

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Hospital Management Studio»

Виконав студент групи ПЗк-20-1
ВОЙТКО Павло Олексійович

Керівник роботи:
ЧИЖМОТРЯ Олена Геннадіївна

Рецензент:
ЄФРЕМОВ Юрій Миколайович

Житомир – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

В.о. зав. кафедри інженерії
програмного забезпечення
Андрій МОРОЗОВ
«14» лютого 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **Войтко Павло Олексійович**

Керівник роботи: **ЧИЖМОТРЯ Олена Геннадіївна**

Тема роботи: **«Hospital Management Studio»**,

затверджена Наказом закладу вищої освіти від **«14» лютого 2023 р., №70с**

Вихідні дані для роботи: використання десктоп-орієнтованих технологій для розробки зручного та ефективного інструментарію для керування лікарнею.

Предметом дослідження є використання сучасних функціональних можливостей для виконання поставленої мети.

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О.Г.	16.02.2023	16.02.2023
2	Чижмотря О.Г.	16.02.2023	16.02.2023
3	Чижмотря О.Г.	16.02.2023	16.02.2023

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка завдання	16 лютого 2023 – 18 лютого 2023	виконано
2	Аналіз аналогів програмного продукту	18 лютого 2023 – 23 лютого 2023	виконано
3	Визначення вимог до додатку	23 лютого 2023 – 2 березня 2023	виконано
4	Вибір інструментальних засобів	2 березня 2023 – 4 березня 2023	виконано
5	Визначення варіантів використання	4 березня 2023 – 11 березня 2023	виконано
6	Проектування та реалізація алгоритмів системи	11 березня 2023 – 25 березня 2023	виконано
7	Написання програмного коду	25 березня 2023 – 1 травня 2023	виконано
8	Тестування роботи програмного продукту	1 травня 2023 – 12 травня 2023	виконано
9	Оформлення пояснювальної записки	12 травня 2023 – 30 травня 2023	виконано
10	Попередній захист	5 травня 2023	виконано

РЕФЕРАТ

Випускна робота бакалавра складається з Hospital Management Studio та пояснювальної записки. Пояснювальна записка до випускної роботи містить 81 сторінку, 34 ілюстрацій, 16 таблиць, 4 додатка та 17 використаних джерел.

Метою роботи є розробка та реалізація додатку «Hospital Management Studio». В роботі було визначено основні завдання на розробку додатку, проаналізовано аналогічні програмні продукти, визначено основний функціонал та вимоги до системи. Також було обрано технології для розробки додатку, спроектовано та реалізовано алгоритми роботи системи. Програмний продукт був повністю створений та протестований.

КЛЮЧОВІ СЛОВА: MSSQL, T-SQL, C#, NET.FRAMEWORK.

					ІПЗ.КР.Б – 121 – 23 - ПЗ						
Змн.	Арк.	№ докум.	Підпис	Дата							
Розроб.		П.О. Войтко			Пояснювальна записка			Літ.	Арк.	Аркушів	
Керівник.		О.Г. Чижмотря								4	81
Реценз.		Ю.М. Єфремов			Hospital Management Studio			Житомирська політехніка, група ІПЗк-20-І			
Зав. Каф..		А.В. Морозов									

ABSTRACT

The bachelor's thesis consists of the Hospital Management Studio application and an explanatory report. The explanatory report for the thesis includes 81 pages, 34 illustrations, 16 tables, 4 appendices, and 17 cited sources.

The aim of this work is to develop and implement the "Hospital Management Studio" application. The main tasks for application development were defined, similar software products were analyzed, and the main functionality and system requirements were determined. Technologies for application development were selected, and system algorithms were designed and implemented. The software product was fully created and tested.

		П.О. Войтко			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						5
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ВСТУП.....	7
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	8
РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ	9
1.1. Технічне завдання на розробку системи	9
1.2. Аналіз аналогів програмного продукту.....	11
1.3. Інструментальні засоби реалізації програмного продукту.....	14
РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ДОДАТКУ	17
2.1. Визначення варіантів використання системи	17
2.2. Алгоритми роботи системи	21
2.3. Структура компонентів системи	24
2.4. Реалізація додатку	26
РОЗДІЛ 3. ПОРЯДОК РОБОТИ ТА ТЕСТУВАННЯ СИСТЕМИ	39
3.1. Архітектура та розгортання додатку	39
3.2. Інтерфейс та порядок роботи з системою	41
3.3. Тестування роботи додатку	46
ВИСНОВКИ	50
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	51
ДОДАТКИ	53

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми: Актуальність Hospital Management Studio викликана тим, що в даний час все частіше потрібно розробити таку модель, яка б забезпечувала кращу наочність його організації та діяльності і, як наслідок, була б більш зручною для подальшої реалізації. Актуальність підтверджується існуванням безлічі систем для виконання даних завдань.

Мета даної роботи: Розробка комплексного рішення для управління медичними закладами, яке спрощує та автоматизує процеси управління, покращує ефективність роботи та забезпечує кращу координацію між медичним персоналом.

Об'єкт дослідження: Об'єктом дослідження є розробка та впровадження прогресивного застосунку, який ефективно управляє різними аспектами функціонування лікарень та поліпшує загальне керування охороною здоров'я

Предмет дослідження: Предметом дослідження є аналіз, розробка та впровадження комплексного застосунку для управління медичними закладами. Дослідження спрямоване на вивчення та покращення процесів управління, медичним персоналом.

Методи дослідження: у процесі розробки додатку будуть використані наступні методи дослідження: аналіз літератури та досліджень, аналіз вимог та моделювання, проектування архітектури та інтерфейсу додатку, розробка та тестування програмного забезпечення. Крім того, для забезпечення коректності та ефективності додатку будуть використані методи тестування.

Загальний результат дослідження буде представлений у вигляді функціонального додатку «Hospital Management Studio», який покращить ефективність роботи та забезпечить кращу координацію між медичним персоналом.

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

CRUD – (create read update delete) 4 основні функції управління даними «створення, читання, оновлення і вилучення».

БД – База даних.

T-SQL – Transact-SQL (T-SQL) є діалектом мови запитів SQL (Structured Query Language).

WPF – Windows Presentation Foundation (Фреймворк мови програмування C#).

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ

1.1. Технічне завдання на розробку системи

Аналіз проблематики включає оцінку актуальності теми, пояснення необхідності розробки Hospital Management Studio та виявлення основних проблем, з якими стикаються медичні заклади та їх персонал.

Проводиться розгляд бізнес-вимог, вимог користувачів, функціональних та нефункціональних вимог, системних вимог, бізнес-правил та обмежень. Це дозволяє визначити конкретні функції, які має виконувати Hospital Management Studio, враховуючи потреби користувачів та вимоги бізнесу, а також необхідні обмеження і вимоги для розробки системи з використанням C#, WPF та T-SQL [12].

Даний розділ описує високорівневі вимоги, якими повинна володіти «Hospital Management Studio», щоб мати можливість безперервно виконувати покладені на неї функції, тим самим відповідно задовольнивши всі зазначені стандарти, специфікації та інші формальні документи. Зазначені умови, яких повинен дотримуватись користувач для роботи з даною інформаційною системою.

Бізнес-вимоги:

1. Основні цілі: проект створюється з метою обліку лікарів.
2. Звіт: надання користувачам контролювати фінансову частину малого бізнесу.
3. Представлення проекту: проект буде реалізовано у додатку, що містить структуровану інформацію по даній тематичній області.

Вимоги користувачів:

Зовнішні користувачі:

1. Перегляд основних даних по лікарям, їх відпусткам, спеціалізацій, департаментів.

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

2. Пошук по ключовим словам.
3. Друк відповідної таблиці.
4. Перегляд фінансових репортів.

Внутрішні користувачі:

1. CRUD БД [3]
2. Редагування дизайну інтерфейсу.
3. Бекап БД

Функціональні вимоги:

1. Авторизація користувачів в системі: В системі повинна бути представлення можливість реєстрації користувача через БД та присвоєння йому відповідної ролі (, бухгалтер, адміністратор).
2. Можливість збереження інформації: Система повинна зберігати інформацію і надавати можливість користувачу керувати нею.
3. Можливість вивести на друк відповідну таблицю.
4. Можливість пошуку по ключовому слову.

Нефункціональні вимоги:

1. Продуктивність - таблиця має завантажуватися макс. за 2 сек.
2. Технічне обслуговування – ремонтпридатність програмної системи та легкість підтримки системи.
3. Локалізація – підтримка різних мов в додатку [17].
4. Безпека – паролі повинні зберігатися в форматі SHA256.

Системні вимоги:

1. Вимоги до середовища виконання: Система повинна задовольняти вимогам на комп'ютері, що знаходиться в наступній мінімальній комплектації:
 1. 128Мб пам'яті
 2. Процесор з тактовою частотою 600MHz
 3. Операційна система Windows 98

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

4. Загальна освіта з інформатики для користувачів.
2. Вимоги до СУБД та доступу до даних:
 1. У ядрі системи повинна бути представлена СУБД реляційного доступу.

Бізнес правила:

1. Система повинна відповідати всім стандартам інтерфейса користувача Microsoft Windows.

Антивимоги:

1. Не використовувати інформацію, заборонену нормами моралі та законодавством.
2. Не використовувати інформацію, що є недійсною або може зашкодити психіці користувачів.

1.2. Аналіз аналогів програмного продукту

Під час проведення аналізу аналогів були виявлені наступні подібні додатки.

Переглянувши веб-сайти та додатки, я виділив один з них, а саме Hospital Information System. Цей застосунок (рис. 1.1) є десктопним і дозволяє вести облік лікарів та створювати записи залежно від потреб медичного персоналу.

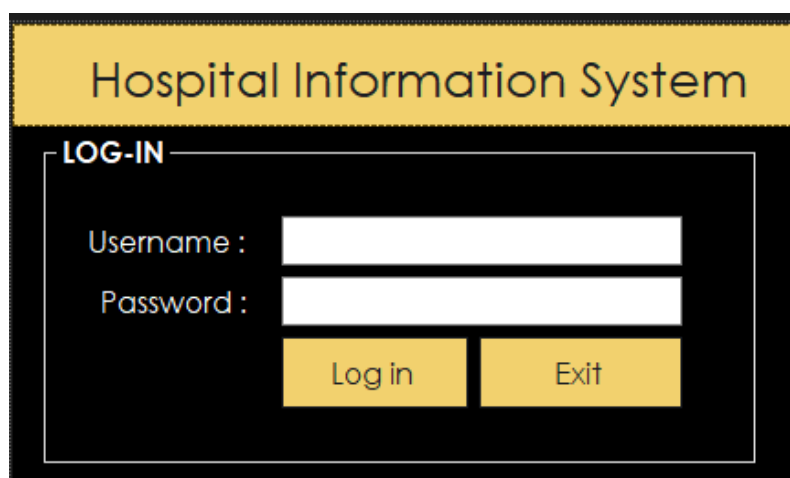


Рисунок 1.1 - Головне меню застосунку Hospital Information System

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

За допомогою даного застосунку користувач може робити наступні дії вести облік лікарів:

Додавати, редагувати та видаляти інформацію про лікарів, їх спеціалізацію, контактні дані та графік роботи (рис. 1.2).

Рисунок 1.2 – Форма по обліку лікарів Hospital Management Studio

Управляти пацієнтами: Записувати нових пацієнтів, зберігати їх особисті дані, медичну історію, призначення та результати обстежень (рис. 1.3).

Рисунок 1.3 – Форма по обліку пацієнтів Hospital Management Studio

Керувати налаштуваннями користувача: Створювати користувача з такими параметрами, як ім'я та фамілія. Можливість редагувати паролі.

Рисунок 1.4 – Форма по обліку користувачів Hospital Management Studio

Забезпечувати доступ до інформації: Забезпечувати правильний та безпечний доступ до медичної інформації для лікарів, медичного персоналу та адміністраторів.

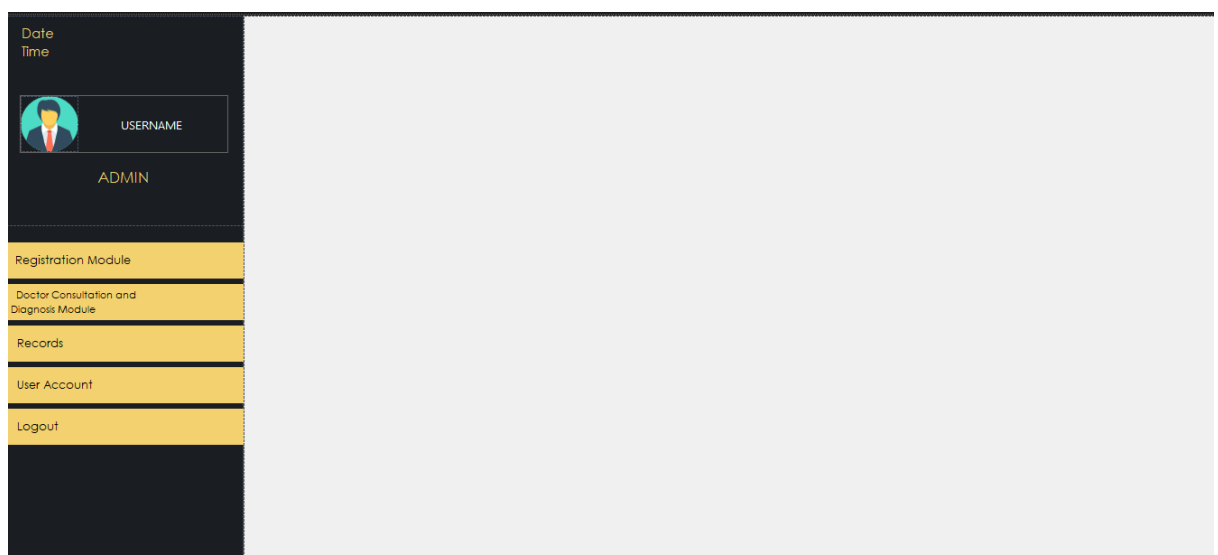


Рисунок 1.5 – Приборна панель Hospital Management Studio

1.3. Інструментальні засоби реалізації програмного продукту

Використання мови програмування C#, фреймворку WPF (Windows Presentation Foundation) та бази даних T-SQL (Transact-SQL) має численні переваги і допомагає розробникам створювати потужні та ефективні додатки. Ось кілька аргументів, які підтверджують мій вибір цих технологій [4]:

1. C# - мова програмування C# є однією з найпопулярніших мов для розробки програмного забезпечення на платформі .NET. Вона має простий синтаксис, що полегшує розробку, та широкий набір бібліотек і фреймворків, що сприяє швидкому та ефективному програмуванню [11].

2. WPF - WPF є потужним фреймворком для розробки десктопних додатків під операційну систему Windows. Він надає гнучкість та можливості для створення багатофункціональних і графічно привабливих інтерфейсів користувача. WPF дозволяє розробникам легко взаємодіяти з різноманітними елементами і візуальними компонентами [5].

