник: _____ Ходаков Данііл Вікторович
(підпис) **(ПІБ)**

Нормоконтролер: _____ Боровик Володимир Миколайович
(підпис) **(ПІБ)**

Київ 2021

ВСТУП

На сьогоднішній день існує безліч систем автоматизованого тестування людей. Відійшовши від паперових методів перевірки якості знань, постало питання правильного проектування та реалізації функціоналу та дизайну, оскільки система потребує максимальної інтуїтивності і водночас серйозної та правильно продуманої системи контролю від шахрайства.

В житті кожної людини настає момент, коли доводиться підтверджувати свій рівень кваліфікації у тому чи іншому напрямі. Екзамени у школі, вступні тести у ВУЗ, отримання ліцензії водія, підвищення кваліфікації – ці всі процедури потребують перевірки. Прогрес не стоїть на місці. Постійно виникає потреба у досконалості, з’являються нові технології, які дозволяють виконувати роботу у певній області швидше. Тому, у наш час, неможливо обійтися без системи контролю та перевірки майже усюди.

З іншого боку, людям властиво помилятись. Неодноразово виникають ситуації, у яких від людського фактору невірно будується і сама архітектура, яка систематично допускає помилку, як наслідок недостатньої кваліфікації розробника. Даного роду системи можуть бути універсальними, але кожен окремий модуль повинен чітко виконувати закладений у нього функціонал. Актуальність та потреба такої системи постійно збільшується, як у плані автоматизації та економії часу, так і у точності та інформативності отриманих результатів.

Постає необхідність повсякденного користування даної системи, а також, у першу чергу у правильності побудови архітектури та інформативності додатку. В рамках дипломного проекту поставлено завдання розробити конструктор тестів.

РОЗДІЛ 1
ОГЛЯД ТЕХНОЛОГІЙ РЕАЛІЗАЦІЇ СИСТЕМИ ТЕСТУВАННЯ

1.1. Система комп’ютерного тестування та її особливості

Фахівців у галузі інформаційних технологій цікавлять питання комп’ютерного контролю знань. За останні декілька десятків років було вивчено різні види контролю [3]; визначено більше десяти типів питань, їх компонентів і мета даних, використовуваних, як правило, при формуванні набору контрольних завдань [4]; розроблені математичні методи оцінки знань [5–7] і різні методи проведення контролю. На сьогодні існує низка цікавих розробок, які присвячено різним аспектам контролю знань і заснованих на сучасних досягненнях науки і комп’ютерної техніки. Серед них можна відзначити [8] формування набору завдань для контролю знань, який здійснюють, зазвичай, випадковим чином, іноді враховуючи параметри завдань, і лише в окремих випадках використовують адаптивну видачу контрольних завдань на базі моделі студента.

Для контролю знань використовуються як традиційні, так і сучасні методи контролю. Найширше розповсюдження знаходять методи контролю знань шляхом тестування. Зокрема, впровадження модульно-рейтингової системи в навчальний процес вищих закладів освіти вимагає застосування тестового контролю для оцінки знань студентів, що забезпечує високу технологічність проведення контролю та об’єктивність його результатів. Для підвищення ефективності організації тестового контролю його доцільно проводити з використанням комп’ютерних тестових програм, що дозволяє автоматизувати процес проведення контролю та обробку результатів тестування.

Аналіз сучасних методів тестування та практичних особливостей проведення тестового контролю, а також загальних вимог до комп'ютерних навчальних систем та практичного досвіду їх використання дозволяє визначити загальні вимоги, яким повинні задовольняти комп'ютерні тестові програми. Дотримання цих вимог визначає основні та додаткові можливості тестових програм.

Електронна система для проведення тестування базується на тих самих принципах, що і паперове або усне тестування.

Основними задачами електронної системи тестування є:

- проводити тестовий контроль на базі тестових завдань закритого та відкритого типів, тобто з наданням (відображенням для вибору) та без надання варіантних відповідей на питання тестових завдань, оскільки тестові завдання інших типів (на відповідність, на встановлення правильної послідовності) можуть бути перетворені в одне або декілька тестових завдань закритого та відкритого типів;
- формувати зовнішні бібліотеки тестових завдань, тобто бібліотеки, котрі не входять до складу самої тестової програми, що дозволить користувачам тестових програм розробляти і використовувати власні тести;
- використовувати декілька бібліотек тестових завдань при проведенні одного тесту з послідовним виконанням тестових завдань з кожної бібліотеки, що дозволить розмістити в окремих бібліотеках тестові завдання з різних тем;
- використовувати бібліотеки тестових завдань в незашифрованому та зашифрованому вигляді, що дозволить або не дозволить ознайомитись зі змістом та виконати тест без введення пароля стороннім особам;
- оцінювати правильну відповідь або будь-яку відповідь на кожне тестове завдання певною кількістю балів та підраховувати загальну кількість набраних балів і відповідну бальну оцінку за заданою шкалою оцінок;
- відводити або не відводити певний час на виконання всього тесту або кожного тестового завдання окремо, що дозволить виконувати тест в різних режимах обмеження часу;

- вибирати тестові завдання з бібліотек тестових завдань в заданій кількості та відображати варіантні відповіді в тестових завданнях закритого типу в випадковому порядку.

1.2. Характеристика методу реалізації

Для реалізації додатків з графічною оболонкою користуються технологією Windows Presentation Foundation. Створення готового програмного продукту, розробник розділяє на три етапи: створення графічного відображення додатку, написання логіки програми(тобто реалізація додатку на мові високого рівня C#) та спосіб зберігання та обробки даних про користувача системи – технологія ADO.NET(англ. ActiveX Data Objects). Написання усього додатку реалізується у середовищі розробки (надалі IDE) Visual Studio.

В основі WPF лежить векторна система візуалізації, яка не залежить від роздільної здатності пристрою виведення і створена з урахуванням можливостей сучасного графічного обладнання. WPF надає засоби для створення візуального інтерфейсу,елементи управління, прив'язку даних, макети, двомірну і тривимірну графіку, анімацію, стилі, шаблони, документи, текст, мультимедіа і оформлення.

Графічна технологія, що лежить в основі WPF, є DirectX. Продуктивність WPF вище, ніж Windows Forms за рахунок використання апаратного прискорення графіки через DirectX.

ADO (об'єкти даних ActiveX) це сукупність програмних компонентів, що надають програмний інтерфейс для доступу до джерел даних із клієнтських додатків. ADO діє як шар для доступу до будь-якого сховища даних узагальнено з коду програми. Це виключає необхідність володіння знаннями про реалізацію бази даних і зменшує складність роботи з кодом низького рівня, необхідним для обробки даних.

ADO використовує постачальник OLEDB для підключення до джерел даних та доступу до даних. OLEDB – це програмний інтерфейс на основі компонентів, який надається для взаємодії з різними джерелами даних. Ці джерела даних можуть бути як реляційними, так і нереляційними базами даних,

такими як об'єктні бази даних, веб-сторінки, електронні таблиці або повідомлення електронної пошти. До OLEDB та ADO, популярною моделлю, що застосовується в додатках на різних платформах, була ODBC (підключення до відкритої бази даних).

Об'єктна модель ADO містить чотири колекції з дванадцяти об'єктів. У різних колекціях – поля, властивості, параметри та помилки. Кожна колекція складається з наступних дванадцяти предметів.

1. З'єднання - для підключення до джерела даних через OLE DB
2. Команда - для відправлення інструкції (SQL-запит або збережена процедура) постачальнику даних
3. Набір записів - група записів, що представляють дані
4. Негайне - набір записів, заблокований оптимістично чи песимістично
5. Пакетна - для здійснення або здійснення транзакцій з відкатної бази даних
6. Транзакція - транзакція з базою даних
7. Запис - набір полів
8. Потік - для читання та запису потоку байтів
9. Параметр - для зміни функціональності
10. Поле - стовпець у базі даних
11. Властивість - можливість провайдера OLEDB
12. Помилка - помилка, з якою стикається постачальник OLEDB під час її виконання

1.3. Практичне значення комп’ютерного тестування

Вхідне та поточне комп’ютерні тестування можуть відбуватися в межах планових навчальних занять (навчальних занять за розкладом), проводяться і контролюються викладачами, які проводять навчальні заняття.

Контрольні комп’ютерні тестування відбуваються поза межами планових навчальних занять за встановленим розкладом контрольного тестування.

Комп’ютерне тестування здійснюють у формі самостійного діалогу

студента з комп’ютером у присутності відповідальної за організацію тестування особи або без неї, з можливістю збереження результатів тестування.

Результати комп’ютерного тестування використовуються для контролю і корегування навчального процесу та розробки заходів щодо підвищення його якості.

Результати контрольних комп’ютерних тестувань оформлюють у вигляді протоколів тестування і використовують при виставленні залікових або екзаменаційних оцінок з відповідних дисциплін.

Електронна система тестування дозволяє використовувати у тестах завдання відкритої (студенти друкують відповідь на тестове завдання) або закритої форм (студенти обирають варіант відповіді на тестове завдання) таких типів:

– правильно – неправильно (закрита форма);

– вибір з множини (закрита форма);

– вибір відповідності (закрита форма);

– встановлення послідовності (закрита форма);

– коротка текстова відповідь (відкрита форма);

– коротка числова відповідь (відкрита форма);

– розрахунок за формулою (відкрита форма);

Тестові завдання повинні відповідати певним вимогам:

а) текст тестового завдання – це коротке судження або систему суджень;

б) тестове завдання повинно мати такі риси:

– однозначність – студент не повинен замислюватися, у якому сенсі трактується означене судження;

– ясність – студент повинен розуміти, що від нього хочуть;

– стислість – судження не повинне займати багато місця на екрані комп’ютера, розмір оптимального судження до 10 слів;

– істинність судження – дія студента повинна перетворювати тестове завдання у істинне судження;

в) визначальна ознака повинна бути необхідною і достатньою;

г) вимоги до використання термінів у тестовому завданні:

– студент повинен знати терміни, які використано у тестовому завданні;

– не повинні використовуватися терміни, які мають різні тлумачення у різних джерелах;

д) загальні вимоги:

– мінімум слів (що менше слів, то швидше студент «схопить» суть тестового завдання);

– відсутність частки «не» (частка «не» погано сприймається у тестовому завданні);

– від 4 до 6 варіантів відповідей (для забезпечення складності відгадування);

– правильна відповідь не повинна відрізнятися від інших варіантів відповідей;

– тестове завдання не повинно бути у взаємозалежності від інших тестових завдань.

Практичне значення впровадження комп'ютерного тестування є перспективним напрямом сучасного освітнього процесу. Водночас зазначимо,що комп'ютерне тестування не може (і не повинно) перебирати на себе всі контролюючі функції щодо навчальних досягнень учнів, натомість повинно стати однією зі складових діагностики знань.

Запорукою широкого впровадження такого виду контролю має бути наукове обґрунтування, потужна психолого-педагогічна і матеріально-технічна база. При впровадженні комп'ютерного тестування слід враховувати не лише переваги, але й ризики, які його супроводжують. Серед останніх слід відзначити такі:

- відсутність безпосереднього контакту з учнем під час тестування підвищує ймовірність впливу випадкових факторів на результат оцінювання;
- комп'ютерне тестування з низки навчальних предметів не дасть картину глибинного розуміння предмету.

Узагальнення досвіду проведення комп'ютерного тестування дозволяє зробити висновки, що його впровадження сприяє:

– систематичному відстеженню якості та динаміки навчальних досягнень

студентів;

– отриманню статистично достовірної картини індивідуального прогресу кожного студента;

– створенню регіонального комп’ютерного банку даних навчальних досягнень студентів із предметів за тривалий час навчання;

– інтенсифікації навчального процесу завдяки збільшенню обсягу навчального матеріалу в аудиторії;

– підвищенню зацікавленості студентів навчально-виховним процесом;

– можливості творчого і практичного застосування знань, умінь і навичок;

– можливості виконувати завдання не лише під контролем викладача, а й здійснювати самоконтроль навчальної діяльності.

Вибір конкретного методу тестування та типу тестових завдань залежить від цільової мети тестового контролю і попередньо обраних показників оцінки рівня знань. Класифікація тестових завдань залежно від їх характерних ознак представлено на рис. 1.1.

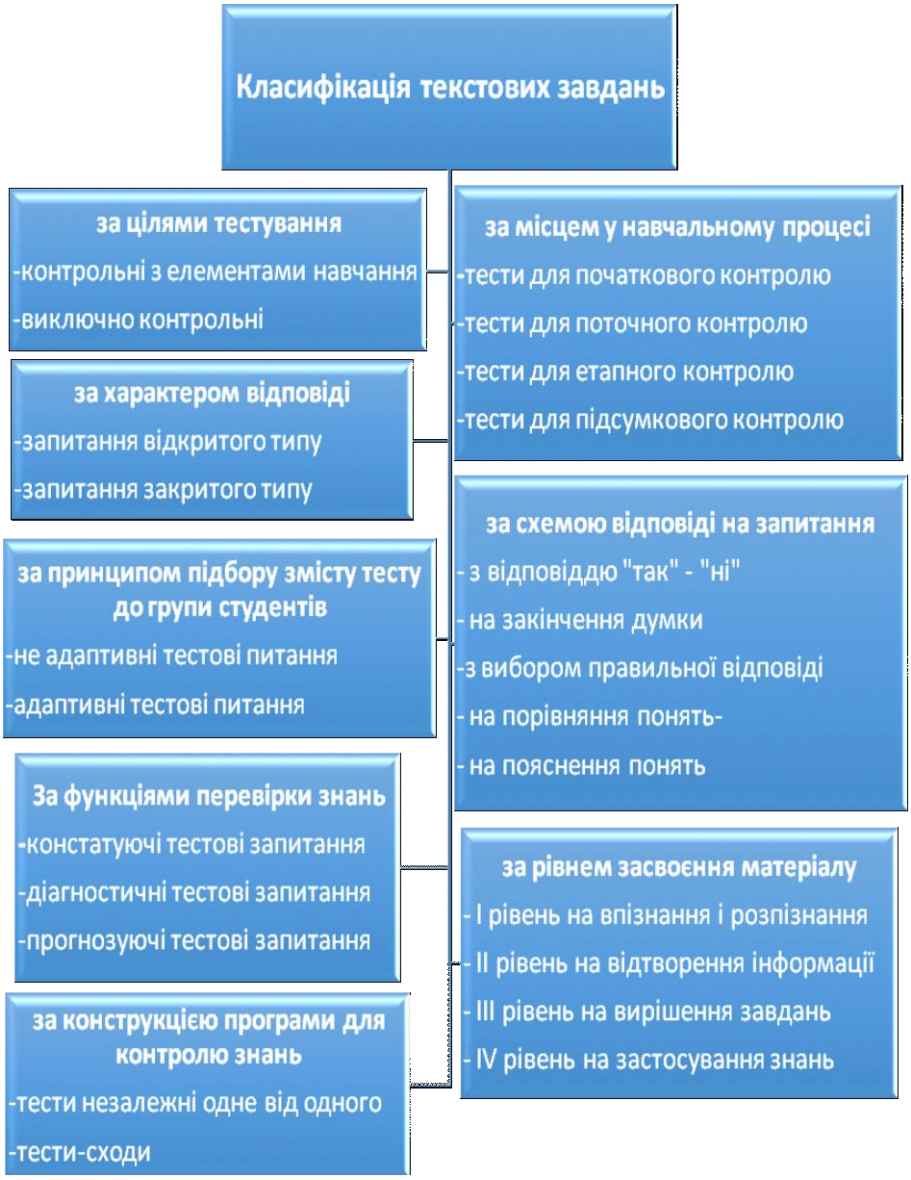


Рис. 1.1. Класифікація тестових завдань

При розробці тестових завдань слід враховувати їх різноманітність і орієнтуватися на таксономічні цілі:

– знання (фактів, ключових ідей, дат і так далі),

– розуміння (причин, наслідків),

– вживання (знань для вирішення нових завдань, проблем),

– аналіз (частин, компонентів, прихованих значень),

– синтез (нових ідей, виводів і висновків),

– оцінка (ідей, цінності теорій, уявлень).

Час на здачу тесту в загальному випадку складається з декількох величин:

$$T = To + Tур + Tз,$$

де To (оперативний час) – час, необхідний для фізичного виконання всіх операцій протягом тестування: вхід в систему, введення відповідей, перехід між питаннями, завершення тестування;

Tур (час на ухвалення рішення) – час, необхідний студенту, щоб обдумати відповідь на питання;

Tз (запас часу) – час, який потрібно надати додатково, на випадок виникнення непередбачених ситуацій, наприклад, якщо віддалені користувачі звертаються до серверу через канали з низькою пропускною здатністю.