3. T-SQL - T-SQL є діалектом мови SQL, що використовується для роботи з базами даних Microsoft SQL Server. Він має потужний набір операцій та функцій для маніпулювання даними, створення складних запитів та керування базою даних. T-SQL забезпечує швидку та ефективну обробку даних, що є важливим для системи управління лікарнями.

На відміну від інших найвідоміших СУБД, MS SQL Server має низку переваг. Найголовнішим і те, що він забезпечує створення інформаційних систем з архітектурою "клієнт-сервер", де він ж виконує функцію серверу баз даних. Однією з важливих властивостей Microsoft SQL Server є можливість управління цілісністю даних [6].

Можна сказати, що він відповідає вимогам, що ставляться до системам розподіленої обробки інформації. Ця СУБД підтримує: тиражування даних, паралельну обробку, створення умов та обробку великих баз даних на недорогих апаратних платформах, відрізняється простотою управління і перспективи використання(таблиця 1.1).

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
						14
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 1.1

Характеристики СУБД

Характеристика	MS SQL SERVER	MongoDb	MySQL
Модель даних	Реляційна	Документно-орієнтована	Реляційна
Адміністративне керування	Добре	Добре	Відмінно
Графічні інструменти	Відмінно	Відмінно	Відмінно
Простота обслуговування	Відмінно	Відмінно	Відмінно
Робота с декількома ЦП	Задовільно	Добре	Добре
Підключення к Web	Задовільно	Добре	Відмінно
Побудова БД	Добре	Задовільно	Добре
Мова SQL	Відмінно	Відсутня (нереляційна СУБД)	Добре
Інтеграція з іншими СУБД	Добре	Добре	Задовільно
Одночасний доступ декількох користувачів	Добре	Добре	Добре
Масштаб застосування	Підприємства малих, середніх та великих розмірів	Підприємства малих, середніх та великих розмірів	Підприємства малих та середніх розмірів

4. Інтеграція з .NET - Використання C# та WPF у поєднанні з T-SQL дозволяє розробникам легко інтегрувати логіку додатків, інтерфейси користувача та операції з базою даних. Це забезпечує єдинообразність та зручність розробки, а також забезпечує швидку обробку даних [13].

В цілому, використання C#, WPF та T-SQL у поєднанні надає розробникам можливість створювати потужні, гнучкі та ефективні додатки для управління лікарнями, що забезпечують швидку обробку даних, графічно привабливий інтерфейс та інтеграцію з базою даних [14].

Висновки до першого розділу

У цьому розділі було проведено аналіз проблематики та методів необхідних для розробки програмного продукту «Hospital Management Studio».

У першому підрозділі «Технічне завдання на розробку системи» були визначені бізнес-вимоги, вимоги користувачів, функціональні вимоги, нефункціональні вимоги, системні вимоги, бізнес-правила, антивимоги, аналіз проекту та тестування системи. Це дало чітку картину того, які вимоги та задачі повинні бути вирішені при розробці програмного продукту.

У другому підрозділі «Аналіз аналогів програмного продукту» було проведено дослідження та огляд існуючого додатка, який виконує схожі функції. Цей аналіз дозволив виявити переваги та недоліки існуючих рішень та визначити потенційні можливості для покращення та унікальності майбутнього програмного продукту.

У третьому підрозділі "Інструментальні засоби реалізації програмного продукту" були розглянуті різні технології та фреймворки для розробки додатків. Порівнявши їх, було встановлено, що використання мови програмування C#, фреймворку WPF та бази даних T-SQL має значні переваги. Ці технології забезпечують широкі можливості для розробки потужних, ефективних та зручних в управлінні додатків. Вибір C#, WPF та T-SQL є обґрунтованим з точки зору продуктивності, зручності розробки та можливості ефективної роботи з базою даних [15].

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ДОДАТКУ

2.1. Визначення варіантів використання системи

Даний підрозділ дипломної роботи розглядає основні функціональні можливості системи "Hospital Management Studio" (HMS) та описує різні сценарії використання, які задовольняють потреби користувачів у контексті управління медичним закладом.

Перша функціональна можливість системи - це перегляд поточних таблиць. Користувачі можуть переглядати різні таблиці, які містять інформацію про лікарів, їх позиції та інші важливі дані, необхідні для ефективного управління медичним закладом.

Друга функціональність полягає в діях над таблицями. Користувачі можуть здійснювати різні дії, такі як додавання, видалення або редагування записів у таблицях. Наприклад, адміністратор може додати новий медичний запис лікаря або редагувати існуючий запис з метою оновлення медичної інформації.

Третя функціональність - бекап. Система надає можливість створювати резервні копії бази даних, що забезпечує збереження інформації про пацієнтів та інші дані у випадку виникнення непередбачуваних ситуацій, таких як випадкове видалення або пошкодження даних [8].

Четверта функціональність – виведення на друк конкретної таблиці. Користувачі можуть виконувати друк конкретної таблиці, наприклад, фінансової звітності або розкладу роботи, для подальшого використання у паперовому форматі або для збереження у вигляді електронного документа.

Остання функціональність - перегляд фінансової звітності. Система HMS надає можливість перегляду фінансової звітності, що включає інформацію про донати, їх кількість, призначення та інші фінансові аспекти медичного закладу. Це дозволяє адміністраторам моніторити фінансовий стан закладу та приймати відповідні управлінські рішення. Продемонстровано приклад (рис. 2.1) , використовуючи UML [2].

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		

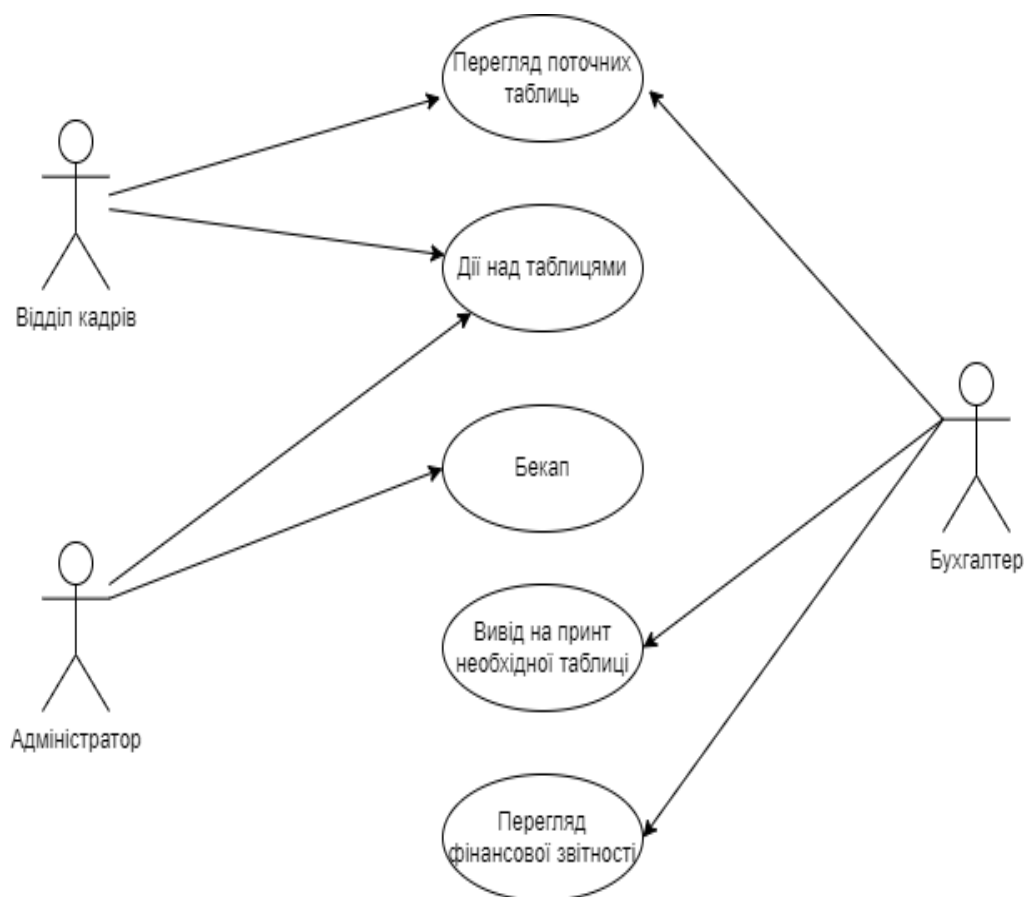


Рисунок 2.1 – Діаграма варіанту використання: взаємодія з системою

Таблиця 2.1

Опис варіанту використання «Перегляд поточних таблиць»

Назва варіанту використання	Перегляд поточних таблиць
Опис варіанту використання	Перегляд та слідкування за актуальною інформації щодо відповідного поля
Актор	Відділ кадрів
Передумови	Користувач повинен увійти в систему
Основні дії	Натиснути на відповідну таблицю
Результат	Обрана таблиця завантажена та готова для взаємодії

Таблиця 2.2

Опис варіанту використання «Оновлення запису у таблиці»

Назва варіанту використання	Оновлення таблиці
Опис варіанту використання	Оновлення запису
Актор	Адміністратор
Передумови	Користувач повинен увійти в систему
Основні дії	Натиснути на відповідну таблицю
	Натиснути на необхідний запис
	Натиснути кнопку оновити
	Обрати необхідне поле
	Змінити значення
	Натиснути кнопку оновити
Результат	У випадку введення валідного значення, запис оновиться. У випадку введення невалідного значення, система покажи помилку.

Таблиця 2.3

Опис варіанту використання «Додавання запису у таблицю»

Назва варіанту використання	Додавання запису у таблицю
Опис варіанту використання	Додавання запису
Актор	Адміністратор
Передумови	Користувач повинен увійти в систему
Основні дії	Натиснути на відповідну таблицю
	Натиснути кнопку додати
	Обрати необхідні поля
	Заповнити їх
	Натиснути кнопку Додати
Результат	У випадку введення валідного значення, запис додається. У випадку введення невалідного значення, система покажи помилку.

Таблиця 2.4

Опис варіанту використання «Видалення запису з таблиці»

Назва варіанту використання	Видалення запису з таблиці
Опис варіанту використання	Видалення запису
Актор	Адміністратор
Передумови	Користувач повинен увійти в систему
	Натиснути на відповідну таблицю
	Натиснути на необхідний запис
	Натиснути кнопку видалити
	Підвередити дію у спливаючому вікні
Результат	Запис видалиться

Таблиця 2.5

Опис варіанту використання «Пошук запису у таблиці»

Назва варіанту використання	Пошук запису у таблиці
Опис варіанту використання	Пошук запису
Актор	Відділ кадрів
Передумови	Користувач повинен увійти в систему
	Натиснути на відповідну таблицю
	Ввести необхідне значення в поле
	Натиснути кнопку шукати
Результат	Якщо запис буде знайдений, система відобразить його. Якщо запис не буде знайдений, система відобразить відповідне сповіщення.

Таблиця 2.6

Опис варіанту використання «Бекап БД»

Назва варіанту використання	Бекап БД
Опис варіанту використання	Бекап бази даних
Актор	Адміністратор
Передумови	Користувач повинен увійти в систему
	Натиснути кнопку бекап

Результат	Резервна копія успішно створиться
-----------	-----------------------------------

Таблиця 2.6

Опис варіанту використання «Друк обраної таблиці»

Назва варіанту використання	Друк обраної таблиці
Опис варіанту використання	Друк таблиці
Актор	Бухгалтер
Передумови	Користувач повинен увійти в систему
	Натиснути на відповідну таблицю
	Натиснути кнопку друк
	Обрати тип друку
Результат	Таблиця буде роздрукована у відповідному форматі

Таблиця 2.7

Опис варіанту використання «Перегляд фінансової звітності»

Назва варіанту використання	Перегляд фінансової звітності
Опис варіанту використання	Перегляд фінансової звітності
Актор	Бухгалтер
Передумови	Користувач повинен увійти в систему
	Натиснути кнопку фінансова звітність
Результат	Фінансова звітність буде відображена

2.2. Алгоритми роботи системи

У даному підрозділі описується логіка та послідовність операцій, які виконує система «Hospital Management Studio» для забезпечення своєї функціональності. Алгоритми визначають, як система взаємодіє з користувачами та виконує різні операції для успішного управління медичним закладом.

Основні алгоритми роботи системи HMS включають:

1. Перегляд поточних таблиць: Система дозволяє користувачам переглядати різні таблиці, такі як "Відпустки", "Лікарі", "Спеціалізації" та інші. Користувач може вибрати необхідну таблицю для перегляду.

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

2. Дії над таблицями: Система надає можливість виконувати різні дії над таблицями, такі як додавання нових записів, редагування існуючих записів та видалення зайвих даних. Користувач може вибрати потрібну дію та виконати її відповідно до прав доступу.

3. Бекап (резервне копіювання): Система дозволяє користувачам створювати резервні копії бази даних для забезпечення безпеки і відновлення інформації в разі потреби. Користувач може ініціювати процес створення резервної копії та зберегти її на зовнішньому пристрої або в окремому файлі.

4. Видід на принт конкретної таблиці: Система дозволяє користувачам вибрати певну таблицю та надрукувати її. Користувач може налаштувати параметри друку та отримати надруковану версію таблиці.

5. Перегляд фінансової звітності: Система надає можливість користувачам переглядати фінансову звітність, що включає інформацію про доходи.

6. Таким чином, описані алгоритми роботи системи HMS дозволяють користувачам ефективно використовувати систему для управління медичним закладом, перегляду даних, виконання різних дій над таблицями, створення резервних копій, друку та аналізу фінансової звітності.

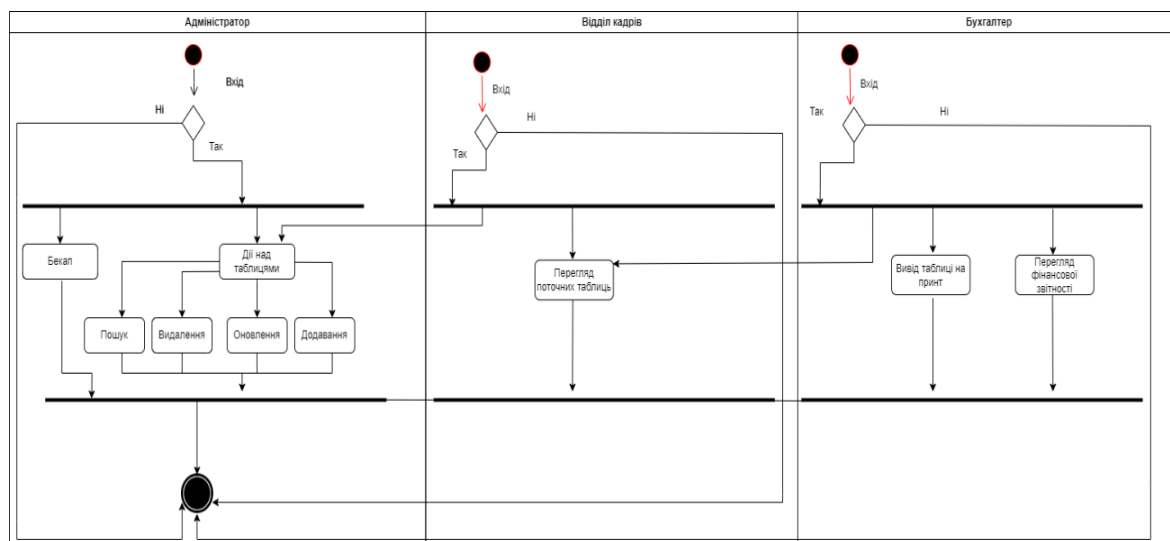


Рисунок 2.2 – Діаграма активності додатку

Спочатку виконується вхід до системи HMS (Hospital Management System), включаючи аутентифікацію та ініціалізацію змінних та налаштування параметрів системи.