Час на тестування з метою самоперевірки можна не обмежувати.

Висновки до розділу 1

В даному розділі розглянуто основні проблеми процесу тестування її часткове або повне заміщення електронними системами, технології для розробки даної системи, її цінність та важливість у користуванні. Також розглянули один із підходів до проектування системи по шаблону. Виділили основні етапи розробки додатку. Проаналізували область застосування та цільову аудиторію користувачів.

РОЗДІЛ 2

ПРОЕКТУВАННЯ ТА ПРОТОТИПУВАННЯ конструктора тестів

2.1. Технічне завдання системи конструктор тестів

Технічне завдання - це документ, у якому детально описані процедури, методи та графічні інтерфейси додатку, згідно яких виконується розробка будь-якого додатку [8].

Замовник не може знати та уявляти повний функціонал додатку. Розробник намагається відтворити функціонал та візуальний вигляд максимально наближеним до замовлення, але це не завжди вдається. Розробка додатку, як і будь-якого іншого продукту, починається на основі початкових даних, у які по ходу розробки, будуть вноситись коригування, що впливає на термін роботи. Зазвичай, це призводить до часткового або повного редагування деяких модулів програми. Цей процес потребує додаткових доплат та змінюється кінцева вартість продукту. Також, не кожен раз, при редагуванні додатку на деяких етапах, доводиться змінювати архітектури додатку, що також впливає на неузгодженість з замовником. Не зважаючи на те, що графічна частина проекту завжди буде розроблена з можливістю подальшого редагування без суттєвого впливу на роботу усієї системи, редагувати структуру проекту, у якій описана логіка взаємодії компонентів – задача не з легких [3].

Маючи під рукою правильно побудоване технічне завдання, розробники мають змогу з максимальною точністю вирахувати вартість та терміни розробки додатку, розпланувати етапи розробки, врахувати можливість доповнення та модифікації в майбутньому, скласти шаблон для графічного інтерфейсу, розробити та врахувати стратегію майбутнього просування.

Для замовника – це можливість повністю уявити масштаби майбутнього проекту, допрацювати усі невраховані складові, почати готувати наповнення та інші дані, які необхідні для запуску додатку. В свою чергу, технічне завдання дає розробникам точні інструкції, можливість розпланувати графік роботи, а замовнику – дату релізу проекту. Як правило, проект, розроблений на основі технічного завдання, виходить якісний та надійний.

Процес розробки системи тестування можна розділити на наступні етапи:

1) аналіз завдань, які повинен вирішувати додаток, визначення цільової аудиторії.
2) написання технічного завдання.
3) моделювання - планування структури майбутнього додатку.
4) розробка графічної моделі додатку.
5) кодування основного функціоналу сайту.
6) інтеграція графічної моделі із основним кодом додатку.
7) створення Бази Даних для збереження необхідної інформації.
8) підключення бази даних до основної частини додатку.
9) тестування та відладка додатку на відмовостійкість.

2.2. Особливості та відмінності MVVM шаблону

MVVM (Model-View-ViewModel) – це шаблон, який з'явився для обходу обмежень патернів проектування MVC і MVP, і об'єднує деякі з їх сильних сторін [16]. Ця модель вперше з'явилася в складі фреймворка Small Talk в 80-х, і була пізніше покращена з урахуванням оновленої моделі презентацій (MVP).

Шаблон MVVM має три основних компоненти: модель, яка представляє бізнес-логіку додатка, представлення призначеного для користувача інтерфейсу XAML, і представлення-модель, в якому міститься вся логіка побудови графічного інтерфейсу і посилання на модель, тому він виступає в якості моделі для подання. На наступному малюнку представлена діаграма, яка показує, як реалізувати шаблон MVVM. Звичайно, це загальна реалізація (Рис.2.1).

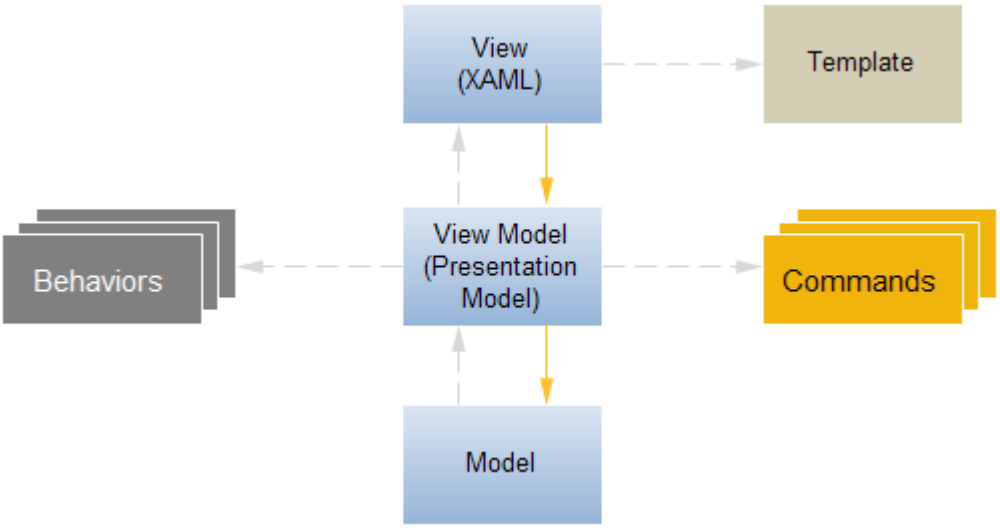


Рис. 2.1. MVVM шаблон

Так як розробка починається на WPF, потрібно скористатися механізмом прив'язок (binding), що надаються цими технологіями. Для цього, модель-представлення повинна реалізувати деякі конкретні інтерфейси, необхідні модулю прив'язки з WPF. Одним з них є INotifyPropertyChanged, введений в .NET Framework починаючи з версії 2 і вище. Цей інтерфейс реалізує систему повідомлень яка активується, коли значення властивостей змінюється. Це потрібно в моделі-представлення, щоб зробити механізм прив'язки призначеного для користувача інтерфейсу XAML динамічним.

Іншим налаштуванням є команди, які надаються інтерфейсом ICommand, які доступні для WPF. Команди можуть прив'язуватися до певного XAML-елементу і визначати поведінку даного елемента при певних діях.

Третім компонентом в показаної вище структурі є шаблон даних DataTemplate, який визначає як структурувати конкретну модель-представлення або особливий стан моделі-представлення.

2.3. Проектування системи комп’ютерного тестування

Розроблюваний додаток складається з наступних основних компонентів:

- додаток, автономний або онлайн з вільним доступом для цільової аудиторії;
- панелі для керування(адміністрування) інформацією та даними.

Інтерфейс додатку розробляється повністю українською мовою. Графічні матеріали надаються у вигляді електронних файлів у форматі JPEG - для растрових зображень і в форматі PDF,SWF,SVG - для векторних зображень. Текстові матеріали надаються у вигляді електронних документів в форматах Microsoft Word або Rich Text Format.

Додаток дає змогу користувачам пройти тестування з певної області у зручний час, та має інформативний інтерфейс.

Розробник має повний доступ до системи на момент розробки додатку, залишає за собою права на адміністрування та підтримки додатку у належному стані.

Розроблено схему взаємозв’язків викладача та студента з використанням автоматизованої системи тестування (рис.2.2).

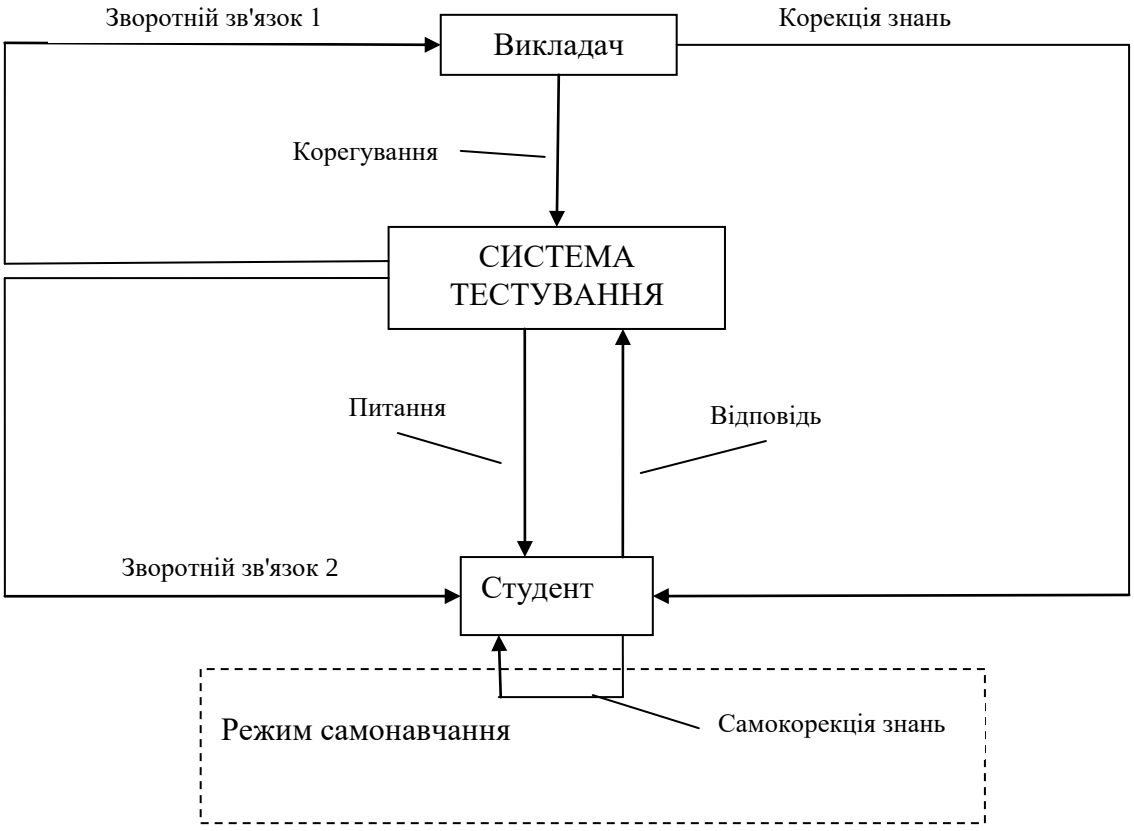


Рис. 2.2. Схема взаємодії в тріаді «Викладач – Система тестування – Студент»

У тріаді «Викладач – Система тестування – Студент» можна виокремити два типи зворотних зв’язків:

– зовнішній;

– внутрішній.

Внутрішній зворотній зв’язок – це інформація, яка надходить від системи тестування до студента у відповідь на його дії при виконанні тестових завдань. Цю інформацію призначено для корекції студентом своїх знань, спонукає студента до рефлексії, є стимулом до подальших дій, допомагає оцінити і скорегувати результати навчання.

При проектуванні комп’ютерних систем контролю знань розрізняють консультуючий і результативний внутрішній зворотній зв’язок.

Консультуючий зворотній зв’язок може бути різним: допомога, роз’яснення, підказка, натяк тощо.

Результативний зворотний зв’язок також може бути різним: від «правильно – неправильно» до демонстрації правильного результату або способу дії.

За допомогою зовнішнього зворотного зв’язку інформація в даній тріаді надходить до викладача і використовується ним для корекції діяльності студента і навчальної програми.

2.4. Специфікація вимог до системи тестування

Специфікація вимог для програмної системи – це повний опис поведінки системи що розробляється. Включає множину прецедентів які описують всі взаємодії, які користувачі мають з програмним забезпеченням. Прецеденти також відомі як функціональні вимоги. На додачу до прецедентів також включає нефункціональні вимоги. Нефункціональні вимоги є вимогами які накладають обмеження на проект, чи реалізацію. Специфікація вимог до системи включає опис варіантів використання.

Наведемо діаграму варіантів використання для акторів системи (рис 2.2.)

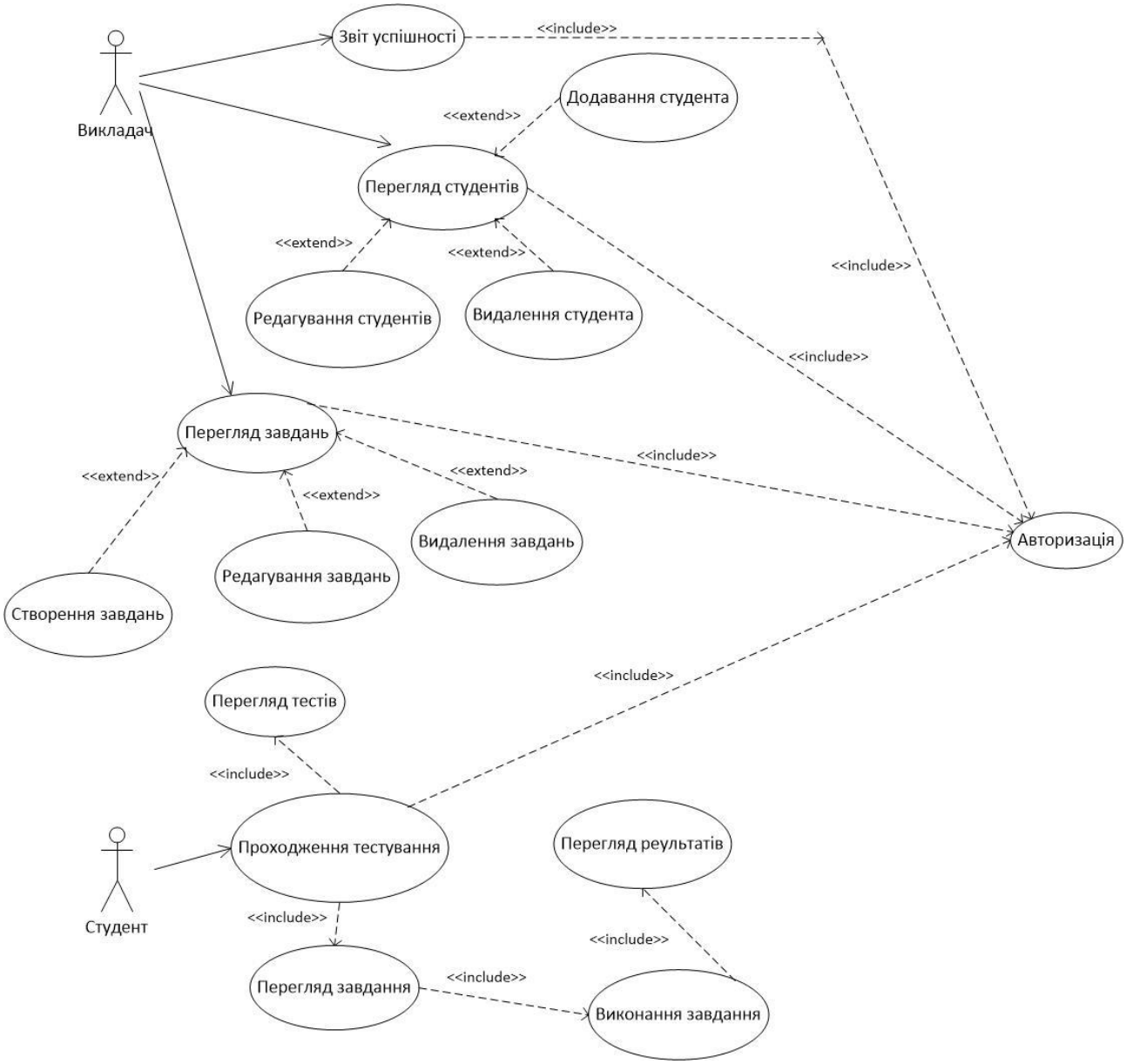


Рис. 2.2. Діаграма варіантів використання системи тестування знань студентів

На рисунку 2.3 представлена діаграма варіантів використання модуля «Студент» системи Конструктор тестів.

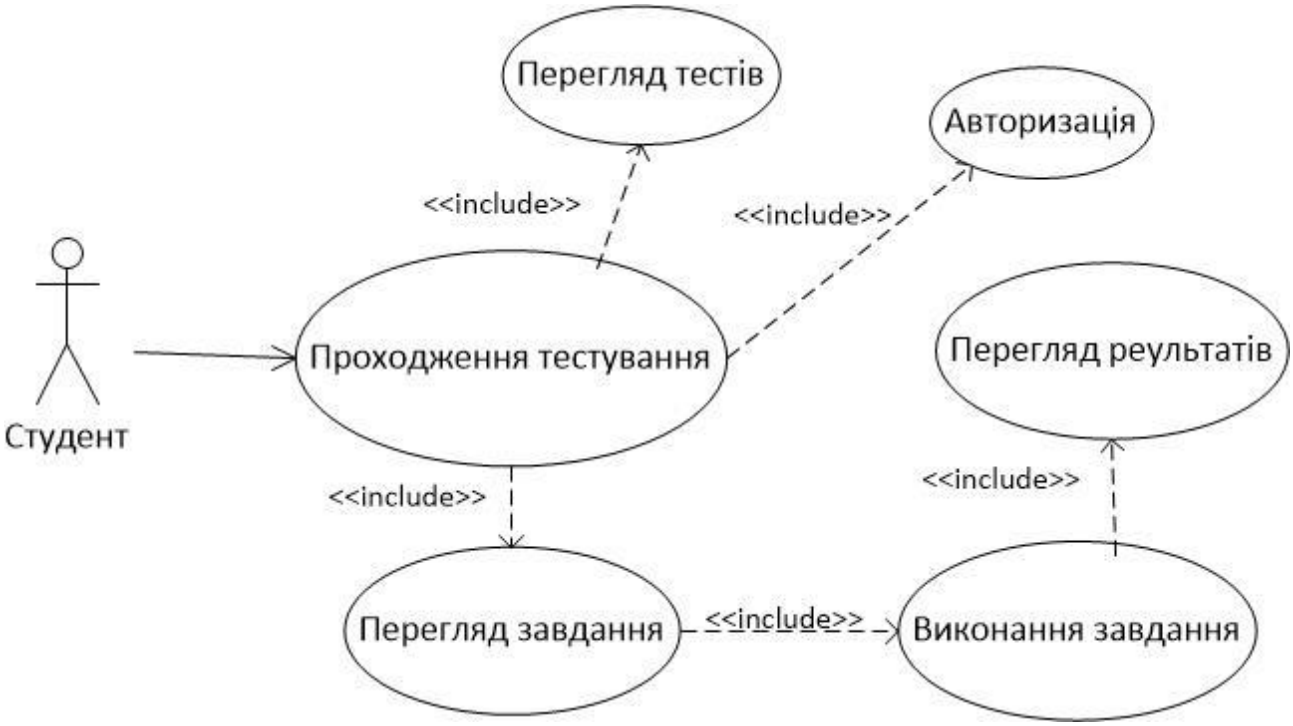


Рис. 2.3. Діаграма варіантів використання модуля «Студент» системи тестування знань студентів

Наступним кроком у визначенні специфікацій вимог до системи буде опис варіантів використання. Варіанти використання представленні у таблицях 3.6-3.11 Варіант використання «Перегляд результату» передбачає перегляд користувачем результату перевірки відповіді на завдання (табл.2.1).

Таблиця 2.1

Варіант використання «Перегляд результату»

Контекст використання	Переглянути результат виконання завдання
Дійові особи	Студент
Передумова	Виконане завдання
Триггер	Завершення виконання завдання
Сценарій	Користувач після завершення виконання завдання отримує результат
Післяумова	-

Варіант використання «Виконання завдання» передбачає введення користувачем відповіді на завдання (табл.2.2).

Таблиця 2.2

Варіант використання «Виконання завдання»

Контекст використання	Введення запиту відповідно до завдання
Дійові особи	Студент
Передумова	Відкрите вікно перегляду завдання
Триггер	Відкриття вікна перегляду завдання
Сценарій	Користувач переглянувши завдання вводить відповідний запит
Післяумова	Перегляд результату

Варіант використання «Авторизація» передбачає ідентифікацію користувача в системі та виявлення доступних для нього завдань (табл.2.3).

Таблиця 2.3

Варіант використання «Авторизація»

Контекст використання	Авторизизація в системі з метою ідентифікації
Дійові особи	Студент, Викладач
Передумова	Запуск додатку
Триггер	Запуск додатку
Сценарій	Користувач вводить дані авторизації
Післяумова	Проходження тестування, Створення тесту

Варіант використання «Перегляд завдання» передбачає перегляд умови завдання з метою його подальшого виконання (табл.2.4).

Таблиця 2.4

Варіант використання «Перегляд завдання»

Контекст використання	Перегляд завдання з метою його виконання
Дійові особи	Студент
Передумова	Користувач вибрав завдання
Триггер	Користувач вибрав завдання
Сценарій	Користувач переглянувши список завдань вибирає бажане завдання та переглядає його
Післяумова	Виконання завдання

Варіант використання «Перегляд загального результату» передбачає перегляд користувачем загальної успішності виконання набору тестових завдань (табл.2.5).

Таблиця 2.5

Варіант використання «Перегляд результатів»

Контекст використання	Перегляд прогресу при проходженні тестування
Дійові особи	Студент
Передумова	Користувач авторизувався в системі
Триггер	Користувач авторизувався в системі
Сценарій	Користувач переглядає прогрес при проходженні тестування
Післяумова	Виконання завдання

Варіант використання «Перегляд тестів» передбачає перегляд користувачем списку завдань та вибору бажаного (табл.2.6).

Таблиця 2.6

Варіант використання «Перегляд списку завдань»

Контекст використання	Перегляд списку тестів з метою вибору бажаного
Дійові особи	Студент
Передумова	Користувач авторизувався в системі
Триггер	Користувач авторизувався в системі
Сценарій	Користувач переглядає список тестів і вибирає бажане
Післяумова	Перегляд завдання

Наступним кроком є створення діаграми класів UML (рис. 2.4).

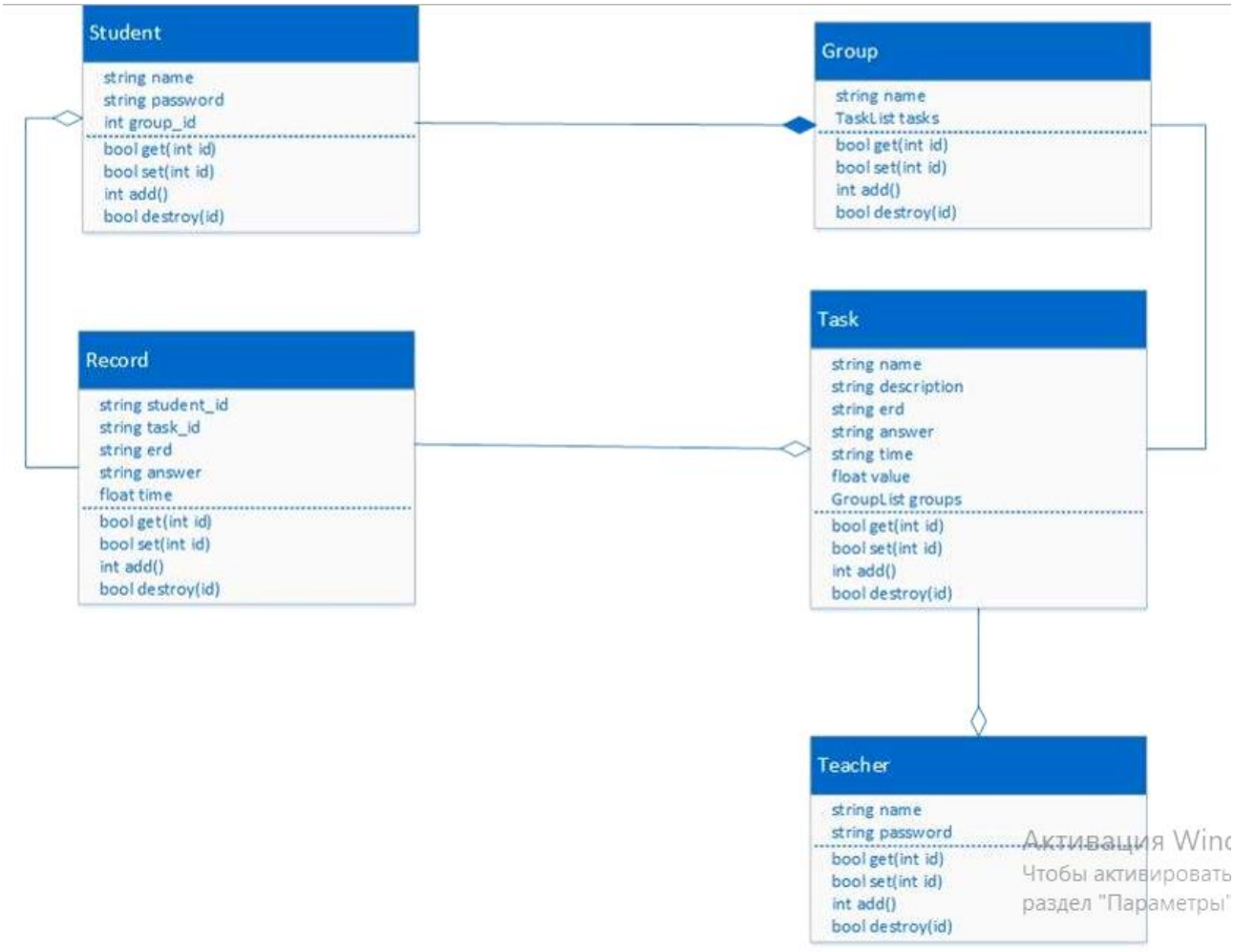


Рис. 2.4. Діаграма класів UML

Найбільше поширеним засобом моделювання даних є діаґрами “сутність-взаємозв'язок” (ERD). З їхньою допомогою визначаються важливі для предметної області об'єкти (сутності), їх властивості (атрибути) і відношення один з одним (зв'язки). ERD безпосередньо використовуються для проектування реляційних баз даних. Тому на підставі аналізу предметної області у роботі спроектовано структуру бази даних у вигляді ERD діаграми (рис. 2.5).

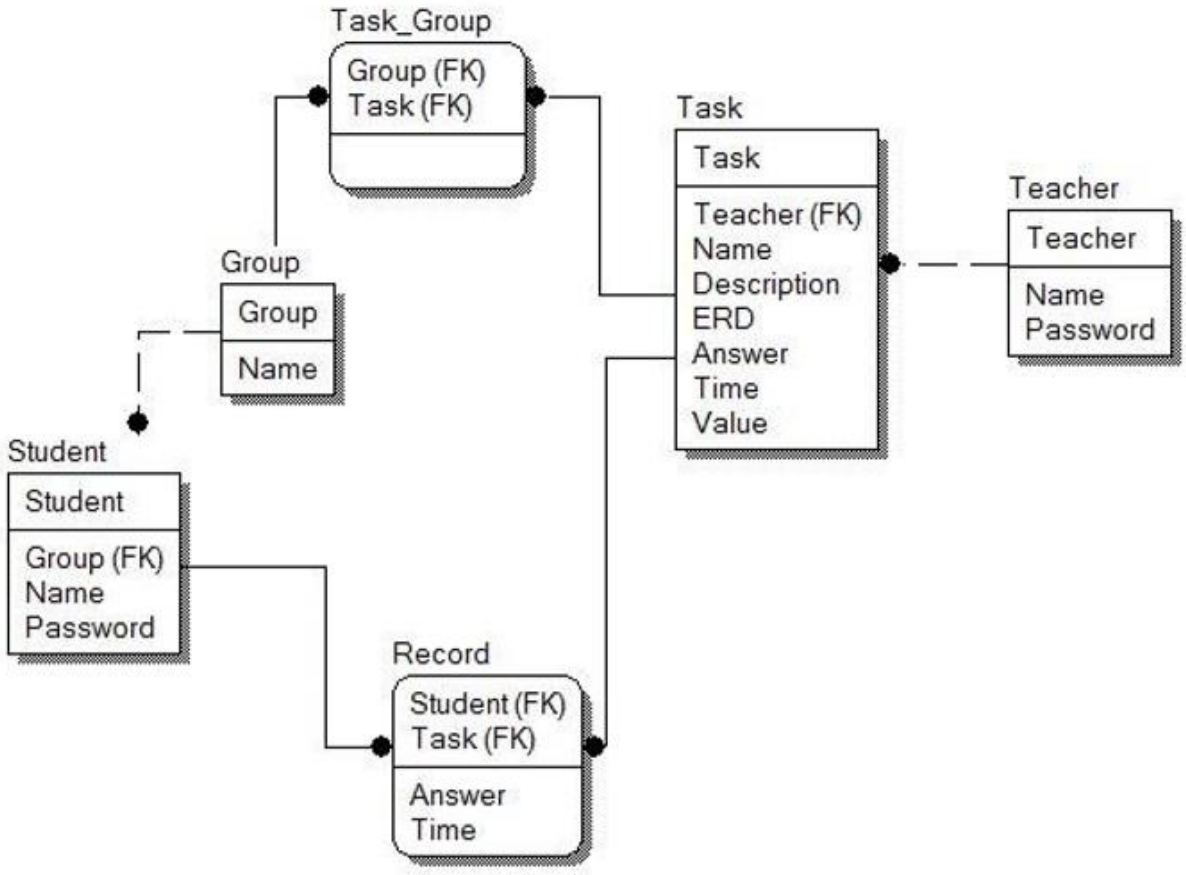


Рис. 2.5. ERD-діаграма

Проектування структури бази даних дозволяє визначити та повністю описати всі відношення та поля і є завершальною ланкою перед етапом програмної реалізації бази даних.

Наступним кроком буде планування індексів. Індекси – це спеціальні таблиці, які використовуються пошуковою системою бази даних для швидшого отримання шуканої інформації. Вони використовуються як вказівники на дані в таблицях. Використання індексів пришвидшують виконання запитів пошуку,

однак запити додавання на редагування даних виконуються повільніше. Всі зовнішні ключі являються індексами:

· group_id – зовнішній ключ таблиць Student та Task_Group · student_id – зовнішній ключ таблиці Record

· task_id – зовнішній ключ таблиць Record і Task_Group · teacher_id – зовнішній ключ таблиці Task Також для оптимізації пошуку було вирішено індексувати наступні поля: value – поле важливості завдання в таблиці Task · name – поле імені студента в таблиці Student.

2.5. Проектування дизайну

При розробці додатку, необхідно використовувати елементи управління, які передають тип операції за їх назвою та відповідним стилем.

Оформлення дизайну має строгий вигляд, звичний для користувачів відповідної ОС, що дає змогу швидше адаптуватися до користування системою. Дизайнерське рішення передбачає відображення усіх вікон, елементів керування та навігації на моніторах та екранах будь-якої роздільної здатності, за допомогою новітніх технологій, які були закладені у систему.

Користувачам з дискретними та інтегрованими графічними адаптерами зображення буде мати однаковий характер без розмитостей та нечітких пікселів.

Вікно «Авторизація». Вікно типового стилю для операційної системи з текстовими полями, надписами та кнопками. Вікно містить в собі достатньо інформації для нового користувача і користування ним навіть у перший раз, не представить проблеми. Інтуїтивно зрозуміло описано вхід до системи та місце для вводу інформації.

Кнопка «Вхід». Слово написано чорним кольором на білому фоні. Кнопка прямокутної форми з закругленими кутами. Розташована у групі з іншими кнопками навігації, такими як: «Реєстрація» та «Забули пароль». Вікно авторизації виконане у вигляді невеликого квадрату із типовою палітрою кольорів для системи. Після натискання кнопки «Вхід» спливає інформаційне

повідомлення про перехід до тестування – вхід від імені користувача, або вхід від імені адміністратора – вхід у панель керування додатком відповідно.

Кнопка «Реєстрація». Даний елемент керування має повністю ідентичну будову як і «Вхід». Кнопка виконана для переходу користувача до вікна «Реєстрація», де він проходить швидкий та простий етап реєстрації облікового запису для подальшого проходження тестів.

Кнопка «Забули пароль». Кнопка відображена у вигляді синього тексту на білому фоні для інтуїтивної зрозумілості дії даного елементу. При наведенні курсору на кнопку, чітко виділяються рамки кнопки та при натисненні на кнопку відбувається перехід на форму «Відновлення паролю», де користувач має змогу відновити пароль по власній електронній пошті, ввівши її в поле та натиснувши кнопку «Відновити».

Вікно «Тестування» виконано у вигляді прямокутника, достатнього розміру без можливості максимізації, адаптоване до перегляду на будь-якому пристрої відображення. У верхній частині відображається елемент керування тексту питання білету, відразу нижче формується група варіантів відповідей до відповідного запитання. У нижній частині вікна знаходиться панель керування тестом, яка складається із групи кнопок «Далі», «Назад» та «Пропустити» та елементів відображення імені користувача, точний час і дата початку тесту, час до закінчення тесту, номер білета і питання.

Перед початком тесту, на екран користувача відображається інформація у вигляді підказок по навігації тестовим вікном. Також у вікні зверху є стрічка навігації, яка дає змогу вивести інформацію на екран по керуванню тестом.

Вікно «Адміністрування». Виконане у тому ж вигляді, що і «Тестування» з таким самим способом відображення. У центрі вікна знаходяться елементи керування для редагування будь-якої інформації стосовно системи в цілому. Текстове поле, чекбокси, випадаючі списки підписані текстовими мітками для максимальної зручності та простоті користування. Деякі з кнопок форми містять у собі переходи на окремі випливаючі модальні вікна для поглибленого адміністрування.