Після успішного входу до системи, користувачеві надається можливість переглянути поточні таблиці. Це дозволяє користувачу отримати огляд інформації про лікарів, спеціалізації та інше.

Далі, користувач може виконувати різні дії над таблицями, такі як додавання, видалення або редагування записів. Наприклад, він може додавати нового лікаря, оновлювати інформацію про відпустки або видаляти старі записи.

Для забезпечення безпеки та запобігання втрати даних, система HMS також надає можливість створення резервних копій бази даних (бекап). Це важлива функція, яка дозволяє відновити дані в разі непередбачених проблем або помилок.

Крім того, користувач може виконати друк (видід на принт) конкретної таблиці, що дозволяє отримати фізичну копію інформації для подальшого використання або аналізу.

Таким чином, алгоритм роботи з HMS включає перегляд поточних таблиць, дії над таблицями, створення резервних копій, видід на принт конкретної таблиці та перегляд фінансової звітності. Цей алгоритм допомагає користувачам ефективно управляти медичними операціями та забезпечувати належну організацію та контроль над даними.

Також представлено діаграму класів, на якій відображено взаємозв'язок між класами (рис. 2.3). Діаграма класів є важливим інструментом для візуалізації структури об'єктно-орієнтованої системи. На діаграмі класів представлені класи системи, їх атрибути та методи, а також зв'язки між класами. Ця діаграма допомагає розуміти взаємодію між класами, їхню ієрархію, а також можливість розширення системи шляхом додавання нових класів або модифікації існуючих.

Діаграма класів також дозволяє виявляти залежності між класами, такі як асоціації, композиції або наслідування, що сприяє зрозумінню структури системи та допомагає в процесі проектування та розробки програмного забезпечення.

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

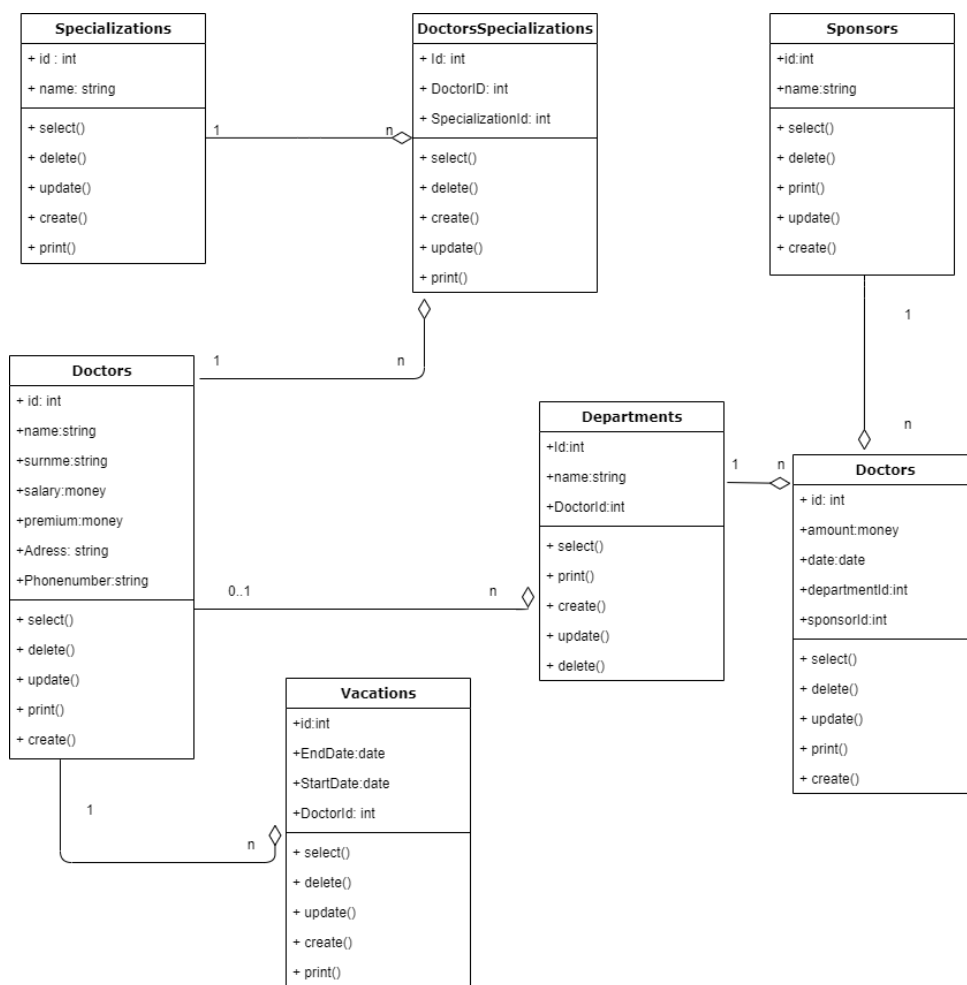


Рисунок 2.3 – Діаграма класів Hospital Management system

2.3. Структура компонентів системи

Даний підрозділ описує структуру розробленої системи та розташування різних компонентів і модулів. Структура системи складається з наступних складових частин :

1. Properties – у даній папці зберігаються різноманітні налаштування та ресурси, які використовуються в додатку. Також зберігаються налаштування додатка, такі як рядкові, числові, булеві значення та інші налаштування, які можуть бути змінюваними користувачем.
2. References – посилання на зовнішні бібліотеки та компоненти, які використовуються в проекті, а саме EntityFramework, PresentationCore, System.Drawing та інші. Цей компонент дозволяє включати залежності в проект та використовувати їх функціональність.

3. DB – зберігається файл з розширенням .edmx в WPF (Windows Presentation Foundation). У Visual Studio представляє собою модель Entity Data Model (EDM). EDM є частиною технології ADO.NET Entity Framework і використовується для описування логічної структури бази даних та її сутностей у вигляді об'єктів. Також, файл .edmx містить інформацію про таблиці бази даних, структуру зв'язків між ними, а також відображення цих таблиць на класи Entity Framework. Цей файл генерується автоматично за допомогою інструментів Visual Studio під час розробки додатка, який використовує Entity Framework.
4. Resources - у цій папці зберігаються картинки та іконки, необхідні для візуального відображення додатку.
5. App config - зберігаються налаштування підключення до бази даних (БД). Це включає рядкове підключення до БД, використовуючи певний провайдер даних.
6. App xaml - файл використовується для описування ресурсів та налаштувань, які використовуються на рівні додатка. Він містить глобальні налаштування і ресурси, які будуть доступні для всіх вікон і сторінок у вашому WPF-додатку.
7. MainWindow – файл, у якому зберігаються основні налаштування головного вікна застосунку.
8. Stats - файл, у якому зберігаються основні налаштування вікна фінансової звітності застосунку.
9. Window1 - файл, у якому зберігаються основні налаштування вікна авторизації застосунку.

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

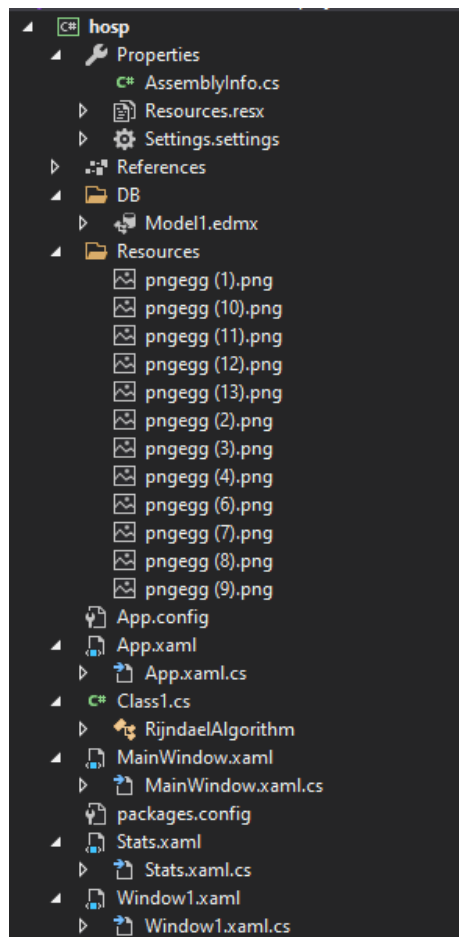


Рисунок 2.4 – Структура додатку

Використання такої структури сприяє підтримці організованого, легко зрозумілого та підтримуваного коду. Кожна складова системи відповідає за свою конкретну функціональність та має визначені обов'язки, що дозволяє розробникам швидко знаходити необхідний код та здійснювати зміни в ньому з мінімальними зусиллями.

2.4. Реалізація додатку

У цьому підрозділі буде розглянуто деталі реалізації додатку «Hospital Management Studio».

Для реалізації проекту було використано Visual Studio і MSSQL. Робота з БД здійснювалася автоматично через в Visual Studio. Реалізація була здійснена у програмі Visual Studio мовою програмування c#, а саме для роботи з БД слугувало розширення ADO.NET EDM [7].

Найбільш важлива функція в проекті – підключення до БД.

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

```
<connectionStrings><add name="hospEntities"
connectionString="metadata=res://*/DB.Model1.csdl|res://*/DB.Model1.ssdl|res://*/DB.Model1.m
sl;provider=System.Data.SqlClient;provider connection string="data
source=COMPUTER\SQLEXPRESS;initial catalog=hosp;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework";"
providerName="System.Data.EntityClient" /></connectionStrings>
```

Даний код представляє конфігурацію рядка підключення до бази даних в системі Hospital Management Studio (HMS). В розділі <connectionStrings> зазначено параметри підключення до бази даних, які використовуються для взаємодії з базою даних HMS.

Конкретний рядок підключення connectionString містить такі важливі компоненти:

metadata: вказує на розташування файлів метаданих (.csdl, .ssdl, .msl), які визначають модель даних HMS.

provider: визначає провайдер бази даних, у цьому випадку використовується провайдер System.Data.SqlClient, який дозволяє взаємодіяти з базою даних SQL Server.

provider connection string: містить параметри підключення до бази даних SQL Server, такі як розташування сервера (data source), назву бази даних (initial catalog), аутентифікацію (integrated security), дозвіл на кілька активних результатів (MultipleActiveResultSets) та інші.

providerName: вказує тип провайдера, в даному випадку System.Data.EntityClient, що використовується для Entity Framework.

Цей рядок підключення використовується системою HMS для налагодження зв'язку з базою даних та виконання операцій збереження, отримання та оновлення даних.

Приклад виводу одної таблиці та її даних з бази:

```
if (current_table == Table.Doctors)
{
var query =
from doctor in db.Doctors
select new { doctor.id, doctor.name, doctor.premium, doctor.salary, doctor.surname };
datagrid.ItemsSource = query.ToList();
}
```

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

У даному коді відбувається вибірка даних з таблиці "Doctors" і встановлення результатів в поле джерела даних ('ItemsSource') об'єкта 'datagrid'. Перш за все, перевіряється, чи змінна 'current_table' має значення "Doctors". Якщо це умова виконується, то виконується блок коду всередині фігурних дужок. У цьому блоку коду виконується запит до бази даних (в даному випадку використовується 'db', що представляє об'єкт бази даних) для вибірки даних з таблиці "Doctors". Запит виконується за допомогою LINQ-синтаксису (Language Integrated Query), де 'db.Doctors' представляє доступ до таблиці "Doctors".

Вибірка включає в себе вибір конкретних полів ('id', 'name', 'premium', 'salary', 'surname') з таблиці "Doctors". Кожен вибраний запис представляється анонімним типом об'єкта '{ doctor.id, doctor.name, doctor.premium, doctor.salary, doctor.surname }'. Результати вибірки перетворюються в список ('ToList()') і присвоюються полю 'ItemsSource' об'єкта 'datagrid', що дозволяє відобразити ці дані у відповідному елементі інтерфейсу користувача (наприклад, у ґрид-вигляді).