Усі вікна у вигляді прямокутника з характерним візуальним стилем для поточної операційної системи. Розмір вікон є сталим для правильного відображення інформації на різного типу пристроїв виводу. Фон форм білий. Межі форми виділяються стилістичними рамками. Всі оздоблювальні лінії в формі мають також чорний колір. Всі вікна, окрім спеціалізованих модальних, випливають поверх попередніх по ієрархії з можливістю повернутись на попереднє після натискання навігаційної кнопки повернення. Всі текстові мітки, поля для вводу, кнопки відображаються однаковим шрифтом та розміром, для підтримки стилізації.

Вікно «Особистий кабінет». Для зареєстрованих користувачів додатку є можливість отримати доступ у особистий кабінет. Вхід у особистий кабінет стає доступним після реєстрації користувача. У даному вікні відображається інформація про дату та час реєстрації, кількість пройдених білетів та результати по тестам.

Смуга у верхній частині вікон. Смуга виконана по усій ширині вікна, сірого кольору та має однаковий із усіма іншими елементами однаковий шрифт для підтримки стилізації проекту. Смуга у кожному вікні відіграє роль довідника по елементам керування додатку.

2.6. Структура системи комп’ютерного тестування та навігація

Структура розроблювального додатку складається з наступних елементів:

1) Навігаційне меню;

2) Елементи вводу даних;

3) Елементи переходу по вікнам;

4) Основний інформаційний блок (тестування, адміністрування, особистий кабінет, вкладені модальні вікна);

5) Інформаційні панелі знизу вікна(крім вікна авторизації, та відновлення паролю).

Вікно «Авторизація» (рис.2.6).

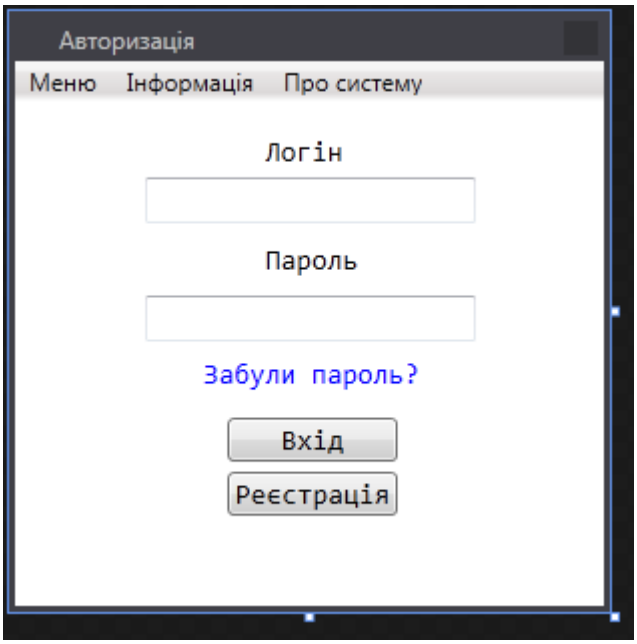


Рис.2.6. Макет вікна №1. Вікно авторизації.

Вікно – основне вікно складається.

- Іконки «Персонажу»
- Найменування вікна.

Навігаційне меню - верхній блок вікна, який складається з наступних елементів.

- Елемент «Меню».
- Елемент «Інформація».
- Елемент «Про систему».

Панель елементів для авторизації.

- Текстова мітка «Логін».
- Текстове поле для вводу логіну.
- Текстова мітка «Пароль».
- Текстове поле для вводу паролю, пароль вводиться у вигляді «***».

Навігаційна панель для переходу по вікнам.

- Кнопка без рамки «Забули пароль?».
- Кнопка «Вхід».
- Кнопка «Реєстрація».

Вікно «Реєстрація» (рис.2.7).

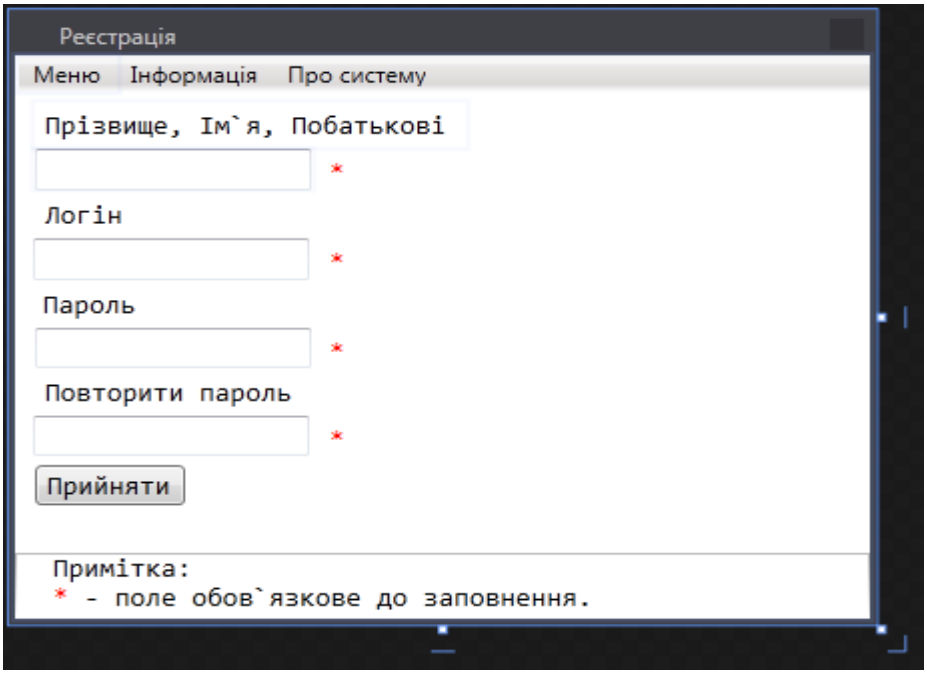


Рис. 2.7. Макет вікна №2. Вікно реєстрації.

Вікно реєстрації нових користувачів складається.

- Іконки «Додавання нового користувача»
- Найменування вікна.

Навігаційне меню - верхній блок вікна, який складається з наступних елементів.

- Елемент «Меню».
- Елемент «Інформація».
- Елемент «Про систему».

Основне наповнення вікна складається із текстових міток, текстових полів введення інформації та кнопки «Прийняти». Також у нижній частині вікна відображена панель, яка містить примітку для заповнення обов’язкових полів. Біля кожного такого поля прикріплена текстова мітка у вигляді «*» червого кольору. Стилістика відображення панелі та шрифту збережена відповідно то оформлення усіх надписів у системі.

Вікно «Відновлення паролю» (рис.2.8).

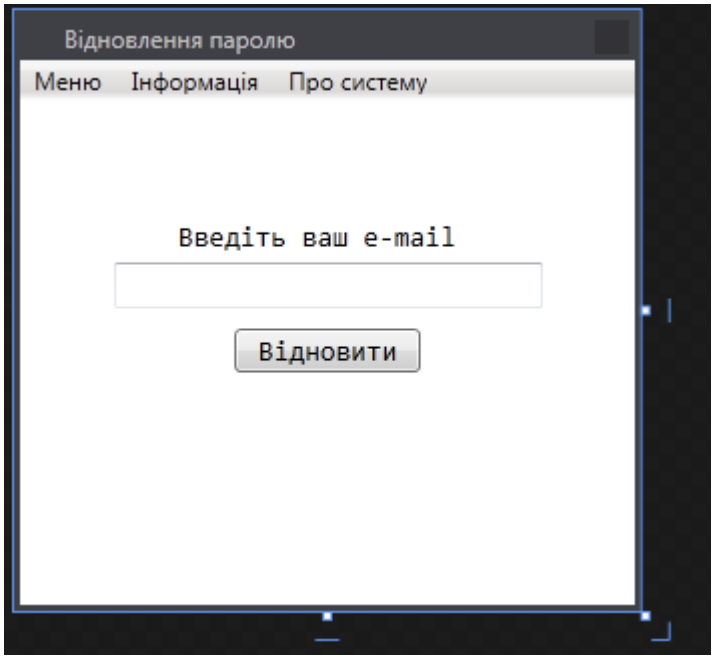


Рис. 2.8. Макет вікна №3. Вікно відновлення паролю.

Як і на усіх немодальних вікнах цієї системи, у верхній частині вікна розташована навігаційне меню. Навігаційне меню - верхній блок вікна, який складається з наступних елементів.

- Елемент «Меню».
- Елемент «Інформація».
- Елемент «Про систему».

Також у вікні, основним наповненням являється текстова мітка «Введіть ваш e-mail», текстове поле для вводу електронної адреси та кнопка «Відновити».

Після натиснення кнопки, відбувається автоматична генерація повідомлення на електронну адресу та сповіщення у діалоговому вікні нового паролю.

Вікно «Адміністрування» (рис.2.9).

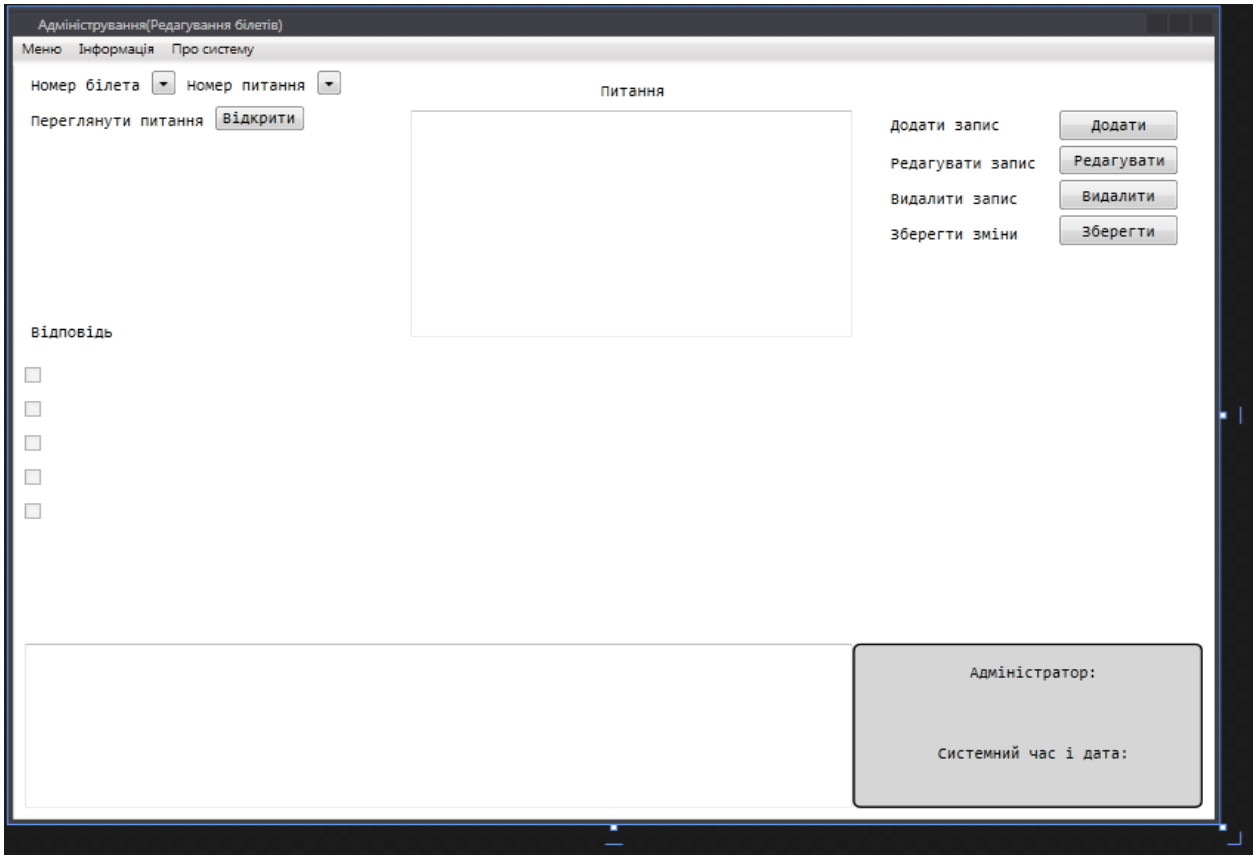


Рис. 2.9. Макет вікна №4. Вікно адміністрування.

Дане немодальне вікно має навігаційне меню, яке ідентичне попереднім та містить у собі:

- Текстові мітки:

• номер білета;

• номер питання;

• переглянути питання;

• питання;

• додати запис;

• редагувати запис;

• видалити запис;

• зберегти зміни;

- елементи випадаючого списку:

• номери білетів;

• номери питань по білету;

- кнопки:

• відкрити питання;

• редагувати запис;

• видалити запис;

• зберегти зміни;

- Текстове поле, виступає у вигляді перегляду обраного питання за допомогою елементів випадаючого списку;
- Текстові мітки, які вміщені у елемент рамку – відображення поточної дати та часу, ім’я адміністратора, який знаходиться у системі;
- Текстове поле для відображення логів системи.
-

Вікно «Тестування» (рис.2.10).

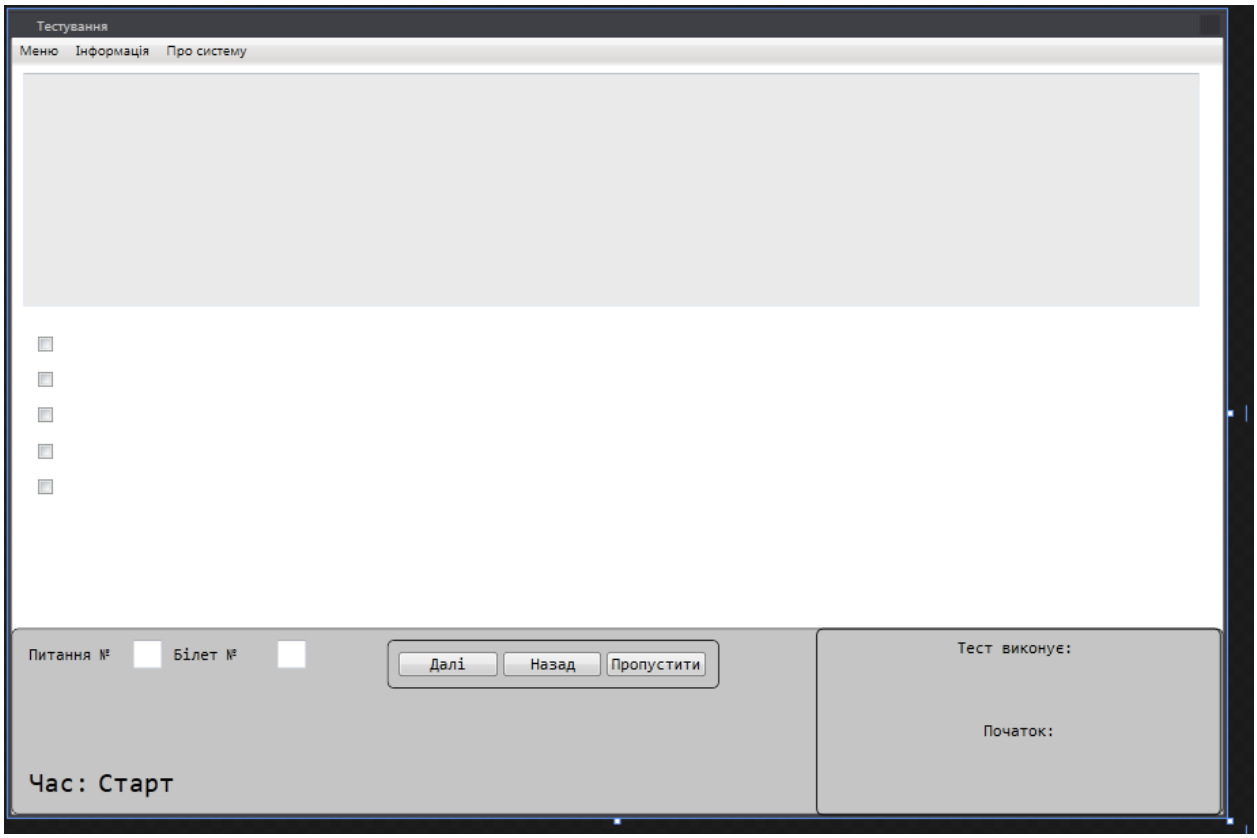


Рис. 2.10. Макет вікна №5. Вікно тестування.