Приклад редагування запису по одній таблиці

```
if (current_table == Table.Doctors)
{
    int i = 0;
    foreach (var doctor in db.Doctors)
    {
        if (i == datagrid.SelectedIndex)
        {
            int columnNumber = 0;
            int rowNum = 1;
            Button buttonAdd = GetButton("Update");
            Button buttonCancel = GetButton("Cancel");
            datagrid2.Children.Clear();
            ShowAddingSection();
            find_t = doctor.id;
            DisplayTableHeader(datagrid2, new Doctors());
            InsertTextBoxIntoGrid(datagrid2, "D", rowNum, columnNumber++,
false, doctor.name);
            InsertTextBoxIntoGrid(datagrid2, "D", rowNum, columnNumber++,
false, doctor.premium.ToString());
            InsertTextBoxIntoGrid(datagrid2, "D", rowNum, columnNumber++,
false, doctor.salary.ToString());
            InsertTextBoxIntoGrid(datagrid2, "D", rowNum, columnNumber++,
false, doctor.surname);
            buttonAdd.Click += UpdateDoctor_Click;
            buttonCancel.Click += Cancel_click;
            InsertButtonIntoGrid(datagrid2, buttonAdd, rowNum,
columnNumber++);
        }
    }
}
```

		П.О. Войтко			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						28
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        InsertButtonIntoGrid(datagrid2, buttonCancel, rowNumber,
columnNumber);
        break;
    }
    i++;
}
}}
private void UpdateDoctor_Click(object sender, RoutedEventArgs e)
{
    decimal ddd = 0;
    var a1 = ((TextBox)datagrid2.Children[6]).Text.Trim();
    var a2 = ((TextBox)datagrid2.Children[7]).Text.Trim();
    var a3 = ((TextBox)datagrid2.Children[8]).Text.Trim();
    var a4 = ((TextBox)datagrid2.Children[9]).Text.Trim();
    if (a1 == "" || a2 == "" || a3 == "" || a4 == "")
    {
        MessageBox.Show("Виникла помилка! Потрібно заповнити всі поля!");
        return;
    }
    if (!decimal.TryParse(a2, out ddd) || !decimal.TryParse(a3, out ddd))
    {
        MessageBox.Show("Дані введено невірно!");
        return;
    }
    var doctor = db.Doctors.Find(find_t);
    doctor.name = a1;
    doctor.premium = Convert.ToDecimal(a2);
    doctor.salary = Convert.ToDecimal(a3);
    doctor.surname = a4;

    db.SaveChanges();
    update();
    datagrid2.Children.Clear();
    datagrid.IsEnabled = true;
    MessageBox.Show("Нові дані успішно оновлені!");
}

```

У першому блоку коду перевіряється, чи `current\_table` має значення "Doctors". Якщо це умова виконується, то виконується блок коду всередині фігурних дужок. У цьому блоку коду за допомогою циклу `foreach` проходиться по кожному об'єкту `doctor` у колекції `db.Doctors`, що представляє дані з таблиці "Doctors".

Умова `if (i == datagrid.SelectedIndex)` перевіряє, чи поточний індекс `i` у циклі відповідає вибраному рядку `datagrid.SelectedIndex` у таблиці `datagrid`. Якщо це так, то виконується блок коду всередині цієї умови. У цьому блоку коду відбувається створення елементів інтерфейсу користувача, таких як кнопки (`buttonAdd` і `buttonCancel`), та їх розміщення в `datagrid2`. Відображається секція редагування, в якій відображається інформація про лікаря з таблиці "Doctors".

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
						29
Змн.	Арк.	№ докум.	Підпис	Дата		

При кліку на кнопку "Update" (`buttonAdd`), спрацьовує обробник події `UpdateDoctor\_Click`. У цьому обробнику отримуються значення текстових полів (`TextBox`) з `datagrid2` і проводяться перевірки їх валідності та правильності даних. Якщо дані не валідні, виводиться повідомлення про помилку. В іншому випадку, оновлюються дані про лікаря в базі даних (`db.Doctors.Find(find\_t)`) з введеними значеннями. Зміни зберігаються в базі даних (`db.SaveChanges()`) і викликається функція `update()` для оновлення відображення даних. Потім елементи в `datagrid2` очищаються, `datagrid` стає доступним для редагування, і виводиться повідомлення про успішне оновлення даних.

Приклад додавання запису по одній таблиці

```
int columnNumber = 0;
int rowNumber = 1;
Button buttonAdd = GetButton("Add");
Button buttonCancel = GetButton("Cancel");

datagrid2.Children.Clear();
ShowAddingSection();

if (current_table == Table.Doctors)
{
    DisplayTableHeader(datagrid2, new Doctors());
    InsertTextBoxIntoGrid(datagrid2, "doc_name", rowNumber, columnNumber++,
false);
    InsertTextBoxIntoGrid(datagrid2, "doc_premium", rowNumber, columnNumber++,
false);
    InsertTextBoxIntoGrid(datagrid2, "doc_salary", rowNumber, columnNumber++,
false);
    InsertTextBoxIntoGrid(datagrid2, "doc_surname", rowNumber, columnNumber++,
false);
    buttonAdd.Click += AddDoctor_Click;
}
private void AddDoctor_Click(object sender, RoutedEventArgs e)
{
    decimal ddd = 0;
    var t2 = ((TextBox)datagrid2.Children[6]).Text;
    var t3 = ((TextBox)datagrid2.Children[7]).Text;
    var t4 = ((TextBox)datagrid2.Children[8]).Text;
    var t5 = ((TextBox)datagrid2.Children[9]).Text;
    if(t2==" " || t3 == " " || t4 == " " || t5 == " ")
    {
        MessageBox.Show("Виникла помилка!Потрібно заповнити всі поля!");!!);
        return;
    }
    if(!decimal.TryParse(t3,out ddd)|| !decimal.TryParse(t4, out ddd))
    {
        MessageBox.Show("Дані введені невалідно!");
        return;
    }
    var doctor = new Doctors()
    {
```

		П.О. Войтко			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        name = t2,
        premium = Convert.ToDecimal(t3),
        salary = Convert.ToDecimal(t4),
        surname = t5
    };

    db.Doctors.Add(doctor);
    db.SaveChanges();
    update();
    datagrid.IsEnabled = true;
    datagrid2.Children.Clear();
    MessageBox.Show("Нові дані успішно оновлені!");
}

```

У першому блоку коду встановлюються значення змінних `columnName` і `rowNumber`, а також створюються кнопки "Add" і "Cancel". `datagrid2.Children.Clear()` очищає вміст `datagrid2`, а `ShowAddingSection()` відображає секцію для додавання. Умова `if (current_table == Table.Doctors)` перевіряє, чи поточна таблиця є "Doctors". Якщо це так, то виконується блок коду всередині цієї умови. У цьому блоку коду відбувається відображення заголовку таблиці "Doctors" в `datagrid2` за допомогою функції `DisplayTableHeader()`. Далі за допомогою функції `InsertTextBoxIntoGrid()` вставляються текстові поля для введення даних про лікаря (ім'я, премія, зарплата, прізвище). Також оброблюється подія `buttonAdd.Click` за допомогою функції `AddDoctor_Click`.

У функції `AddDoctor_Click` перевіряється валідність та коректність введених даних. Якщо хоча б одне з полів є порожнім, виводиться повідомлення про помилку. Якщо дані не валідні (неможливо перетворити на числовий формат), також виводиться повідомлення про помилку. В іншому випадку, створюється новий об'єкт `Doctors` з введеними значеннями, додається до бази даних (`db.Doctors.Add(doctor)`) і зберігаються зміни (`db.SaveChanges()`). Потім викликається функція `update()` для оновлення відображення даних, `datagrid` стає доступним для редагування, `datagrid2` очищається, і виводиться повідомлення про успішне додавання нових даних.

Приклад видалення запису по одній таблиці

```

if (current_table == Table.Doctors)
{
    var result = MessageBox.Show("Ви впевнені, що бажаєте видалити це??",
    "Deleting Data",
    MessageBoxButton.YesNo);
}

```

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
						31
Змн.	Арк.	№ докум.	Підпис	Дата		

```

if (result == DialogResult.Yes)
{
    int i = 0;
    foreach (var doctor in db.Doctors)
    {
        if (i == datagrid.SelectedIndex)
        {
            if(!db.DoctorsSpecializations.Where(d=>d.id==doctor.id).Any())
            {
                db.Doctors.Remove(doctor);
                MessageBox.Show("Лікар успішно видалений!!!");
            }
            else
            {
                MessageBox.Show("Помилка. Цей лікар пов'язаний з іншою
таблицею!");
            }
            break;
        }
        i++;
    }
    db.SaveChanges();
    update();}}

```

Тут ідображається діалогове вікно з питанням користувача " Ви впевнені, що бажаєте видалити це??" за допомогою MessageBox.Show(). Параметр MessageBoxButtons.YesNo вказує, що у вікні мають бути кнопки "Так" і "Ні". Після того як користувач обрав відповідь, результат зберігається в змінну result. Якщо результат дорівнює DialogResult.Yes, тобто користувач підтвердив, що бажає видалити дані, виконується блок коду всередині цієї умови.

У цьому блоку коду виконується ітерація через кожен об'єкт doctor в таблиці "Doctors" (db.Doctors). Порівнюється індекс об'єкта doctor з індексом, який вибрав користувач у datagrid (datagrid.SelectedIndex).

Якщо індекси співпадають, перевіряється, чи існують зв'язки між лікарем і іншими таблицями за допомогою db.DoctorsSpecializations.Where(d => d.id == doctor.id).Any(). Якщо не існує жодного зв'язку, то об'єкт doctor видаляється з бази даних (db.Doctors.Remove(doctor)) і виводиться повідомлення "Doctor was delete!!!" за допомогою MessageBox.Show(). Якщо існують зв'язки, виводиться повідомлення "This doctor is connect with other table!".

Після цього зміни зберігаються до бази даних (db.SaveChanges()) і виконується функція update() для оновлення відображення даних.

Приклад пошуку запису по одній таблиці

```

if (current_table == Table.Doctors)
{
    var query =

```

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
						32
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        from doctor in db.Doctors
        where doctor.name == text_find.Text
        select new { doctor.id, doctor.name, doctor.premium, doctor.salary,
doctor.surname };
    if (query.ToList().Count!=0)
    {

        datagrid.ItemsSource = query.ToList();
    }
    else
    {
        MessageBox.Show("Нічого не знайдено!"); } }

```

У даному коді перевіряється, чи поточна таблиця є таблицею лікарів (Doctors). Якщо це так, то виконується запит до бази даних для вибірки лікарів, які мають вказане значення у полі "name" (введене користувачем значення з текстового поля text_find). Результат запиту представляється у вигляді анонімного об'єкта з деякими властивостями лікаря (id, name, premium, salary, surname).

Якщо результат запиту містить хоча б один запис (Count != 0), то виводиться цей результат у відображувану таблицю (datagrid). Якщо результат порожній, тобто не знайдено відповідних лікарів, то виводиться повідомлення "Нічого не знайдено!" у вікні сповіщень (MessageBox).

Адже додаток має бути повністю автономним, дуже велику важливість має підтримка збереження бази даних та відновлення збереженої копії, адже користувачу може знадобитися перенести дані на інший пристрій.

Лістинг коду

```

try
{
    string dbname = db.Database.Connection.Database;
    string sqlCommand = @"
BACKUP DATABASE hosp
TO DISK = 'C:\Program Files\Microsoft SQL
Server\MSSQL15.SQLEXPRESS\MSSQL\Backup\hosp_FullDbBkup.bak'
WITH INIT, NAME = 'Hospital Full Db backup',
DESCRIPTION = 'Hospital Full Database Backup'";

    db.Database.ExecuteSqlCommand(System.Data.Entity.TransactionalBehavior.DoNotEnsureTransaction,
    string.Format(sqlCommand, dbname, ""));
}
catch
{
    MessageBox.Show("БЕКАП БД ПРОЙШОВ НЕВДАЛО! СПРОБУЙ ЩЕ РАЗ");
    return;
}
MessageBox.Show("БЕКАП БД ПРОЙШОВ УСПІШНО!");

```

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
						33
Змн.	Арк.	№ докум.	Підпис	Дата		

Код починається з отримання назви поточної бази даних, до якої підключений об'єкт db. Потім формується SQL-запит, який містить команду резервного копіювання бази даних. Копія бази даних зберігається на диску за вказаним шляхом та назвою файлу (у даному випадку 'C:\Program Files\MicrosoftSQLServer\MSSQL15.SQLEXPRESS\MSSQL\Backup\hosp_FullDb Bkup.bak'). Далі, за допомогою методу ExecuteSqlCommand об'єкту db.Database виконується SQL-запит для створення резервної копії бази даних.

Якщо створення резервної копії пройшло успішно, виводиться повідомлення "БЕКАП БД ПРОЙШОВ УСПІШНО!" у вікні сповіщень (MessageBox). У випадку помилки під час створення копії, виводиться повідомлення "БЕКАП БД ПРОЙШОВ НЕВДАЛО! СПРОБУЙ ШЕ РАЗ" у вікні сповіщень (MessageBox).

Тепер перейдемо до теми вибору СУБД. На найвищому рівні абстракції це, звісно, буде клієнт-серверна архітектура, проте потрібно взяти до уваги, що один сервер має працювати і з веб-додатком. У MSSQL цього можна досягти розподілом шляхів за допомогою роутерів, що буде оброблювати запити для додатку і в залежності від вимог або працюватиме з базою даних, або буде відображати необхідні сторінки. У процесі формування відповіді на запит обробник запитів звертатиметься до бази даних, проте, беручи до уваги велику кількість даних, потрібних для поліклініки, написання звичайних SQL-запитів буде занадто ускладнене та складно модифіковане, тому було прийнято рішення використовувати T-SQL, котрий спростить отримання даних з БД та дозволить перетворювати сутності бази даних в об'єкти, якими зручно оперувати за допомогою C# [1].

Проектування бази даних відбувалося у SQL SERVER. Після прототипування ми прийшли до структури бази даних, що складається з декількох основних частин. База даних лікарня містить інформацію про лікарів і пожертвування.

Лікарі, що працюють в лікарні представлені у вигляді таблиці лікарі (Doctors), в якій зібрана основна інформація, така як: ім'я, прізвище, а також

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
						34
Змн.	Арк.	№ докум.	Підпис	Дата		

зарплата. Спеціалізації лікарів представлені у вигляді таблиць спеціалізації (Specializations) і лікарі і спеціалізації (DoctorsSpecializations). Також база даних містить інформацію про відпустки лікарів, що розташовується в таблиці відпустки (Vacations). Дані відносяться до спонсорів і пожертвам зберігаються в таблицях спонсори (Sponsors) і пожертвування (Donations) відповідно.

Також в базі даних міститься інформація про палатах лікарні в таблиці палати (Wards), які тісно пов'язані з відділеннями, в яких вони розташовуються, та які представлені таблицею відділення (Departments).

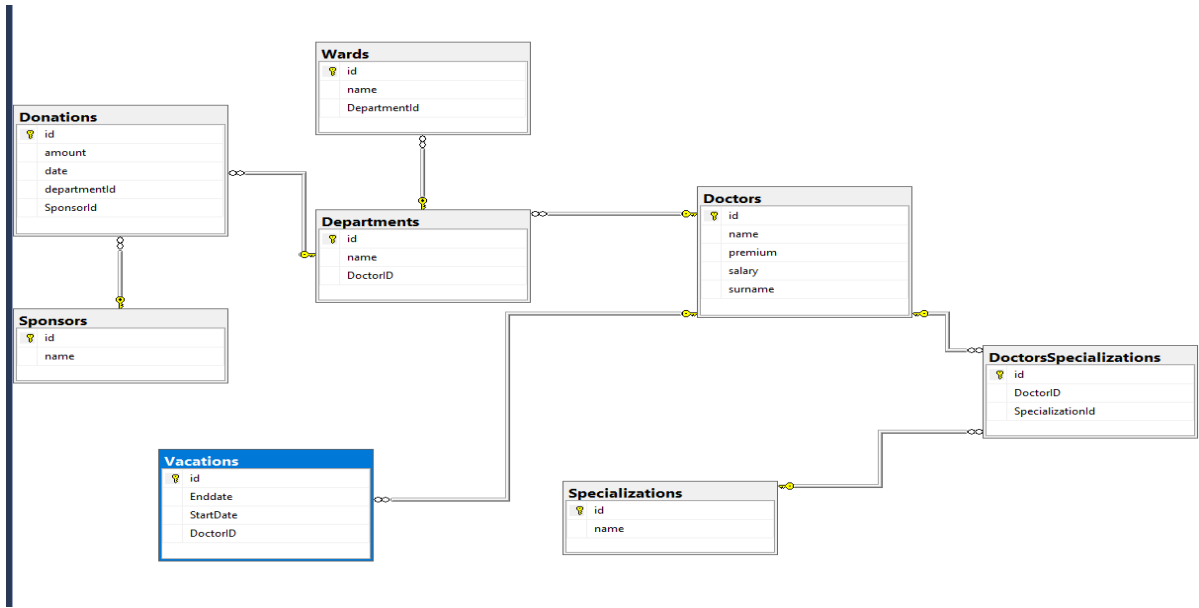


Рисунок 2.5 – Діаграма бази даних

Опис кожної таблиці, її полів та їх призначення.

Doctors	
id	Ключ
name	
premium	
salary	
surname	

Рисунок 2.6 – Таблица doctors

Таблица 2.8

Таблица doctors

Поле	Ключ	Тип	Призначення
id	РК	int	Ідентифікатор
name	-	varchar(255)	Ім'я

premium	-	money	Надбавка
salary	-	money	Зарплата
surname	-	varchar(255)	Фамілія

Specializations	
 id	
name	

Рисунок 2.7 – Таблиця specializations

Таблиця 2.9

Таблиця specializations

Поле	Ключ	Тип	Призначення
id	РК	int	Ідентифікатор
name	-	varchar(50)	Назва

DoctorsSpecializations	
 id	
DoctorID	
SpecializationId	

Рисунок 2.8 – Таблиця DoctorsSpecializations

Таблиця 2.10

Таблиця DoctorsSpecializations

Поле	Ключ	Тип	Призначення
id	РК	int	Ідентифікатор
DoctorID	FK	int (50)	Ідентифікатор лікаря
SpecializationId	FK	int	Ідентифікатор спеціалізації

Vacations	
 id	
Enddate	
StartDate	
DoctorID	

Рисунок 2.9 – Таблиця Vacations

Таблиця Vacations

Поле	Ключ	Тип	Призначення
id	PK	int	Ідентифікатор
Enddate	-	date	Назва
StartDate	-	date	Опис
DoctorID	FK	int	Ідентифікатор лікаря

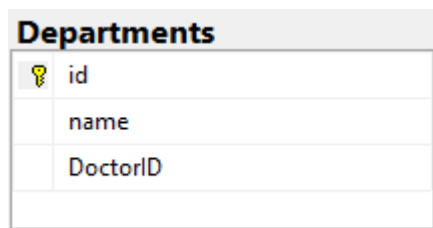


Рисунок 2.10 – Таблиця Departments

Таблиця Departments

Поле	Ключ	Тип	Призначення
id	PK	int	Ідентифікатор
name	-	varchar(50)	Назва
DoctorID	FK	int	Ідентифікатор лікаря



Рисунок 2.11 – Таблиця Wards

Таблиця Wards

Поле	Ключ	Тип	Призначення
id	PK	int	Ідентифікатор
name	-	varchar(50)	Назва
DepartmentId	FK	int	Ідентифікатор відділення

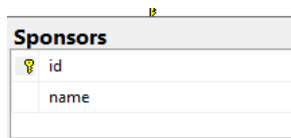


Рисунок 2.12 – Таблица Sponsors

Таблица 2.14

Поле	Ключ	Тип	Призначення
id	PK	int	Ідентифікатор
name	-	varchar(50)	Назва

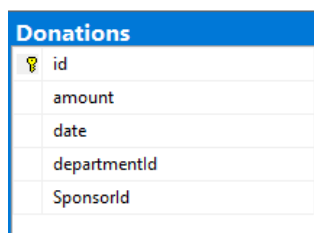


Рисунок 2.13 – Таблица Donations

Таблица 2.15

Поле	Ключ	Тип	Призначення
id	PK	int	Ідентифікатор
amount	-	varchar(50)	Назва
date	-	date	Дата пожертвування
DepartmentId	FK	int	Ідентифікатор відділення
SponsorId	FK	int	Ідентифікатор спонсора

Висновки до другого розділу

У другому розділі «Проектування та реалізація додатку» було проведено розгляд структури системи, розроблено механізми навігації між різними компонентами та успішно реалізовано додаток.

У підрозділі «Структура системи» була представлена загальна структура системи, що включає файли з різними налаштуваннями та ресурсами

У підрозділі «Реалізація додатку» було продемонстровано приклади реалізації функцій та їх роз'яснення.

Завдяки правильно розробленій структурі системи та використанню відповідних інструментів, таких як навігаційна бібліотека WPF, було забезпечено зручний інтерфейс взаємодії системи з БД. Це сприяє покращенню користувацького досвіду і забезпечує легку навігацію між функціональними складовими додатку.

РОЗДІЛ 3. ПОРЯДОК РОБОТИ ТА ТЕСТУВАННЯ СИСТЕМИ

3.1. Архітектура додатку та розгортання додатку

Архітектура додатку Hospital Management Studio використовує структуру файлів WPF (Windows Presentation Foundation) та базу даних, організовану засобами T-SQL [16]. Давайте розглянемо структуру файлів WPF та налаштування бази даних:

1. Properties (Властивості): У цій папці знаходяться різноманітні налаштування та ресурси, які використовуються в додатку Hospital Management Studio. Тут можна зберігати налаштування додатка, такі як рядкові, числові, булеві значення та інші налаштування, які можуть бути змінюваними користувачем.
2. References (Посилання): В цій папці знаходяться посилання на зовнішні бібліотеки та компоненти, які використовуються в проєкті Hospital Management Studio. Наприклад, використовується бібліотека EntityFramework для забезпечення зв'язку з базою даних, PresentationCore для візуального відображення елементів у додатку, System.Drawing для маніпуляції зображеннями та інші.