Вікно, призначене для кінцевого користувача, яке містить:

- Текстове поле для відображення питання;
- Можливі варіанти відповіді на питання;
- Текстові мітки, які відображають:
    • Користувача, який виконує тест;
    • Дату та час початку тесту;
    • Залишок часу до кінця проходження тесту;
    • Поточний білет та номер питання;
- Кнопки навігації по екзаменаційному білету:
    • «Далі»;
    • «Назад»;
    • «Пропустити»;

2.7. Створення дизайну системи на основі MVVM шаблону

Шаблон Model-View-ViewModel (MVVM) є загальним для багатьох додатків XAML. Рекомендації на користь MVVM наступні.

- Поділ проблем. Завжди корисно розділити проблему на більш дрібні елементи, і шаблон типу MVVM або MVC дозволяє розділити додаток (або навіть один елемент управління) на більш дрібні елементи: представлення (view), логічну модель представлення (view-model) і незалежну від представлення логіку додатка (model). Зокрема, зараз популярний такий робочий процес, в якому дизайнери розробляють уявлення за допомогою одного засобу, розробники розробляють модель за допомогою іншого засобу, а інтегратори збирають модель представлення, використовуючи обидва засоби.

- Модульне тестування. Можна виконати модульне тестування моделі представлення (view-model) (а потім і незалежної від представлення логіки додатка (model)), окремо від представлення (view). Зберігаючи невеликий розмір представлення, ви зможете протестувати значну частину програми.

- Швидка адаптація до змін процесу взаємодії з користувачем. Представлення (view) найбільш часто піддається змінам, часто на пізніх етапах, так як взаємодія з користувачем оптимізується на основі відгуків користувачів.

Якщо уявлення міститься окремо, ці зміни можна вносити швидше і з меншим втручанням в роботу програми.

Висновки до розділу 2

В даному розділі розглянуто основні положення проектування системи тестування у вигляді технічного завдання та рекомендації MVC та MVVM шаблонів, що використовувались перед початком виконання завдань поставлених перед розробкою. Продумана модель та архітектура програмної реалізації додатку. Розроблені прототипи основних сторінок додатку, деталі його функціонування. Обраний шаблон проектування додатку. Можна сказати, що завдяки даному розділу отримано повністю описану систему, яку потрібно тільки реалізувати.

РОЗДІЛ 3

РЕАЛІЗАЦІЯ ПРОГРАМНОЇ ЧАСТИНИ ДОДАТКУ КОНСТРУКТОР ТЕСТІВ

3.1. Програмна частина конструктора тестів

Програмна частина проекту реалізована на мові C# з використанням об’єктної орієнтації та патерну проектування MVC.

Model-View-Controller (MVC, «Модель-Представлення-Контролер», «Модель-Вид-Контролер») - схема поділу даних програми, призначеного для користувача інтерфейсу і керуючої логіки на три окремих компоненти: модель, представлення і контролер - таким чином, що модифікація кожного компонента може здійснюватися незалежно.

Модель (Model) надає дані і реагує на команди контролера, змінюючи свій стан. Модель надає дані і методи роботи з ними: запити до бази даних, перевірка на коректність. Модель не залежить від уявлення і контролера, не знає як дані візуалізувати, не має точок взаємодії з користувачем, надаЄТЬСЯ доступ до даних і управління ними.

Будується таким чином, щоб відповідати на запити, змінюючи свій стан, при цьому може бути вбудовано повідомлення «спостерігачів».

За рахунок незалежності від візуального представлення, може мати кілька різних уявлень для однієї «моделі».

Представлення (View) відповідає за відображення даних моделі користувачеві, реагуючи на зміни моделі. Представлення відповідає за отримання необхідних даних з моделі і відправляє їх користувачеві. Не обробляє жодних введених користувачем даних.

Подання може впливати на стан моделі повідомляючи моделі про це.

Контролер (Controller) інтерпретує дії користувача, сповіщаючи модель про необхідність змін. Контролер забезпечує «зв'язок» між користувачем і системою. Контролює і направляє дані від користувача до системи і навпаки. Використовує модель і преставлення для реалізації необхідної дії.

Оскільки MVC не має суворої реалізації, то реалізований він може бути по-різному. Немає загальноприйнятого визначення, де повинна розташовуватися бізнес-логіка. Вона може знаходитися як в контролері, так і в моделі. В останньому випадку, модель буде містити всі бізнес-об'єкти з усіма даними і функціями.

Деякі платформи жорстко встановлюють де повинна розташовуватися бізнес-логіка, інші не мають таких правил.

Також, не обов’язково вказувати, де повинна знаходитися перевірка введених користувачем даних. Проста валідація може зустрічатися навіть у поданні, але частіше вони зустрічаються в контролері чи моделі.

Інтернаціоналізація і форматування даних також не має чітких вказівок по розташуванню.

Система також включає у себе шаблони для стеження за діями користувача у системі, тобто виконується логування в усіх вікнах.

Програмно реалізовано передачу об’єкта-користувача в усі потребуючи цього класи. Це дає змогу не повторювати реалізацію однакової частини коду у всіх класах, що суттєво знижує навантаження на ресурси пам’яті та процесора користувачів, а і приховати реалізацію даних. До цих шаблонів реалізована архітектура зв’язку з базою даних та безпосередньо до таблиць у ній, з правильним використанням транзитивності, що також суттєво знижує затрати ресурсів та зменшує час на запити до бази.

3.2. З’єднання модулів додатку

Розробка системи з використанням шалбону-патерну MVC передбачає три етапи:

- модель (Model)
- представлення (View)
- управління (Control).

У якості моделі даної системи виступає програмна частина додатку: описання класів-шаблонів, об’єктна-орієнтація та встановлення зв’язку з таблицями бази даних. Обробка подій на кожен активний елемент управління.

Представлення має зовнішній вигляд кожного вікна, розміщення елементів управління та відображення на вікнах даних з таблиць.

Управління передається безпосередньо мовою T-SQL, розширення запитів, які викликаються обробкою подій з елементів моделі.

Можливості.

- Доступ до вихідного коду можливий тільки розробнику.
- Створення необмеженої кількості білетів та питань.
- Налаштування автоматичного резервного копіювання даних користувачів.
- Запис помилок у системний лог-файл.

Переваги.

- Розповсюджується безкоштовно.
- Низькі затрати ресурсів (Рис.3.1).
- Зручний та інформативний інтерфейс додатку.
- Додаток розроблений на новітній технології з підтримкою подальшого оновлення, що дає змогу підключення зручних сервісів в подальшому користуванні.
- Можливість оновлення як через інтернет, так і шляхом копіювання оновленого ПЗ з носіїв будь-якого типу.

Основні характеристики:

- Необмежена кількість білетів та користувачів.
- Відмовостійкість
- Відкритий вихідний код
- Безкоштовна документація

- Підтримка усіх операційних систем Microsoft.

Вимоги до операційної системи:

- Windows XP або вище з встановленим пакетом .NET Framework 3.0
- Процесор Intel Inside або вище
- 1Гб оперативної пам’яті
- 5Мб дискового простору
- Графічний адаптер з підтримкою DirectX 9.0 або вище

Для встановлення та користування необхідно завантажити дистрибутив на локальний комп’ютер та запустити додаток.

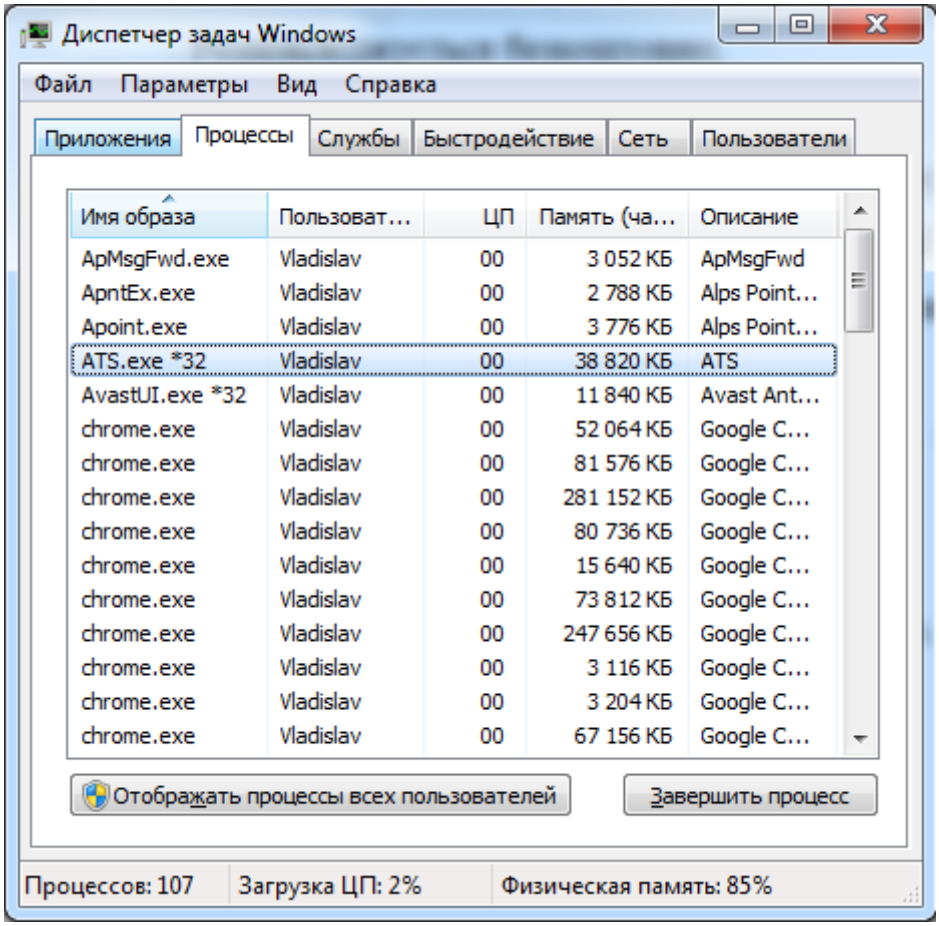


Рис. 3.1. Низькі затрати ресурсів.

3.3. Розробка основних вікон

На етапі розробки кожного окремого вікна, в першу чергу звертається увага на побажання замовника, а в подальшому, виконуються редагування зі сторони розробника. Стилістика додатку повинна мати однотонний характер,

для того, щоб зі сторони клієнта не виникало візуального дисонансу між різними елементами керування та навігації. Дизайн кожної форми різний. При проектуванні також враховується, з логічної точки зору: «Що це вікно буде відображати?» та «Як воно це буде відображати?». Особливо, якщо мова заходить про використання додатку на дисплеях та моніторах різної роздільної здатності, для збереження цілісності відображення інформації на кожній із них.

Коли робота над зовнішнім виглядом додатку завершена, наступає процес прив'язки стилів до основного коду. За допомогою мови розмітки додатків відбувається процес інтеграції зовнішнього вигляду до стартової точки(або точки заходу) додатку. Після прив'язки, до кожного вікна інтегруються програмні обробники подій та відбувається збірка рішення. Далі відбувається тестовий пуск додатку, а за ним слідує етап зв'язування БД з шаблонами, описаними об'єктно-орієнтованою мовою C#.

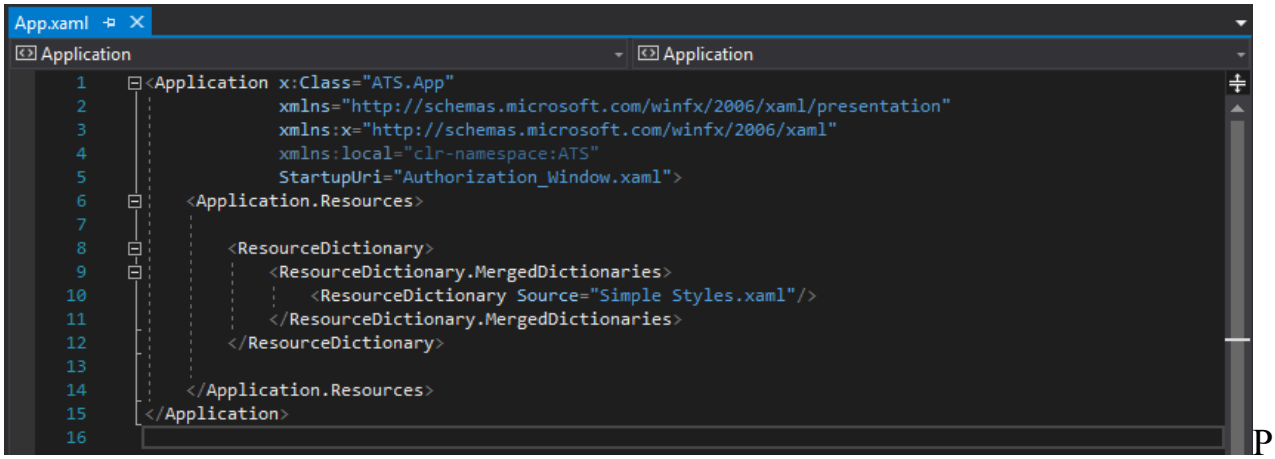


Рис. 3.2. Точка входу в додаток

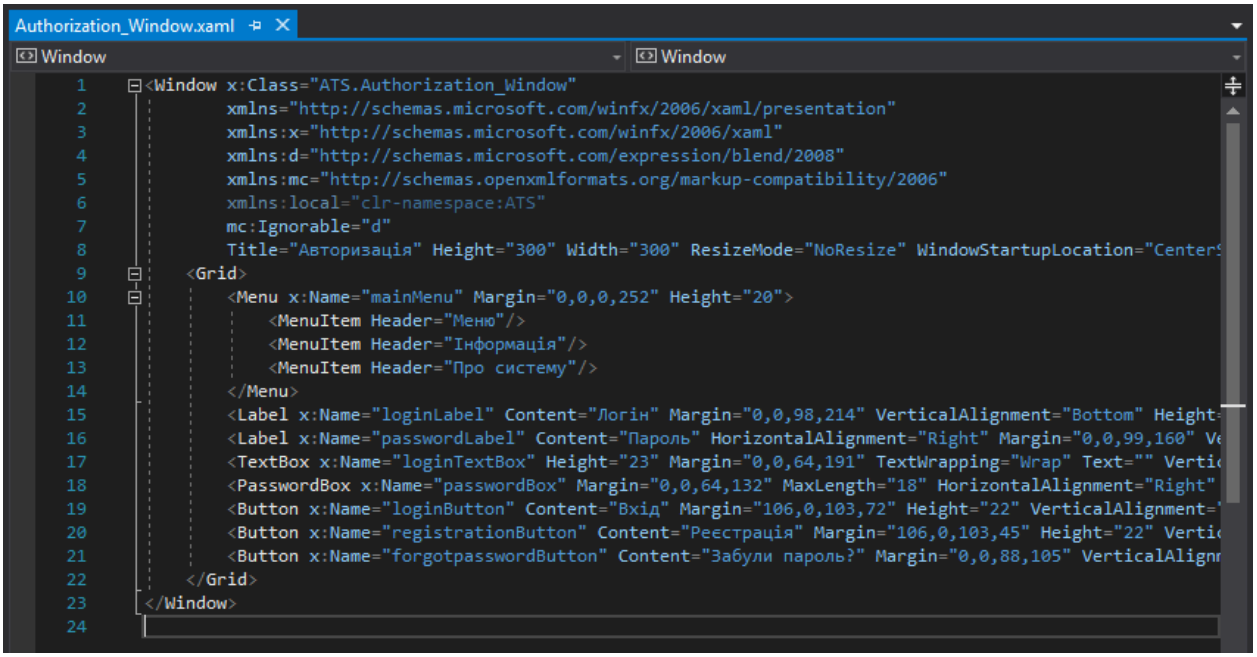


Рис. 3.3. Зв’язування вікна авторизації.

Розробкою графічного відображення вікон у додатку займаються дизайнери. В свою чергу їх задача, розписати усі елементи до відповідного стилю написання розробника, або згідно до побажань замовника.

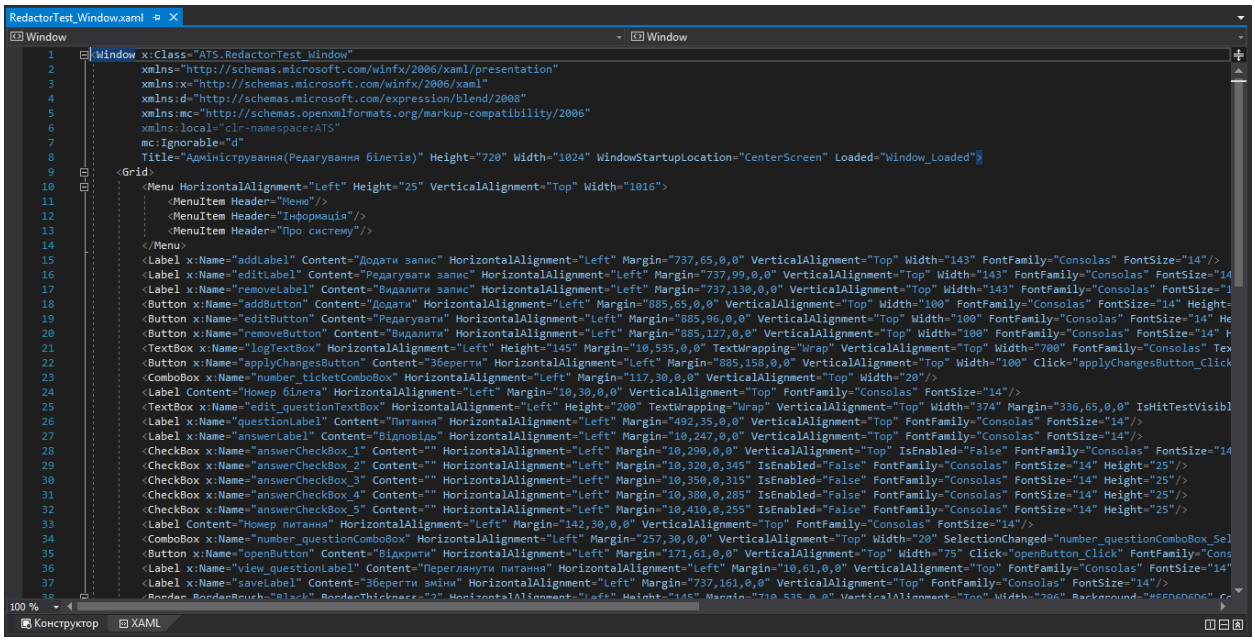


Рис. 3.4. Зв’язування вікна адміністрування.

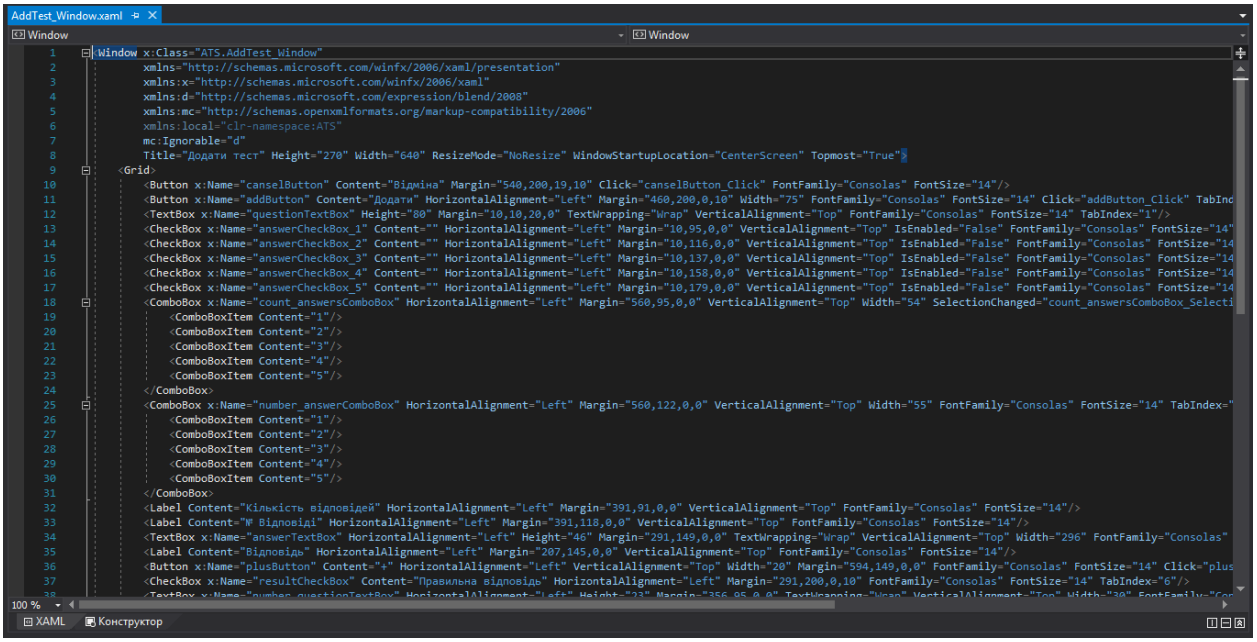


Рис.3.5. Зв’язування вікна додавання записів.

3.4. Доступ для адміністрування

Для того щоб увійти до системи в якості адміністратора або надати прав для адміністрування потрібно підключитись до бази даних в таблицю «Users», та зробити помітку у полі користувача в колонці «admin», змінивши значення з «0» на «1». Для того щоб виконати дану операцію необхідно встановити інструмент для перегляду бази Microsoft SQL Server Management Studio та увійти під логіном «admin» та паролем «admin» відповідно (рис.3.6).

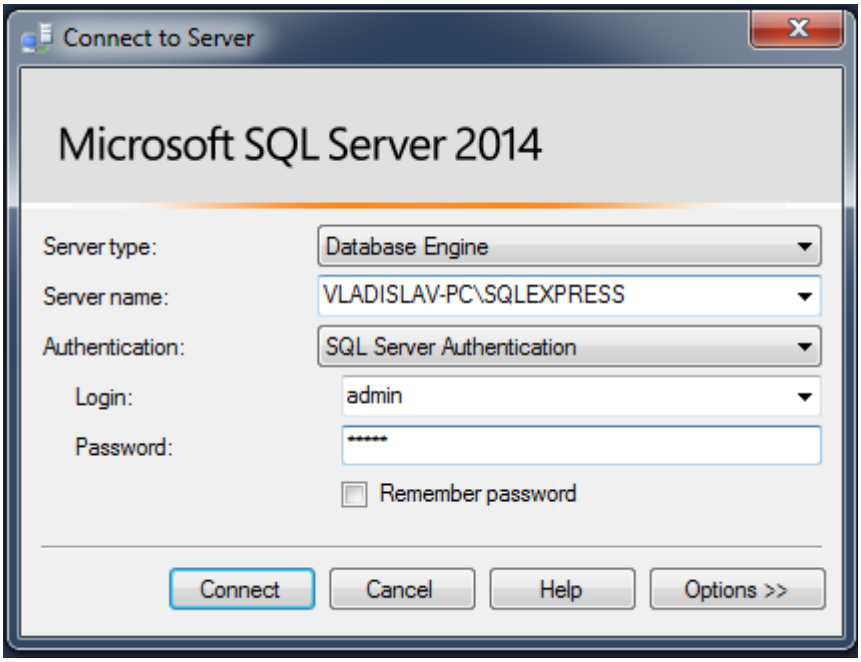


Рис.3.6. Вхід для адміністрування БД

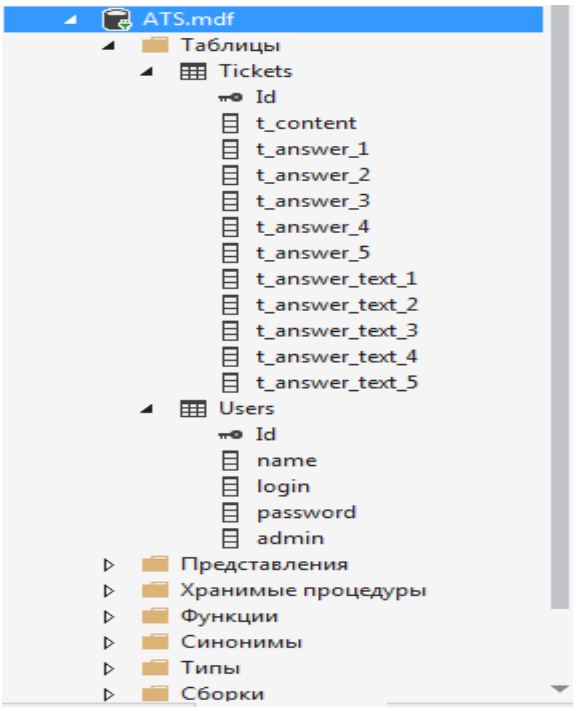


Рис.3.7. Відображення БД

3.5. Програмна реалізація основних елементів

Керування даними відбувається безпосередньо у самій системі. Для редагування даних, керування користувачами та доступом здійснюється шляхом авторизації у систему від імені адміністратора (Рис.3.8). У меню адміністратора є панелі редагування питань (Рис.3.9), відповідей, білетів та користувачів. Розширення прав користувачів до рівня адміністратора програмно виключена.

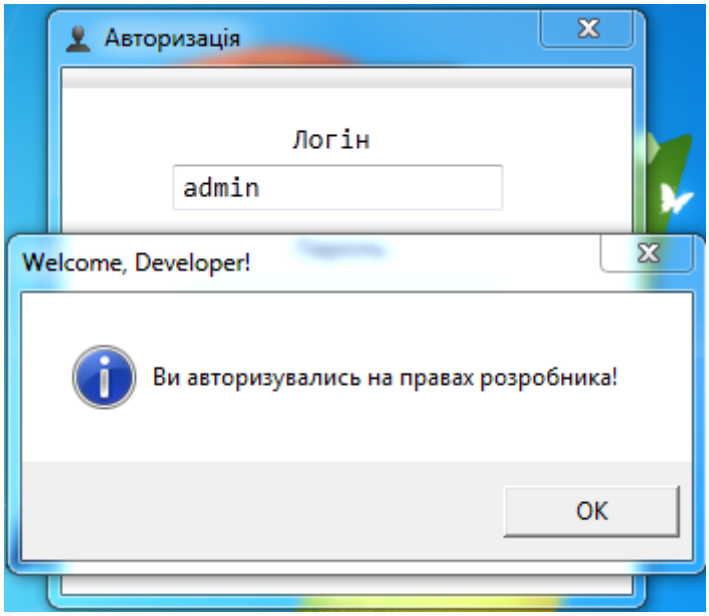


Рис. 3.8. Авторизація адміністратора

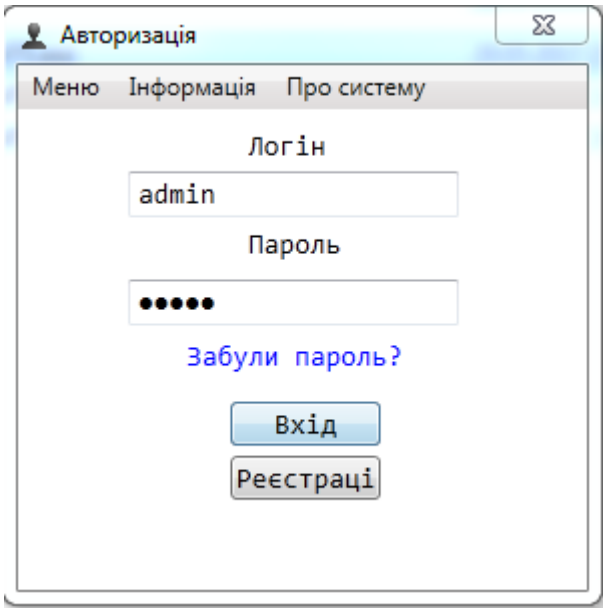
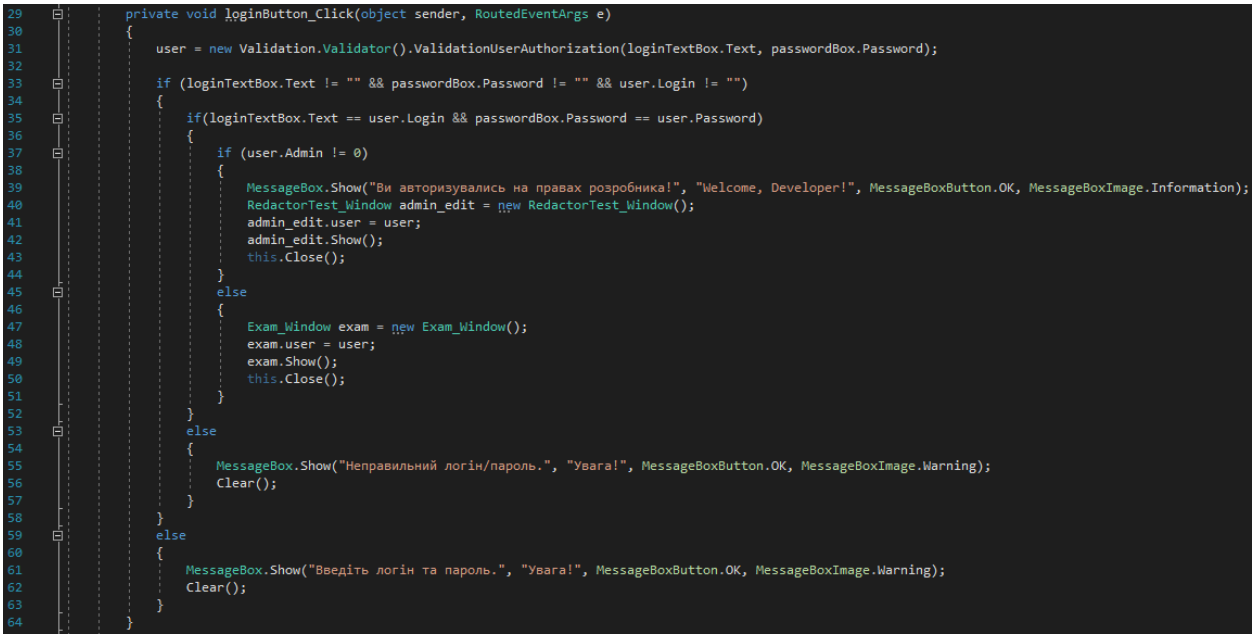


Рис.3.9. Авторизація користувачів

Для авторизації використовувались наступні елементи керування. Кнопка «Вхід». Даний функціонал реалізований за допомогою наступного коду (Рис.3.10):

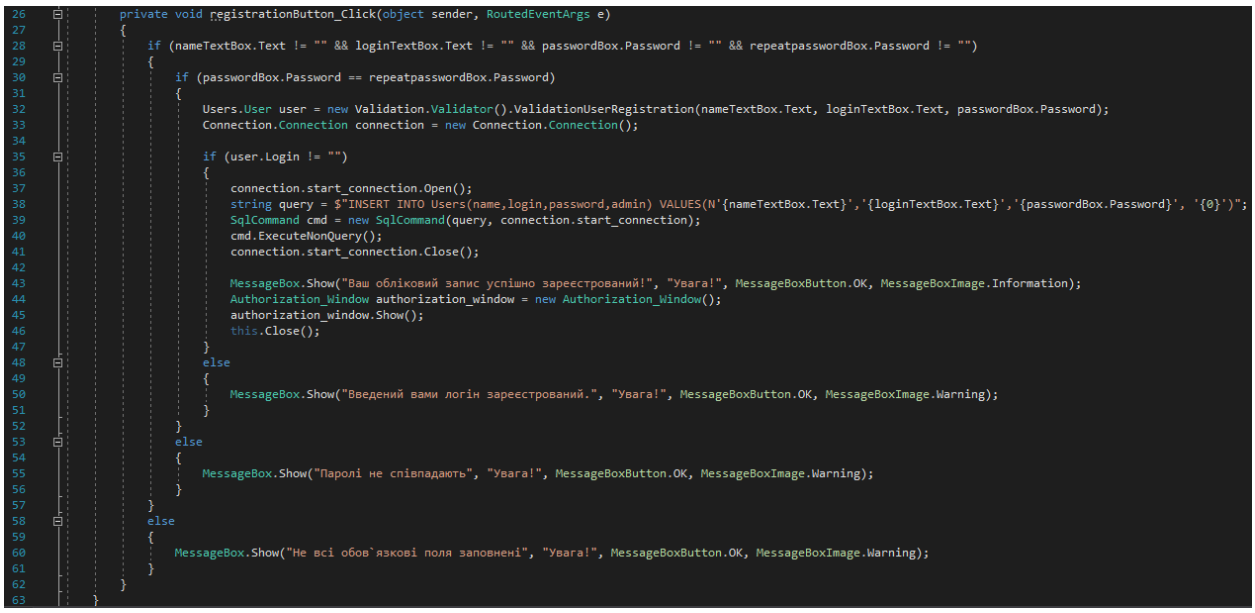


```
private void loginButton_Click(object sender, RoutedEventArgs e)
{
    user = new Validation.Validator().ValidationUserAuthorization(loginTextBox.Text, passwordBox.Password);

    if (loginTextBox.Text != "" && passwordBox.Password != "" && user.Login != "")
    {
        if(loginTextBox.Text == user.Login && passwordBox.Password == user.Password)
        {
            if (user.Admin != 0)
            {
                MessageBox.Show("Ви авторизувались на правах розробника!", "Welcome, Developer!", MessageBoxButton.OK, MessageBoxImage.Information);
                RedactorTest_Window admin_edit = new RedactorTest_Window();
                admin_edit.user = user;
                admin_edit.Show();
                this.Close();
            }
            else
            {
                Exam_Window exam = new Exam_Window();
                exam.user = user;
                exam.Show();
                this.Close();
            }
        }
        else
        {
            MessageBox.Show("Неправильний логін/пароль.", "Увага!", MessageBoxButton.OK, MessageBoxImage.Warning);
            Clear();
        }
    }
    else
    {
        MessageBox.Show("Введіть логін та пароль.", "Увага!", MessageBoxButton.OK, MessageBoxImage.Warning);
        Clear();
    }
}
```