3. Resources (Ресурси): У цій папці знаходяться різноманітні ресурси, такі як картинки та іконки, які використовуються для візуального відображення додатку Hospital Management Studio. Ці ресурси можуть бути використані для прикраси інтерфейсу користувача та поліпшення його зовнішнього вигляду.
4. DB (База даних): В цій папці знаходиться файл підключення до бази даних та налаштування, пов'язані з базою даних Hospital Management Studio. Наприклад, може бути використаний файл з розширенням .edmx в WPF, який представляє собою модель Entity Data Model (EDM). Цей файл містить інформацію про структуру бази даних, таблиці, зв'язки між ними та відображення цих таблиць на класи Entity Framework.

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ПЗ	Арк.
						39
Змн.	Арк.	№ докум.	Підпис	Дата		

5. T-SQL (База даних): Для розгортання додатку Hospital Management Studio використовується база даних, яка організована за допомогою мови запитів T-SQL (Transact-SQL). Запити T-SQL використовуються для створення таблиць, виконання запитів до бази даних, оновлення даних та інших операцій з базою даних, що вимагаються для роботи додатку Hospital Management Studio.

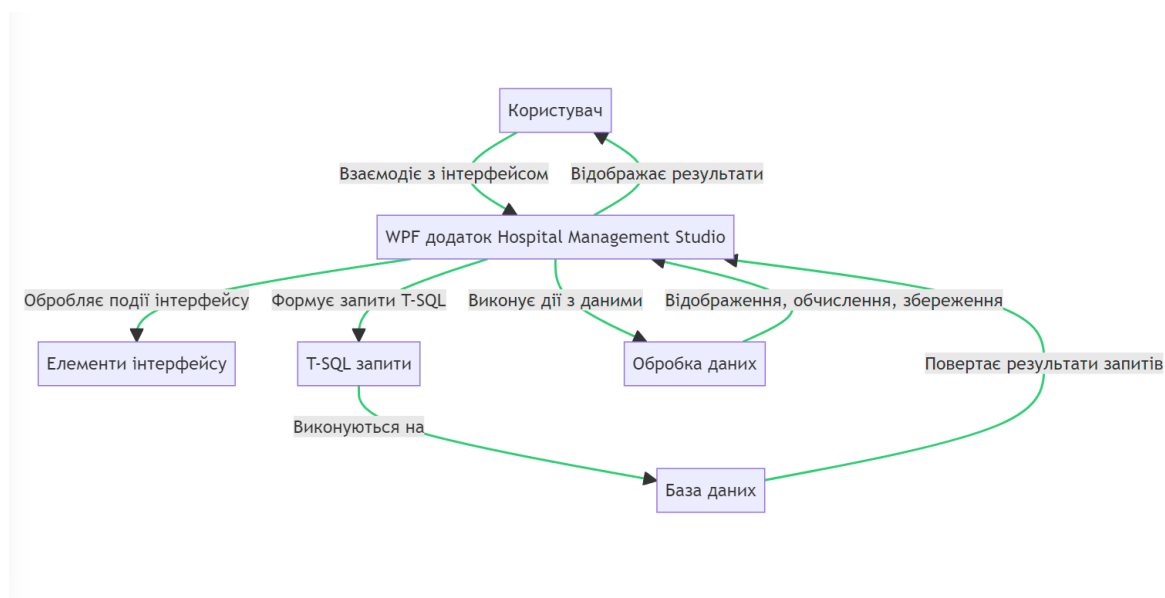


Рисунок 3.1 – Схема роботи T-SQL + WPF

1. Користувач взаємодіє з інтерфейсом WPF додатку Hospital Management Studio.
2. WPF додаток містить різні вікна та сторінки, які відображаються користувачеві.
3. При взаємодії користувача з елементами інтерфейсу (наприклад, кнопки, тексти, форми), WPF додаток обробляє ці події та виконує відповідні дії.
4. Для отримання, збереження та оновлення даних, WPF додаток взаємодіє з базою даних.
5. Запити до бази даних формуються з використанням мови запитів T-SQL.
6. Запити T-SQL виконуються на базі даних, що забезпечує доступ до даних та виконання різних операцій, таких як створення таблиць, вставка, оновлення та видалення даних.
7. Результати запитів T-SQL повертаються назад до WPF додатку для відображення або подальшої обробки.

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

8. WPF додаток виконує відповідні дії з отриманими даними, такі як відображення на інтерфейсі, обчислення, збереження тощо.
9. Процес взаємодії користувача з WPF додатком та базою даних T-SQL повторюється залежно від дій користувача

Таким чином, файлова структура WPF та база даних T-SQL взаємодіють між собою, забезпечуючи функціональність та ефективність Hospital Management Studio.

3.2. Інтерфейс та порядок роботи з системою

У цьому підрозділі буде розглянуто інтерфейс додатку «Hospital Management Studio» і описано порядок роботи з системою.

Після запуску програми з'являється форма (рис. 3.2), де у користувача є можливість пройти авторизацію за допомогою своїх кредів.

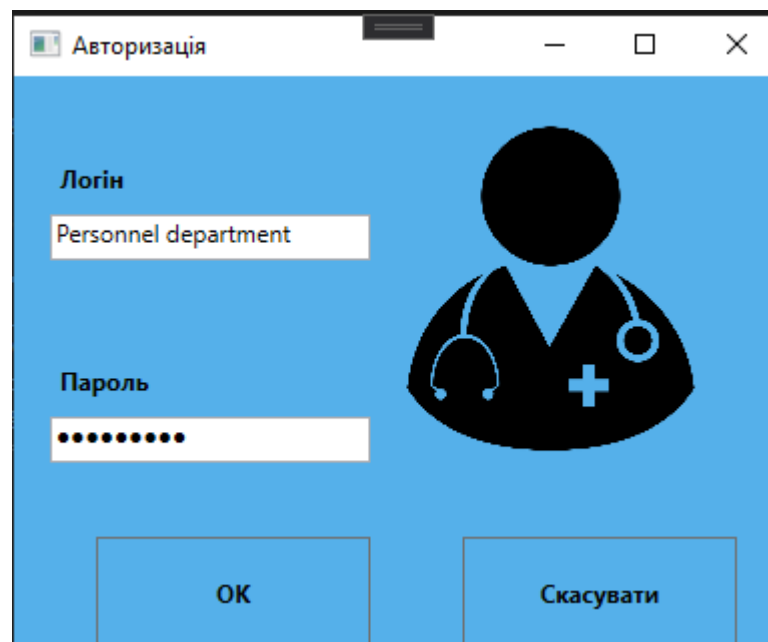


Рисунок 3.2 – Вікно авторизації системи

Після успішної авторизації система відображає наступну форму (рис. 3.3), де є меню для вибору таблиці, оновленні, пошуку, редагування та видалення. Також, в залежності від прав користувача, система буде відображати певний перелік функцій, а саме – друк, фінансовий звіт, повернення до авторизації.

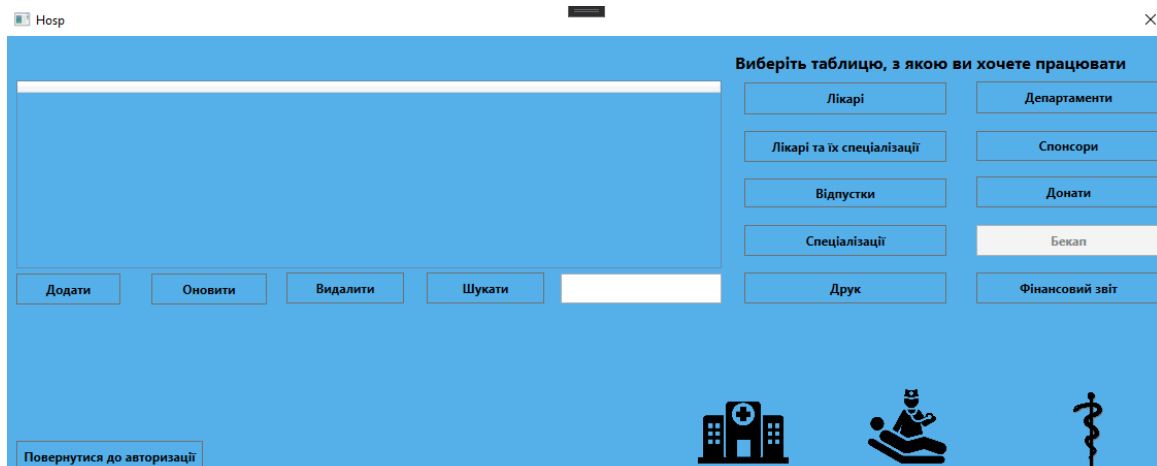


Рисунок 3.3 – Головне вікно системи

Після того як користувач вибере таблицю, система повинна підвантажити дані з таблиці (рис. 3.4), з якими він може робити дії відносно його прав.

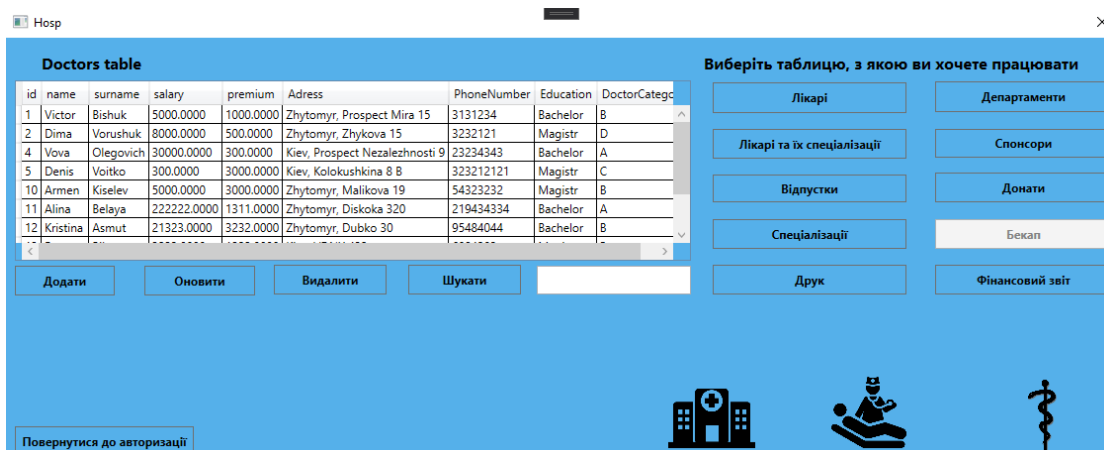


Рисунок 3.4 – Вікно таблиці лікарі

Користувач, в залежності від його прав, може обирати таблицю для перегляду, оновлення, пошуку.

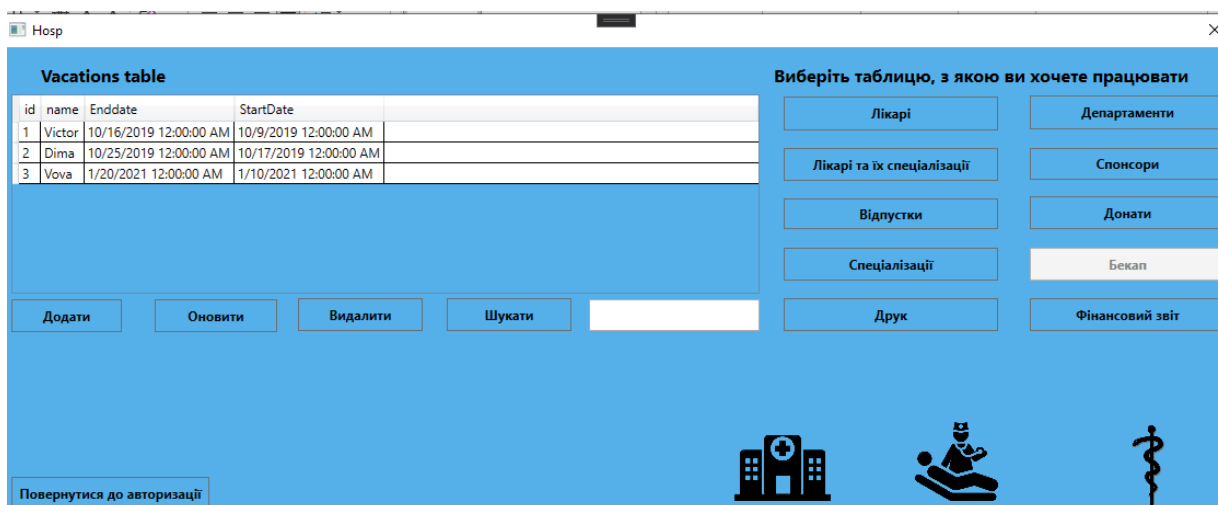


Рисунок 3.4 – Вікно таблиці відпустки

Для додавання запису користувачу потрібно вибрати таблицю, в яку він захоче додати запис. Далі користувач повинен натиснути кнопку додати. Система повинна підвантажити поля (рис. 3.5), які йому необхідно заповнити. Також з'являться дві кнопки(додати чи відмінити додавання).

Рисунок 3.5 – Вікно з можливістю додати запис

Рисунок 3.6 – Вікно додавання запису

При успішному додаванні система відобразить спливаюче вікно.

Для оновлення запису користувачу потрібно вибрати таблицю, в якій він оновити запис. Далі користувач повинен обрати запис та натиснути кнопку оновити. Система повинна підвантажити поля (рис. 3.7), які йому необхідно заповнити. Також з'являться дві кнопки(додати чи відмінити додавання).

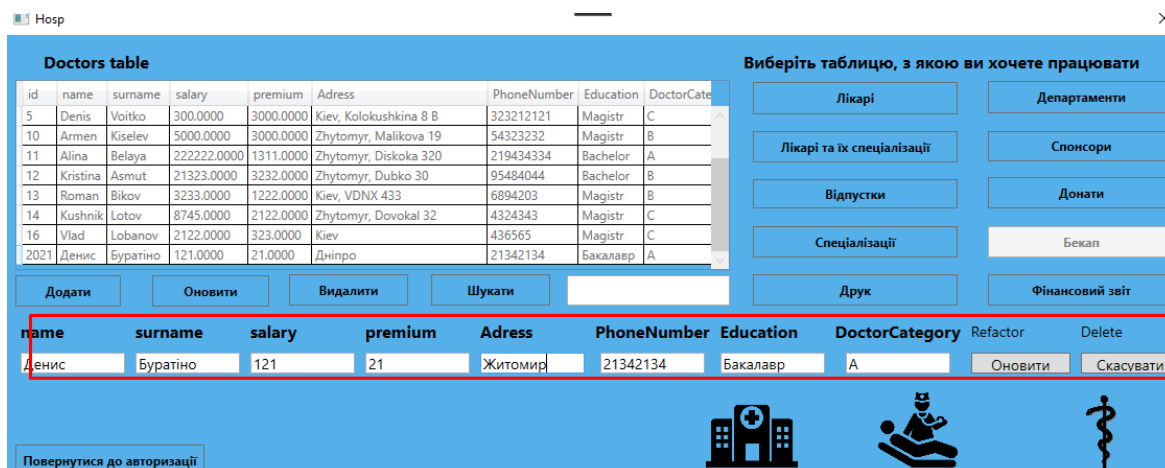


Рисунок 3.7 – Приклад оновлення даних

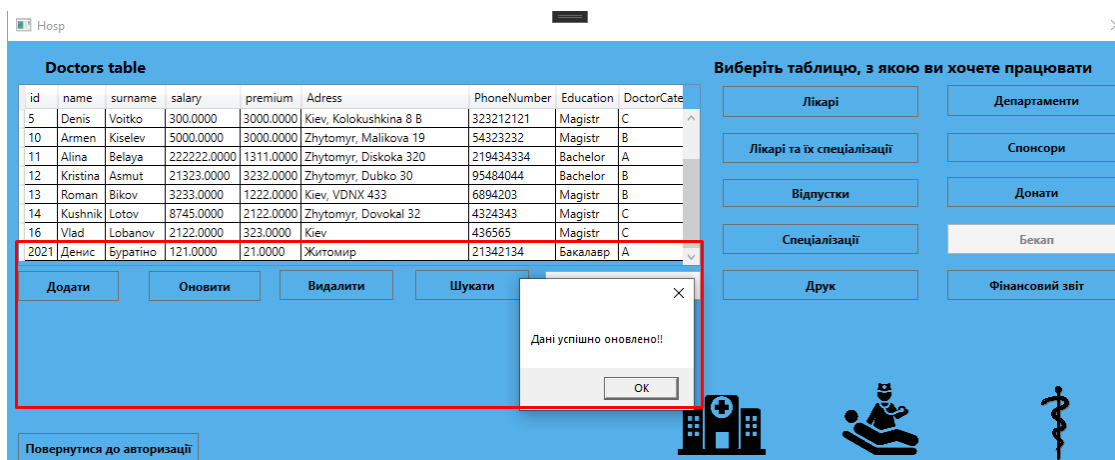


Рисунок 3.8 – Вікно оновлення запису

Якщо у користувача з'явиться потреба для видалення, йому потрібно буде обрати запис. Після чого він зможе натиснути кнопку видалення. Після кліку на кнопку видалення, система відобразить спливаюче вікно з підтвердженням (рис. 3.9).

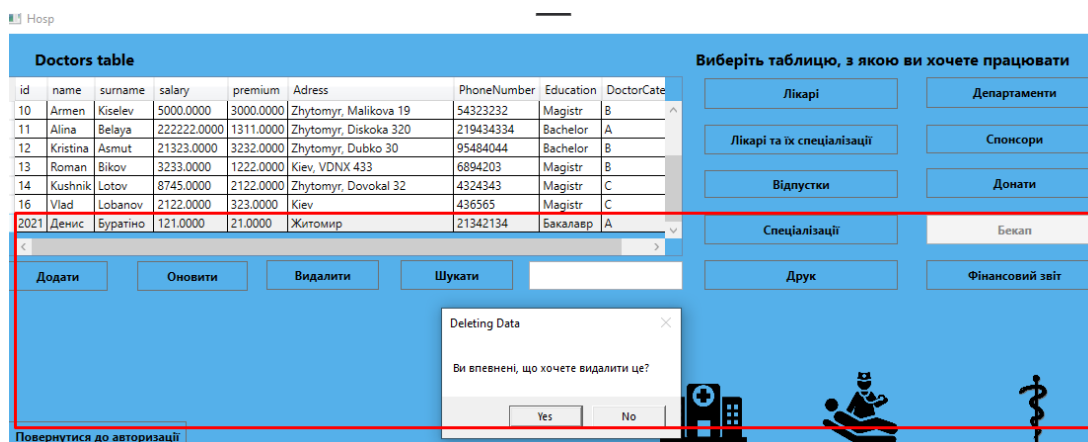


Рисунок 3.9 – Приклад видалення даних

Після видалення, система покаже спливаюче вікно з підтвердженням видалення. Видалений запис більше не буде відображатися в системі.

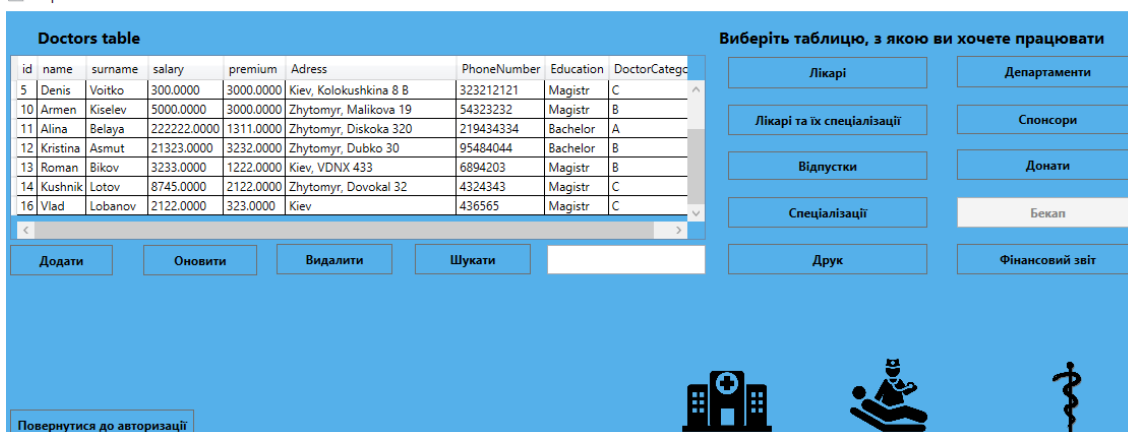


Рисунок 3.10 – Приклад видалення даних

Для пошуку користувачу необхідно заповнити текстове поле та натиснути кнопку шукати (рис. 3.11). Пошук буде відбуватися в рамках обраної таблиці.

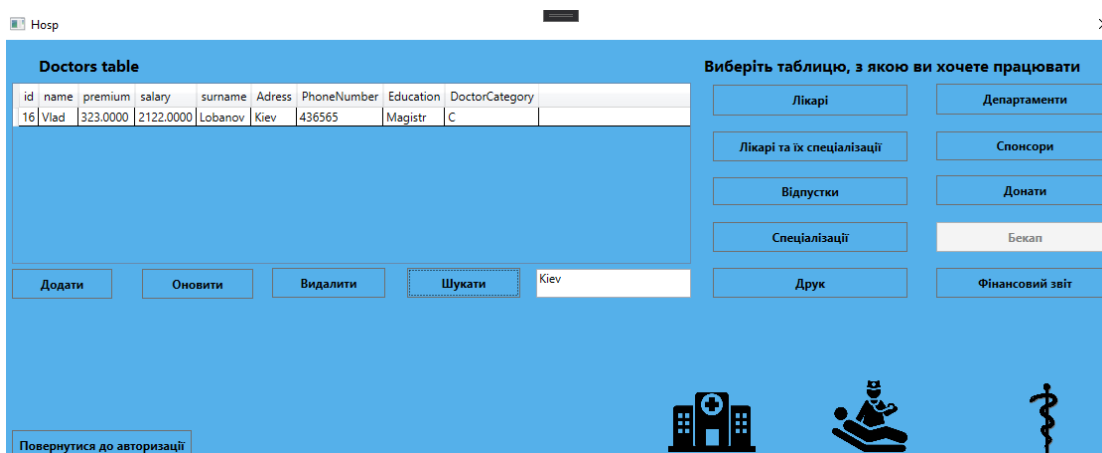


Рисунок 3.11 – Приклад пошуку даних

Адже додаток має бути повністю автономним, дуже велику важливість має підтримка збереження бази даних та відновлення збереженої копії, адже користувачу може знадобитися перенести дані на інший пристрій.

Для резервного копіювання користувачу потрібно натиснути кнопку бекап. Після натиснення кнопки відбудеться повна резервна копія та система ввідобразить спливаюче вікно (рис. 3.12).

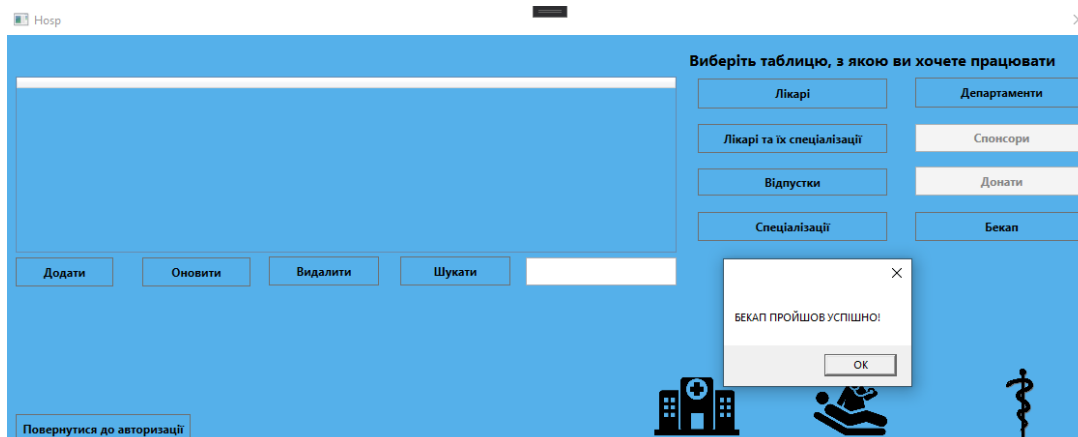


Рисунок 3.11 – Спливаюче вікно при успішному бекапі

Отже, інтерфейс має всі необхідні елементи. Він достатньо зрозумілий та не має надмірного навантаження на користувача.

3.3. Тестування роботи додатку

Тестування є невід'ємною частиною процесу розробки додатку [9]. У проекті було активно використано мануальне тестування для перевірки різних аспектів додатку. Воно відіграє важливу роль у перевірці його якості та функціональності. Для тестування програмного забезпечення була використана техніка "Еквівалентний поділ" [10]. Ця техніка дозволила ефективно визначити набір вхідних даних, які потрібно перевірити для виявлення помилок та несправностей в програмі.

Під час застосування техніки "Еквівалентний поділ" було розбито множину можливих вхідних даних на класи еквівалентності. Кожен клас еквівалентності представляє групу вхідних даних, які можна вважати еквівалентними з точки зору поведінки програми. Таким чином, замість тестування кожного окремого вхідного значення, можна перевірити одне представницьке значення з кожного класу еквівалентності.

Застосування техніки "Еквівалентний поділ" дозволило ефективно скоротити кількість тестових сценаріїв, що потребують виконання, зберігаючи при цьому високу покриття вхідних даних.

Тепер перейдемо до розгляду ситуацій.

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
						46