Рис.3.10. Код реалізації кнопки «Вхід»

Схожий функціонал має обробка події кнопки «Реєстрація», яка знаходить у вікні реєстрації нового користувача(Рис.3.11).



```
private void registrationButton_Click(object sender, RoutedEventArgs e)
{
    if (nameTextBox.Text != "" && loginTextBox.Text != "" && passwordBox.Password != "" && repeatpasswordBox.Password != "")
    {
        if (passwordBox.Password == repeatpasswordBox.Password)
        {
            Users.User user = new Validation.Validator().ValidationUserRegistration(nameTextBox.Text, loginTextBox.Text, passwordBox.Password);
            Connection.Connection connection = new Connection.Connection();

            if (user.Login != "")
            {
                connection.start_connection.Open();
                string query = $"INSERT INTO Users(name,login,password,admin) VALUES(N'{nameTextBox.Text}','{loginTextBox.Text}','{passwordBox.Password}', '{0}')";
                SqlCommand cmd = new SqlCommand(query, connection.start_connection);
                cmd.ExecuteNonQuery();
                connection.start_connection.Close();

                MessageBox.Show("Ваш обліковий запис успішно зареєстрований!", "Увага!", MessageBoxButton.OK, MessageBoxImage.Information);
                Authorization_Window authorization_window = new Authorization_Window();
                authorization_window.Show();
                this.Close();
            }
            else
            {
                MessageBox.Show("Введений вами логін зареєстрований.", "Увага!", MessageBoxButton.OK, MessageBoxImage.Warning);
            }
        }
        else
        {
            MessageBox.Show("Паролі не співпадають", "Увага!", MessageBoxButton.OK, MessageBoxImage.Warning);
        }
    }
    else
    {
        MessageBox.Show("Не всі обов`язкові поля заповнені", "Увага!", MessageBoxButton.OK, MessageBoxImage.Warning);
    }
}
```

Рис.3.11. Код реалізації кнопки «Реєстрація»

Кнопка «Забули пароль» має дещо інакшу структуру тому, що у момент натиснення кнопки «Відновити», що знаходиться у вікні «Відновлення паролю», відбувається генерація паролю програмно, який випадковим чином формує тимчасовий пароль довжиною 6 символів і одразу передає його у таблицю «Users» (Рис.3.12).

```
private void recoverpasswordButton_Click(object sender, RoutedEventArgs e)
{
    if (recoverpasswordTextBox.Text != "")
    {
        Connection.Connection connection = new Connection.Connection();
        Random random = new Random();
        string temppass = "";
        Char[] pwdChars = new Char[36]
        {   'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h',
        'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p',
        'q', 'r', 's', 't', 'u', 'v', 'w', 'x',
        'y', 'z', '0', '1', '2', '3', '4', '5',
        '6', '7', '8', '9'
        };
        for (int i = 0; i < 6; i++)
        {
            temppass += pwdChars[random.Next(0, 35)];
        }

        connection.start_connection.Open();
        string query = $"UPDATE Users SET password = '{temppass}' WHERE login = '{recoverpasswordTextBox.Text}'";
        SqlCommand cmd = new SqlCommand(query, connection.start_connection);
        cmd.ExecuteNonQuery();
        connection.start_connection.Close();

        MessageBox.Show("Ваш новий пароль: " + temppass.ToString(),
            "Відновлення паролю",
            MessageBoxButton.OK, MessageBoxImage.Information);
        recoverpasswordTextBox.Clear();
        Authorization_Window authorization_window = new Authorization_Window();
        authorization_window.Show();
        this.Close();
    }
    else
    {
        MessageBox.Show("Будь-ласка, заповніть поле для відновлення паролю.", "Увага!", MessageBoxButton.OK, MessageBoxImage.Warning);
    }
}
}
```

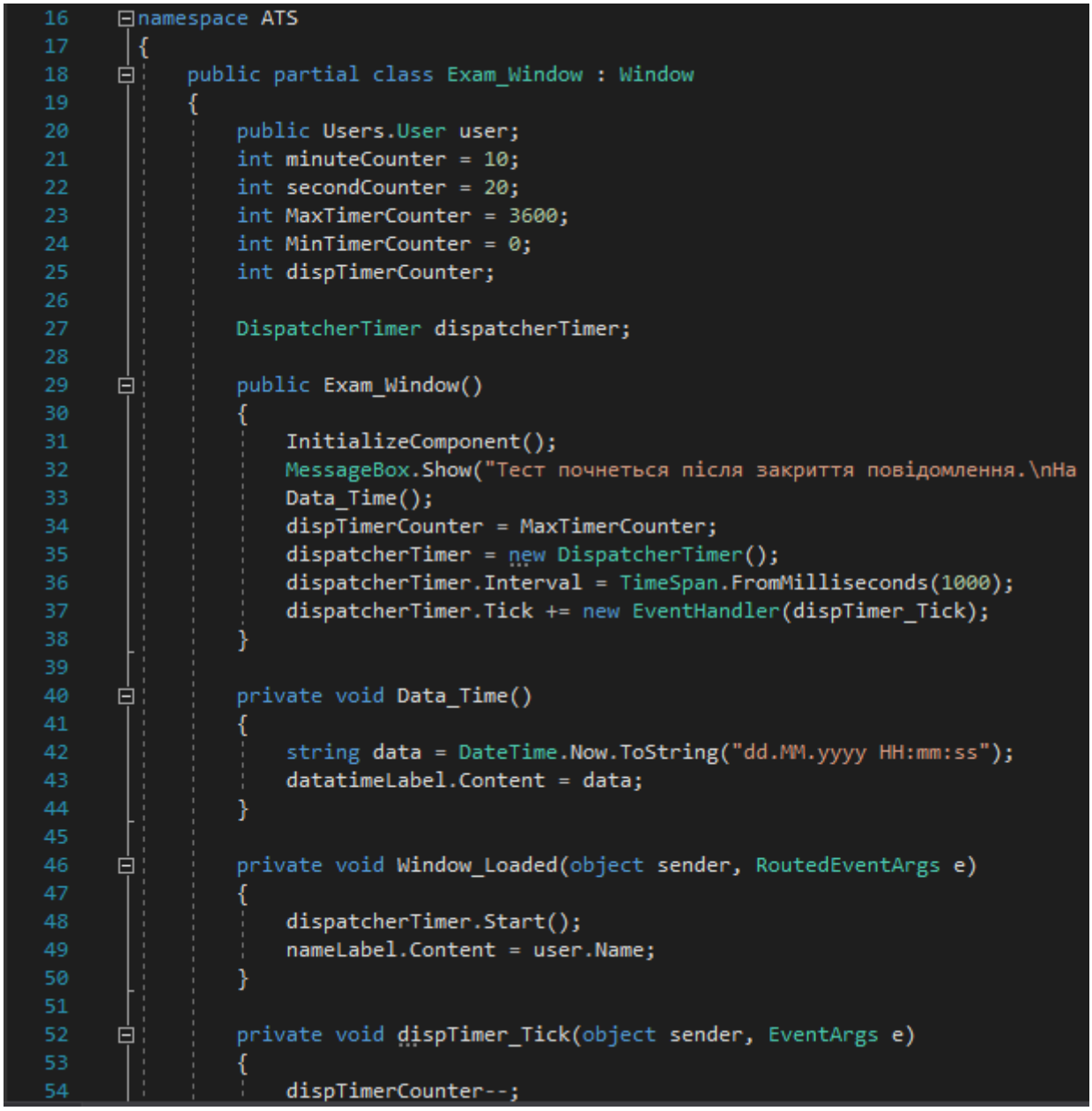
Рис.3.12. Код реалізації кнопки відновлення паролю

Реалізація XAML коду для вікна «Авторизація» та усіх елементів керування та навігації (Рис.3.13).

```
Authorization_Window.xaml
Window                                                    Window
<Window x:Class="ATS.Authorization_Window"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:ATS"
        mc:Ignorable="d"
        Title="Авторизація" Height="300" Width="300" ResizeMode="NoResize" WindowStartupLocation="CenterScreen" Background="White" Icon="Resources/user.png" Topmost="True">
    <Grid>
        <Menu x:Name="mainMenu" Margin="0,0,0,240" Height="32">
            <MenuItem Header="Меню"/>
            <MenuItem Header="Інформація"/>
            <MenuItem Header="Про систему"/>
        </Menu>
        <Label x:Name="loginLabel" Content="Логін" Margin="0,0,98,209" VerticalAlignment="Bottom" Height="26" FontFamily="Consolas" FontSize="14" HorizontalAlignment="Right" Widt
        <Label x:Name="passwordLabel" Content="Пароль" HorizontalAlignment="Right" Margin="0,0,99,160" VerticalAlignment="Bottom" Height="26" Width="75" FontFamily="Consolas" For
        <TextBox x:Name="loginTextBox" Height="23" Margin="0,0,64,186" TextWrapping="Wrap" Text="" VerticalAlignment="Bottom" MaxLength="25" HorizontalAlignment="Right" Width="16
        <PasswordBox x:Name="passwordBox" Margin="0,0,64,132" MaxLength="18" HorizontalAlignment="Right" Width="165" Height="23" VerticalAlignment="Bottom" FontFamily="Consolas"
        <Button x:Name="loginButton" Content="Вхід" Margin="106,0,103,72" Height="22" VerticalAlignment="Bottom" FontFamily="Consolas" FontSize="14" TabIndex="4" Click="loginButt
        <Button x:Name="registrationButton" Content="Реєстрація" Margin="106,0,103,45" Height="22" VerticalAlignment="Bottom" FontFamily="Consolas" FontSize="14" TabIndex="5" Cli
        <Button x:Name="forgotpasswordButton" Content="Забули пароль?" Margin="0,0,88,105" VerticalAlignment="Bottom" BorderBrush="{x:Null}" Background="{x:Null}" FontFamily="Con
    </Grid>
</Window>
```

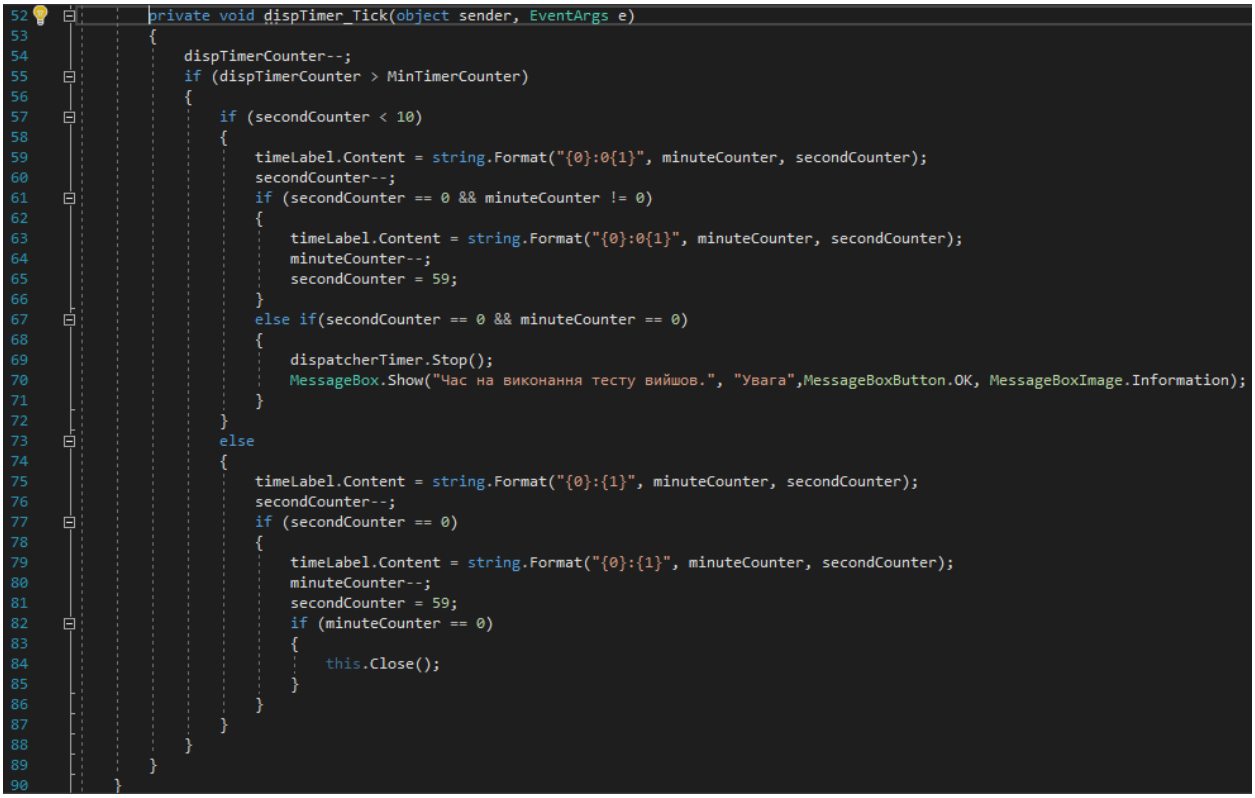
Рис.3.13. Код вікна «Авторизація»

Головним вікном для проходження тесту є «Тестування». Функціональний код для реалізації інформаційних надписів у вікні (Рис.3.14, рис.3.15).



```
16  namespace ATS
17  {
18      public partial class Exam_Window : Window
19      {
20          public Users.User user;
21          int minuteCounter = 10;
22          int secondCounter = 20;
23          int MaxTimerCounter = 3600;
24          int MinTimerCounter = 0;
25          int dispTimerCounter;
26
27          DispatcherTimer dispatcherTimer;
28
29          public Exam_Window()
30          {
31              InitializeComponent();
32              MessageBox.Show("Тест почнеться після закриття повідомлення.\nНа
33              Data_Time();
34              dispTimerCounter = MaxTimerCounter;
35              dispatcherTimer = new DispatcherTimer();
36              dispatcherTimer.Interval = TimeSpan.FromMilliseconds(1000);
37              dispatcherTimer.Tick += new EventHandler(dispTimer_Tick);
38          }
39
40          private void Data_Time()
41          {
42              string data = DateTime.Now.ToString("dd.MM.yyyy HH:mm:ss");
43              datatimeLabel.Content = data;
44          }
45
46          private void Window_Loaded(object sender, RoutedEventArgs e)
47          {
48              dispatcherTimer.Start();
49              nameLabel.Content = user.Name;
50          }
51
52          private void dispTimer_Tick(object sender, EventArgs e)
53          {
54              dispTimerCounter--;
```