Змн.	Арк.	№ докум.	Підпис	Дата		

Якщо користувач під час створення нового запису, не заповнить всі обов'язкові поля, то система відобразить спливаюче вікно (рис. 3.12). з помилкою.

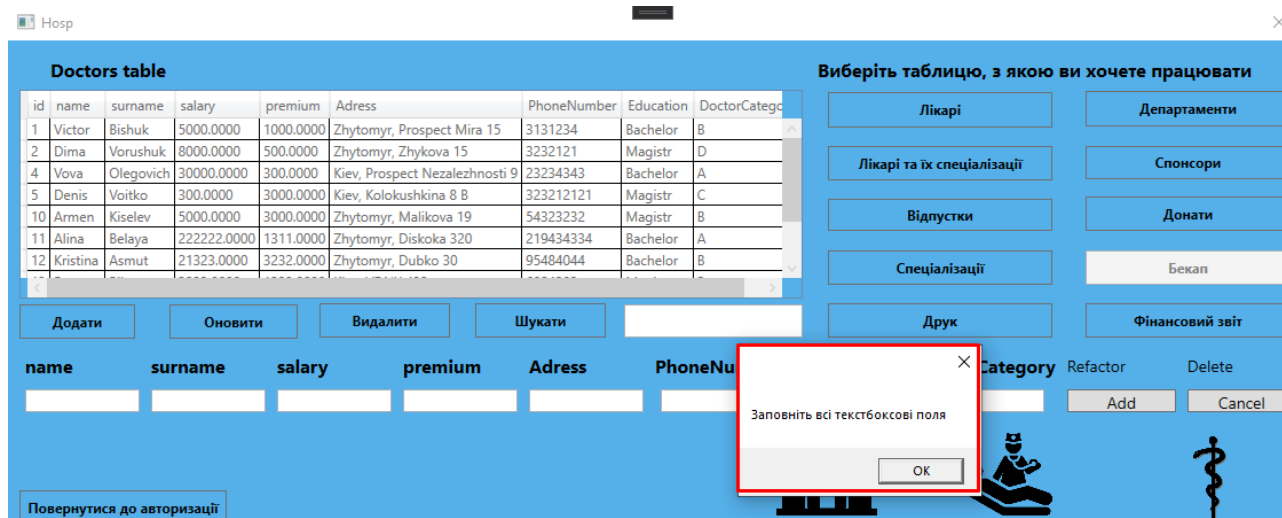


Рисунок 3.12 – Спливаюче вікно при додаванні запису

Якщо користувач не правильно заповнить поле до типу даних, то також система відобразить спливаюче вікно (рис. 3.13) з помилкою.

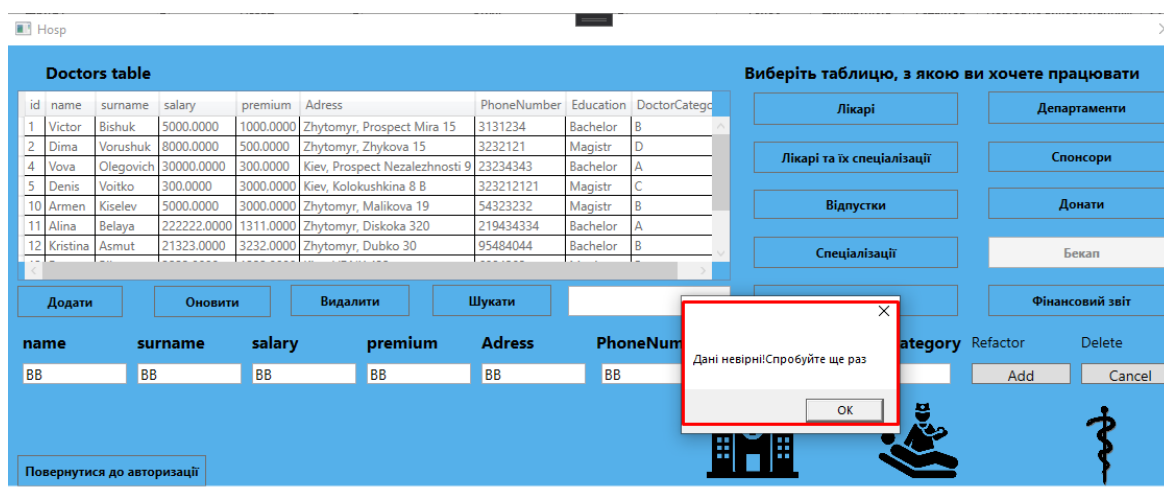


Рисунок 3.13 – Спливаюче вікно при додаванні запису

Якщо при видаленні користувач не обере конкретний запис, то система відобразить спливаюче вікно (рис. 3.14) з помилкою.

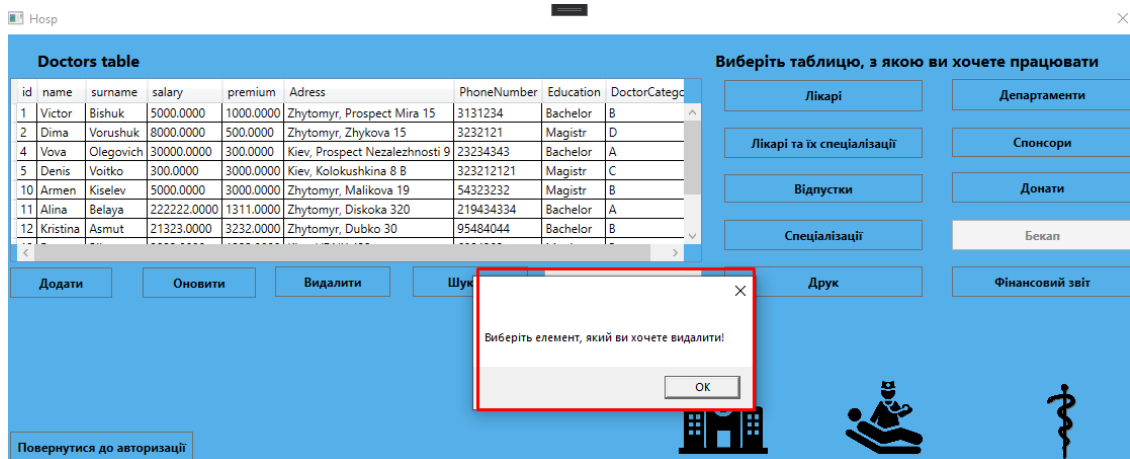


Рисунок 3.14 – Спливаюче вікно при видаленні запису

Схожа ситуація і при оновленні даних в таблиці, так і при пошуку (рис. 3.15).

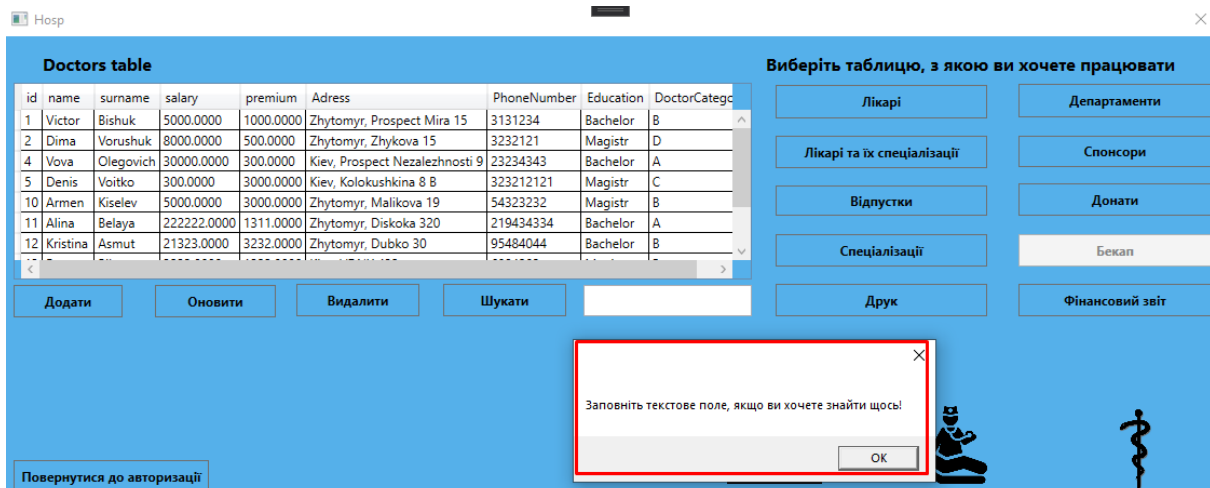


Рисунок 3.15 – Спливаюче вікно при пошуку запису

Якщо користувач введе невірний логін або пароль, система також відобразить спливаюче вікно (рис. 3.16) з помилкою.

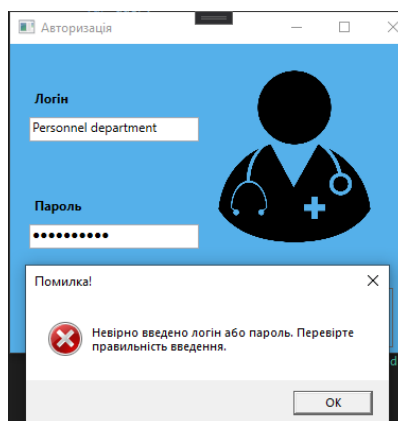


Рисунок 3.16 – Спливаюче вікно невдалій авторизації

Тестування роботи додатку виявилось надзвичайно корисним і результативним процесом. Воно допомогло виявити та виправити помилки та недоліки програми на ранніх етапах розробки, що сприяло поліпшенню її якості та надійності. Крім того, тестування позитивно позначилося на користувацькому досвіді, забезпечивши безперебійну та задовільну роботу додатку. Усі ці фактори сприяли успішному впровадженню програми та досягненню її мети.

Висновки до третього розділу

Архітектура та розгортання додатку, інтерфейс та порядок роботи з системою, а також тестування роботи додатку грають критичну роль у розробці успішного програмного забезпечення. Вдало побудована архітектура дозволила забезпечити ефективну роботу додатку, його масштабованість. Ретельно розроблений інтерфейс і зрозумілий порядок роботи з системою дозволить користувачам легко орієнтуватися та використовувати додаток без зайвих зусиль.

Тестування роботи додатку виявляло помилки, неполадки та неочікувану поведінку, через що допомогло виправити їх та покращити якість додатку перед впровадженням. Це забезпечує надійну та безпроблемну роботу програмного забезпечення, зменшує ризик негативного впливу на користувачів та підвищує задоволення від використання додатку.

Комбінація всіх цих аспектів допомогла створити високоякісний додаток, який відповідає вимогам користувачів та досягає поставлених цілей. Правильно спроектована архітектура дозволила ефективно використовувати ресурси, підтримувати легку розширюваність та утримувати стабільність системи. Інтуїтивний інтерфейс забезпечує зручну навігацію та взаємодію з додатком, що сприяє покращенню користувацького досвіду.

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
						49
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

В рамках дипломної роботи "Hospital Management Studio" було успішно розроблено десктопний застосунок, який має важливе значення для полегшення управління медичним закладом. Протягом всієї роботи був проведений детальний аналіз проблематики управління медичним закладом та визначені вимоги до застосунку. Було розглянуто різні аспекти реалізації та вибрані оптимальні методи та інструменти для розробки.

У розділі "Аналіз проблематики, методів та засобів вирішення задачі" були визначені вимоги до застосунку і проведений докладний аналіз аналогів та технічних вимог до системи. Це дозволило визначити оптимальний напрямок роботи та забезпечити високу функціональність застосунку.

У розділі "Проектування та реалізація застосунку" була розроблена архітектура застосунку та створені всі необхідні складові. Був розроблений інтерфейс користувача, логіка управління медичним закладом, система навігації та збереження даних.

У розділі "Порядок роботи та тестування роботи застосунку" був описаний порядок роботи з застосунком для управління медичним закладом. Розглянуті кроки, що потрібно виконати для ефективного управління закладом, включаючи створення нових записів, збереження даних та генерацію звітів.

Також у цьому розділі було виконане тестування роботи застосунку з використанням різних методів, включаючи ручне тестування та проведення тестових сценаріїв. Тестування дозволило виявити та виправити помилки, забезпечуючи стабільну та надійну роботу застосунку.

В результаті роботи було створено ефективний та зручний у використанні застосунок, який допомагає управляти медичним закладом. Застосунок має чітку структуру та забезпечує зручний інтерфейс для користувачів. Результати цієї роботи можуть бути використані для поліпшення процесу управління медичним закладом та забезпечення ефективної роботи.

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

Список використаних джерел

1. Документація C# [Електронний ресурс] – Режим доступу до ресурсу:
<https://learn.microsoft.com/en-us/dotnet/csharp/>
2. Документація Star UML [Електронний ресурс] – Режим доступу до ресурсу: <https://staruml.sourceforge.net/>
3. Документація CRUD [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.c-sharpcorner.com/article/crud-operations-in-windows-applications-using-c-sharp/>
4. Документація T-SQL [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.simplilearn.com/tutorials/sql-tutorial/transact-sql>
5. Документація WPF [Електронний ресурс] – Режим доступу до ресурсу:
<https://learn.microsoft.com/en-us/dotnet/desktop/wpf/overview/?view=netdesktop-7.0>
6. Документація MS SQL SERVER [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Microsoft_SQL_Server
7. Документація ADO.NET EDM [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/entity-data-model>
8. Документація Індекс таблиць [Електронний ресурс] – Режим доступу до ресурсу: <https://www.wikiwand.com/uk>
9. Документація Тестування [Електронний ресурс] – Режим доступу до ресурсу: <https://medium.com/slalom-build/value-driven-software-performance-testing-8bf998a3a0cb>
10. Документація Еквівалентний поділ Тестування [Електронний ресурс] – Режим доступу до ресурсу: <https://www.geeksforgeeks.org/equivalence-partitioning-method/>
11. Depth Jon Skeet C# in Depth – 102 с.
12. Jeffrey Richter CLR via C# Jeffrey Richter. – 31 с.

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
						51
Змн.	Арк.	№ докум.	Підпис	Дата		

13. Adam Freeman Pro ASP.NET Core 5: Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages – 89 с.
14. Itzik Ben-Gan Microsoft SQL Server 2019 T-SQL Fundamentals – 230 с.
15. Miguel Cebollero Pro T-SQL Programmer's Guide – 21с.
16. Документація WPF + T-SQL [Електронний ресурс] – Режим доступу до ресурсу: <https://www.c-sharpcorner.com/article/wpf-data-storage-using-sql-server/>
17. Karl Wieggers Software Requirements (Developer Best Practices), 2013 – 421 с

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ІЗ	Арк.
						53
Змн.	Арк.	№ докум.	Підпис	Дата		

Компонент App.config