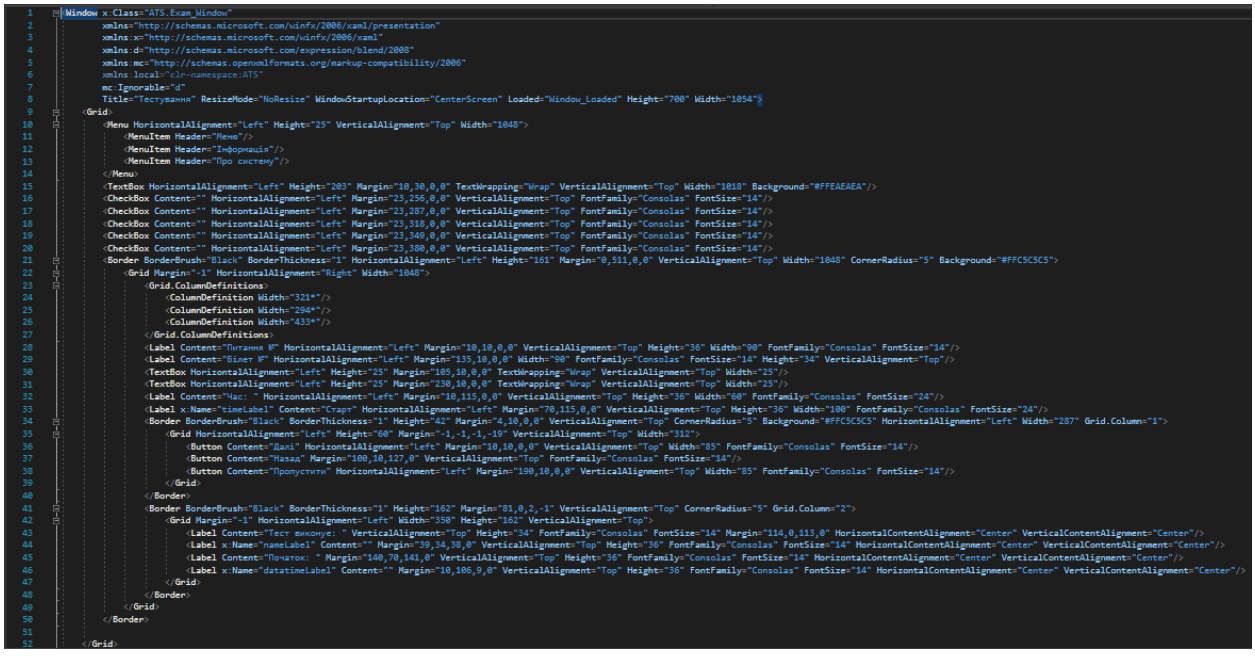
Рис.3.14. Реалізація функцій вікна «Тестування»



```
private void dispTimer_Tick(object sender, EventArgs e)
{
    dispTimerCounter--;
    if (dispTimerCounter > MinTimerCounter)
    {
        if (secondCounter < 10)
        {
            timeLabel.Content = string.Format("{0}:0{1}", minuteCounter, secondCounter);
            secondCounter--;
            if (secondCounter == 0 && minuteCounter != 0)
            {
                timeLabel.Content = string.Format("{0}:0{1}", minuteCounter, secondCounter);
                minuteCounter--;
                secondCounter = 59;
            }
            else if(secondCounter == 0 && minuteCounter == 0)
            {
                dispatcherTimer.Stop();
                MessageBox.Show("Час на виконання тесту вийшов.", "Увага",MessageBoxButton.OK, MessageBoxImage.Information);
            }
        }
        else
        {
            timeLabel.Content = string.Format("{0}:{1}", minuteCounter, secondCounter);
            secondCounter--;
            if (secondCounter == 0)
            {
                timeLabel.Content = string.Format("{0}:{1}", minuteCounter, secondCounter);
                minuteCounter--;
                secondCounter = 59;
                if (minuteCounter == 0)
                {
                    this.Close();
                }
            }
        }
    }
}
```

Рис.3.15. Реалізація функцій вікна «Тестування»

Також код реалізації вікна, випадаючих списків, чекбоксів «Тестування» (Рис.3.16).



```
<Window x:Class="ATS.Exam_Window"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:ATS"
        mc:Ignorable="d"
        Title="Тестування" ResizeMode="NoResize" WindowStartupLocation="CenterScreen" Loaded="Window_Loaded" Height="700" Width="1054">
    <Grid>
        <Menu HorizontalAlignment="Left" Height="25" VerticalAlignment="Top" Width="1048">
            <MenuItem Header="Меню"/>
            <MenuItem Header="Інформація"/>
            <MenuItem Header="Про систему"/>
        </Menu>
        <TextBox HorizontalAlignment="Left" Height="203" Margin="10,30,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="1018" Background="#FFEAEAEA"/>
        <CheckBox Content="" HorizontalAlignment="Left" Margin="23,256,0,0" VerticalAlignment="Top" FontFamily="Consolas" FontSize="14"/>
        <CheckBox Content="" HorizontalAlignment="Left" Margin="23,287,0,0" VerticalAlignment="Top" FontFamily="Consolas" FontSize="14"/>
        <CheckBox Content="" HorizontalAlignment="Left" Margin="23,318,0,0" VerticalAlignment="Top" FontFamily="Consolas" FontSize="14"/>
        <CheckBox Content="" HorizontalAlignment="Left" Margin="23,349,0,0" VerticalAlignment="Top" FontFamily="Consolas" FontSize="14"/>
        <CheckBox Content="" HorizontalAlignment="Left" Margin="23,380,0,0" VerticalAlignment="Top" FontFamily="Consolas" FontSize="14"/>
        <Border BorderBrush="Black" BorderThickness="1" HorizontalAlignment="Left" Height="161" Margin="0,511,0,0" VerticalAlignment="Top" Width="1048" CornerRadius="5" Background="#FFC5C5C5">
            <Grid Margin="-1" HorizontalAlignment="Right" Width="1048">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="321*"/>
                    <ColumnDefinition Width="294*"/>
                    <ColumnDefinition Width="433*"/>
                </Grid.ColumnDefinitions>
                <Label Content="Питання №" HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top" Height="36" Width="90" FontFamily="Consolas" FontSize="14"/>
                <Label Content="Білет №" HorizontalAlignment="Left" Margin="135,10,0,0" Width="90" FontFamily="Consolas" FontSize="14" Height="34" VerticalAlignment="Top"/>
                <TextBox HorizontalAlignment="Left" Height="25" Margin="105,10,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="25"/>
                <TextBox HorizontalAlignment="Left" Height="25" Margin="230,10,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="25"/>
                <Label Content="Час: " HorizontalAlignment="Left" Margin="10,115,0,0" VerticalAlignment="Top" Height="36" Width="60" FontFamily="Consolas" FontSize="24"/>
                <Label x:Name="timeLabel" Content="Старт" HorizontalAlignment="Left" Margin="70,115,0,0" VerticalAlignment="Top" Height="36" Width="100" FontFamily="Consolas" FontSize="24"/>
                <Border BorderBrush="Black" BorderThickness="1" Height="42" Margin="4,10,0,0" VerticalAlignment="Top" CornerRadius="5" Background="#FFC5C5C5" HorizontalAlignment="Left" Width="287" Grid.Column="1">
                    <Grid HorizontalAlignment="Left" Height="60" Margin="-1,-1,-1,-19" VerticalAlignment="Top" Width="312">
                        <Button Content="Далі" HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top" Width="85" FontFamily="Consolas" FontSize="14"/>
                        <Button Content="Назад" Margin="100,10,127,0" VerticalAlignment="Top" FontFamily="Consolas" FontSize="14"/>
                        <Button Content="Пропустити" HorizontalAlignment="Left" Margin="190,10,0,0" VerticalAlignment="Top" Width="85" FontFamily="Consolas" FontSize="14"/>
                    </Grid>
                </Border>
                <Border BorderBrush="Black" BorderThickness="1" Height="162" Margin="81,0,2,-1" VerticalAlignment="Top" CornerRadius="5" Grid.Column="2">
                    <Grid Margin="-1" HorizontalAlignment="Left" Width="350" Height="162" VerticalAlignment="Top">
                        <Label Content="Тест виконує: " VerticalAlignment="Top" Height="34" FontFamily="Consolas" FontSize="14" Margin="114,0,113,0" HorizontalContentAlignment="Center" VerticalContentAlignment="Center"/>
                        <Label x:Name="nameLabel" Content="" Margin="30,34,38,0" VerticalAlignment="Top" Height="36" FontFamily="Consolas" FontSize="14" HorizontalContentAlignment="Center" VerticalContentAlignment="Center"/>
                        <Label Content="Початок: " Margin="140,70,141,0" VerticalAlignment="Top" Height="36" FontFamily="Consolas" FontSize="14" HorizontalContentAlignment="Center" VerticalContentAlignment="Center"/>
                        <Label x:Name="datatimeLabel" Content="" Margin="10,106,9,0" VerticalAlignment="Top" Height="36" FontFamily="Consolas" FontSize="14" HorizontalContentAlignment="Center" VerticalContentAlignment="Center"/>
                    </Grid>
                </Border>
            </Grid>
        </Border>

    </Grid>
```

Рис.3.16. Реалізація вікна «Тестування»

3.6. Інтерфейс користувача системи конструктор тестів

Вікно форми налагодження програми (рис.3.17).

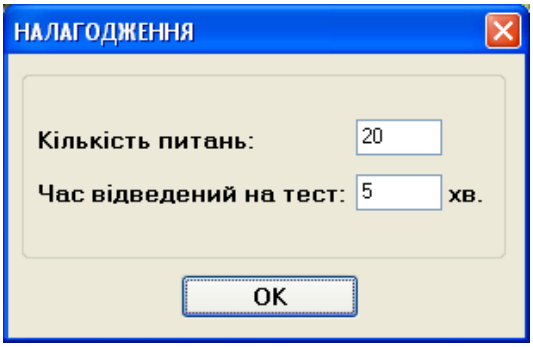


Рис. 3.17. Вікно форми налагодження програми

Панель редагування білетів, тестів, користувачів (рис.3.18) та панель додавання записів (рис.3.19).

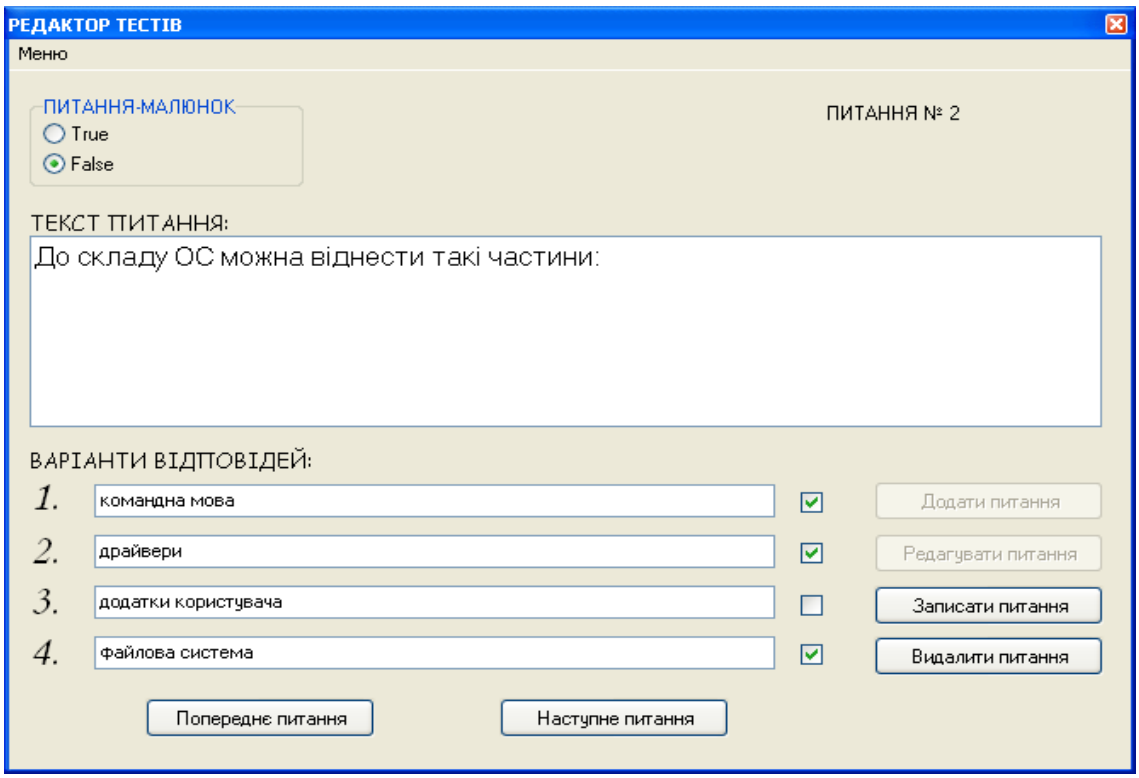


Рис. 3.18. Панель редагування білетів, тестів, користувачів

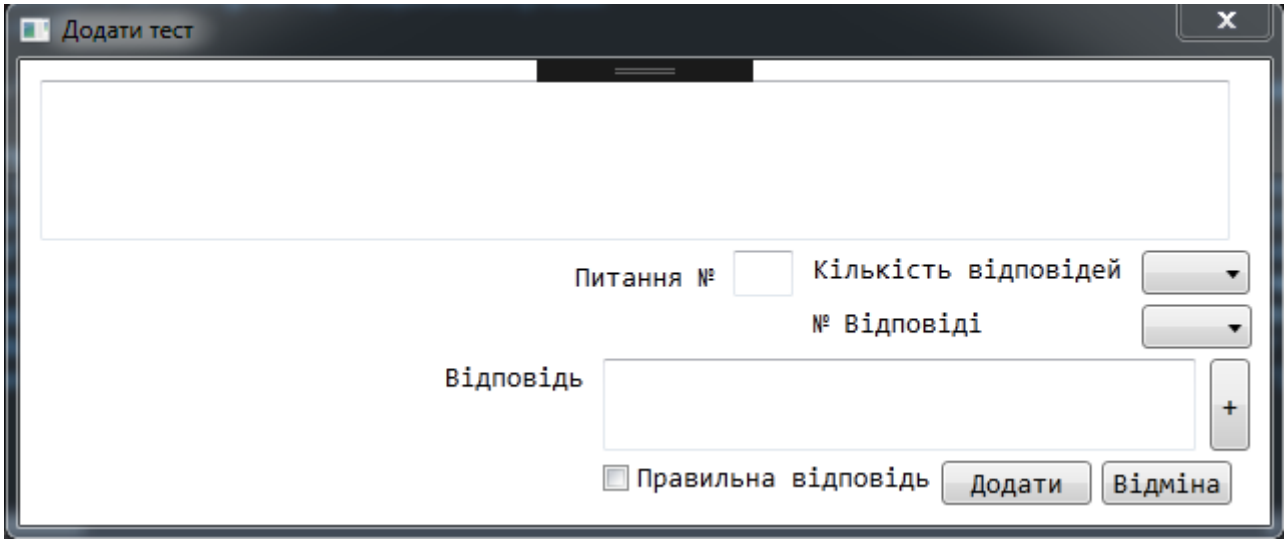


Рис. 3.19. Панель додавання записів

Вікно форми виведення результату тесту (рис.3.20).

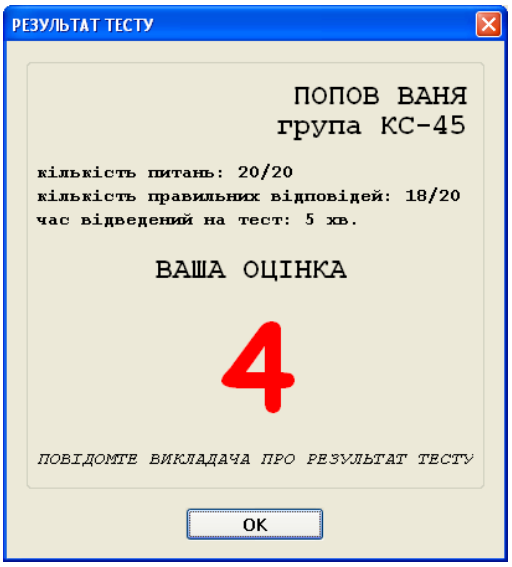


Рис. 3.20. Вікно форми виведення результату тесту

Висновки до розділу 3

У ході виконання дипломного проекту розроблено систему конструктор тестів, повністю готовий до застосування. Дана система здатна тестувати користувачів у необхідній області. Серед аналогів, система гнучка у налаштуванні та має інтерфейс налаштований на користувача.

За допомогою даного додатку, користувачі можуть складати тести у зручному для них місті та у зручний час, переглядати результати по тестам, бути впевненим у чесному та правильному оцінюванні.

При розробці були використані сучасні технології проектування та актуальна інформація, яка дозволила створити продукт високої якості та з розрахунком на подальшу модернізацію.

Реалізована автоматизована система тестування може використовуватися для контролю знань студентів у ВУЗах.

ВИСНОВКИ

Згідно основних особливостей використання комп’ютерного тестування, розглянуті проблеми фактори, які розкривають необхідність переходу на сучасне електронне ПЗ, що призване замінити методології контролю якості знання, такі як: усні та паперові. У дипломному проекті описане проектування системи конструктор тестів. Реалізовано програмний засіб тестування, який є частиною автоматизованої системи. Це дозволить автоматизувати та удосконалити процес складання тестів та полегшить процедуру проходження самого тестування у багатьох галузях країни.

Запропонована загальна архітектура системи конструктора тестів, в якій визначені всі необхідні компоненти.

Розглянуто та проаналізовано основні компоненти системи конструктора тестів та визначені основні вимоги до комп’ютерної системи.

Даний програмний засіб сприятиме підвищенню актуальності та вдосконаленню електронних систем тестування, звільнить час висококваліфікованих спеціалістів та полегшить трудомісткий процес перевірки тестів, дозволить їм швидко та на професійному рівні оцінювати належність та якість перевірених знань.

З появою таких програмних засобів у державних структурах, зменшиться кількість паперових тестів, що сприятиме розміщенню усіх документів у електронній базі, та значно прискорить процес здачі екзаменів за однаковий проміжок часу.