```

<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <!-- For more information on Entity Framework configuration, visit
http://go.microsoft.com/fwlink/?LinkID=237468 -->
    <section name="entityFramework"
type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework,
Version=6.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false"
/>
  </configSections>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
  </startup>
  <entityFramework>
    <providers>
      <provider invariantName="System.Data.SqlClient"
type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />
    </providers>
  </entityFramework>
  <connectionStrings>
    <add name="connect" connectionString="Data Source=DESKTOP-0A7D0F7\SQLEXPRESS;Initial
Catalog=hosp;Integrated Security=True"/>
    <add name="hospEntities"
connectionString="metadata=res://*/DB.Model1.csdl|res://*/DB.Model1.ssdl|res://*/DB.Model1.m
sl;provider=System.Data.SqlClient;provider connection string="data source=DESKTOP-
0A7D0F7\SQLEXPRESS;initial catalog=hosp;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework";"
providerName="System.Data.EntityClient" />
  </connectionStrings>
</configuration>

```

		П.О. Войтко			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

Компонент Window1.xaml

```

namespace db_kursova
{
    /// <summary>
    /// </summary>
    public partial class Window1 : Window
    {

        public Window1()
        {
            InitializeComponent();

            hospEntities db = new hospEntities();

            ~Window1()
            {
                db.Dispose();
            }

            private void btOk_Click(object sender, RoutedEventArgs e)
            {
                if (string.IsNullOrEmpty(tbLogin.Text) ||
string.IsNullOrEmpty(pbPassword.Password))
                {
                    MessageBox.Show("Будь-ласка заповніть усі поля.", "Помилка!",
MessageBoxButton.OK, MessageBoxImage.Error);
                    return;
                }

                bool checkLogin = false;

                foreach (var us in db.users)
                {
                    if (us.UserName == tbLogin.Text)
                    {
                        // doctor123
                        // nurse321
                        // admin111
                        string encrypt_pass = RijndaelAlgorithm.Encrypt(pbPassword.Password);
                        if (us.Passwrod == encrypt_pass)
                        {
                            MainWindow mw = new MainWindow(this,us.UserName);
                            mw.Show();
                            this.Hide();
                            checkLogin = true;
                            break;
                        }
                    }
                }
                if (!checkLogin)
                {
                    MessageBox.Show("Невірно введено логін або пароль. Перевірте
правильність введення.", "Помилка!", MessageBoxButton.OK, MessageBoxImage.Error);
                }

                private void btCancel_Click(object sender, RoutedEventArgs e)
                {
                    this.Close()}}}}}}

```

		П.О. Войтко			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

Компонент Stats.xaml

```

using System.Windows;
using System.Data.SqlClient;
using System.Configuration;
using System.Windows.Input;

namespace db_kursova
{
    public partial class Stats : Window
    {
        public Stats()
        {
            InitializeComponent();
        }
        bool isMoving = false;
        Point CursorToScreenOldPosition, WindowOldPosition;

        private void DockPanel_MouseDown(object sender, MouseButtonEventArgs e)
        {
            isMoving = true;
            WindowOldPosition = new Point(this.Left, this.Top);
            CursorToScreenOldPosition = this.PointToScreen(Mouse.GetPosition(this));
        }

        private void DockPanel_MouseUp(object sender, MouseButtonEventArgs e)
        {
            isMoving = false;
        }

        private void DockPanel_MouseMove(object sender, MouseEventArgs e)
        {
            if (isMoving)
            {
                Point CursorToScreenPosition, delta;

                CursorToScreenPosition = this.PointToScreen(Mouse.GetPosition(this));

                delta = new Point(CursorToScreenPosition.X - CursorToScreenOldPosition.X,
                                CursorToScreenPosition.Y - CursorToScreenOldPosition.Y);

                this.Left = WindowOldPosition.X + delta.X;
                this.Top = WindowOldPosition.Y + delta.Y;
            }
        }
        private void window_loaded(object sender, RoutedEventArgs e)
        {
            label2.Visibility = Visibility.Visible;
            label3.Visibility = Visibility.Visible;
            label4.Visibility = Visibility.Visible;

            SqlConnection connection = new
            SqlConnection(ConfigurationManager.ConnectionStrings["connect"].ConnectionString);

            string query1 = "select (sum(premium) + sum(salary)) from " + " Doctors ";

```

		П.О. Войтко			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						56
Змн.	Арк.	№ докум.	Підпис	Дата		

```

SqlCommand zapros = new SqlCommand(query1, connection);
connection.Open();
SqlDataReader reader = zapros.ExecuteReader();

while (reader.Read())
{
    label3.Content = reader[0].ToString();
}
connection.Close();
reader.Close();

string query2 = "select sum(amount) from " + " Donations ";

zapros = new SqlCommand(query2, connection);
connection.Open();
reader = zapros.ExecuteReader();

while (reader.Read())
{
    label2.Content = reader[0].ToString();
}
connection.Close();
reader.Close();

string query3 = " select sum(amount) - (sum(premium) + sum(salary)) from
Doctors left join Departments on Doctors.id=Departments.DoctorID left join Donations on
Departments.id=Donations.departmentId";
zapros = new SqlCommand(query3, connection);
connection.Open();
reader = zapros.ExecuteReader();

while (reader.Read())
{
    label4.Content = reader[0].ToString();
}
connection.Close();
reader.Close();
    }
}
}

```

		П.О. Войтко			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						57
Змн.	Арк.	№ докум.	Підпис	Дата		

Компонент MainWindow.xaml

```

using db_kursova.DB;
using System;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Data.SqlClient;
using System.Configuration;
using System.Globalization;
using System.Windows.Data;
using System.Windows.Media;

namespace db_kursova
{
    /// <summary>
    /// Логика взаимодействия для MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        int find_t = 0;
        hospEntities db;
        enum Table
        {
            Doctors,
            Departments,
            Donations,
            Specializations,
            Sponsors,
            Vacations,
            Wards,
            DoctorsSpecializations,
            stats2,
            none
        }
        private string[] _skipHeaderNamesList =
        {
            "DoctorsSpecializations",
            "id",
            "Vacations",
            "Specializations",
            "Doctors",
            "Donations",
            "Wards",
            "Departments",
            "Sponsors",
        };

        private Table current_table = Table.none;
        public MainWindow(Window1 window1, string role)
        {
            InitializeComponent();
            window1.Close();
            db = new hospEntities();
            if(role == "Accountant")
            {
                button_ad.IsEnabled = false;
                delete_button.IsEnabled = false;
                update_buton.IsEnabled = false;
            }
        }
    }
}

```

		П.О. Войтко			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        backup_button.IsEnabled = false;

    }
    else if (role == "Admin")
    {
        sponsors.IsEnabled = false;
        donat.IsEnabled = false;
        button_print.Visibility = Visibility.Hidden;
        fin_report.Visibility = Visibility.Hidden;

    }
    else if (role == "Personnel department")
    {
        backup_button.IsEnabled = false;

    }

}

void update()
{
    text_find.Clear();
    if (current_table == Table.Doctors)
    {
        var query =
            from doctor in db.Doctors

            select new { doctor.id, doctor.name, doctor.surname, doctor.salary,
doctor.premium,
                doctor.Adress,
                doctor.PhoneNumber,
                doctor.Education , doctor.DoctorCategory };
        datagrid.ItemsSource = query.ToList();

    }
    else if (current_table == Table.DoctorsSpecializations)
    {
        var query =
            from d_c in db.DoctorsSpecializations

            select new { d_c.id, d_c.Specializations.name,
d_c.Doctors.surname };
        datagrid.ItemsSource = query.ToList();

    }
    else if (current_table == Table.Vacations)
    {
        var query =
            from vac in db.Vacations

            select new { vac.id, vac.Doctors.name, vac.Enddate,
vac.StartDate, };
        datagrid.ItemsSource = query.ToList();

    }
    else if (current_table == Table.Specializations)
    {
        var query =
            from spc in db.Specializations

            select new { spc.id, spc.name };
        datagrid.ItemsSource = query.ToList();
    }
}

```

		П.О. Войтко			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
    else if (current_table == Table.Departments)
    {
        var query =
            from dp in db.Departments

            select new { dp.id, dp.name };
        datagrid.ItemsSource = query.ToList();

    }
    else if (current_table == Table.Sponsors)
    {
        var query =
            from sp in db.Sponsors

            select new { sp.id, sp.name };
        datagrid.ItemsSource = query.ToList();

    }
    //else if (current_table == Table.Wards)
    //{
    //    var query =
    //        from ward in db.Wards

    //        select new { ward.id, ward.name, ward.DepartmentId };
    //    datagrid.ItemsSource = query.ToList();

    //}
    else if (current_table == Table.Donations)
    {
        var query =
            from donat in db.Donations

            select new { donat.id, donat.amount, donat.date,
donat.Departments.name, donat.SponsorId };
        datagrid.ItemsSource = query.ToList();

    }

}

private void Button_Click(object sender, RoutedEventArgs e)
{
    current_table = Table.Doctors;
    label_table.Content = "Doctors table";
    update();
}

private void Button_Click_1(object sender, RoutedEventArgs e)
{
    if (datagrid.SelectedIndex == -1)
    {
        return;
    }
    if (current_table == Table.Doctors)
    {
        int i = 0;
        foreach (var doctor in db.Doctors)

```

		П.О. Войтко			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        {
            if (i == datagrid.SelectedIndex)
            {
                int columnNumber = 0;
                int rowNumber = 1;
                Button buttonAdd = GetButton("Оновити");
                Button buttonCancel = GetButton("Скасувати");
                datagrid2.Children.Clear();
                ShowAddingSection();
                find_t = doctor.id;
                DisplayTableHeader(datagrid2, new Doctors());
                InsertTextBoxIntoGrid(datagrid2, "D", rowNumber, columnNumber++,
false, doctor.name);
                InsertTextBoxIntoGrid(datagrid2, "D", rowNumber, columnNumber++,
false, doctor.surname);
                InsertTextBoxIntoGrid(datagrid2, "D", rowNumber, columnNumber++,
false, doctor.salary.ToString());
                InsertTextBoxIntoGrid(datagrid2, "D", rowNumber, columnNumber++,
false, doctor.premium.ToString());
                InsertTextBoxIntoGrid(datagrid2, "D", rowNumber, columnNumber++,
false, doctor.Adress);
                InsertTextBoxIntoGrid(datagrid2, "D", rowNumber, columnNumber++,
false, doctor.PhoneNumber);
                InsertTextBoxIntoGrid(datagrid2, "D", rowNumber, columnNumber++,
false, doctor.Education);
                InsertTextBoxIntoGrid(datagrid2, "D", rowNumber, columnNumber++,
false, doctor.DoctorCategory);
                buttonAdd.Click += UpdateDoctor_Click;
                buttonCancel.Click += Cancel_click;
                InsertButtonIntoGrid(datagrid2, buttonAdd, rowNumber,
columnNumber++);
                InsertButtonIntoGrid(datagrid2, buttonCancel, rowNumber,
columnNumber);
                break;
            }
            i++;
        }
    }
    else if (current_table == Table.DoctorsSpecializations)
    {
        int i = 0;
        foreach (var d_s in db.DoctorsSpecializations)
        {
            if (i == datagrid.SelectedIndex)
            {
                int columnNumber = 0;
                int rowNumber = 1;

                Button buttonAdd = GetButton("Оновити");
                Button buttonCancel = GetButton("Скасувати");

                datagrid2.Children.Clear();
                ShowAddingSection();

                find_t = d_s.id;
                DisplayTableHeader(datagrid2, new DoctorsSpecializations());

                InsertTextBoxIntoGrid(datagrid2, "D", rowNumber, columnNumber++,
false, d_s.DoctorID.ToString());
                InsertTextBoxIntoGrid(datagrid2, "D", rowNumber, columnNumber++,
false, d_s.SpecializationId.ToString());

                buttonAdd.Click += UpdatedD_s_Click;
            }
        }
    }
}

```

		П.О. Войтко			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        buttonCancel.Click += Cancel_click;
        InsertButtonIntoGrid(datagrid2, buttonAdd, rowNumber,
columnNumber++);
        InsertButtonIntoGrid(datagrid2, buttonCancel, rowNumber,
columnNumber);

        break;
    }
    i++;
}
}
else if (current_table == Table.Vacations)
{
    int i = 0;
    foreach (var vac in db.Vacations)
    {
        if (i == datagrid.SelectedIndex)
        {
            int columnNumber = 0;
            int rowNumber = 1;

            Button buttonAdd = GetButton("Оновити");
            Button buttonCancel = GetButton("Скасувати");

            datagrid2.Children.Clear();
            ShowAddingSection();

            find_t = vac.id;
            DisplayTableHeader(datagrid2, new Vacations());

            InsertTextBoxIntoGrid(datagrid2, "D", rowNumber, columnNumber++,
false, vac.Enddate.ToString());
            InsertTextBoxIntoGrid(datagrid2, "D", rowNumber, columnNumber++,
false, vac.StartDate.ToString());
            InsertTextBoxIntoGrid(datagrid2, "D", rowNumber, columnNumber++,
false, vac.DoctorID.ToString());
            buttonAdd.Click += UpdateVac_Click;
            buttonCancel.Click += Cancel_click;
            InsertButtonIntoGrid(datagrid2, buttonAdd, rowNumber,
columnNumber++);
            InsertButtonIntoGrid(datagrid2, buttonCancel, rowNumber,
columnNumber);

            break;
        }
        i++;
    }
}
else if (current_table == Table.Specializations)
{
    int i = 0;
    foreach (var spc in db.Specializations)
    {
        if (i == datagrid.SelectedIndex)
        {
            int columnNumber = 0;
            int rowNumber = 1;
            Button buttonAdd = GetButton("Оновити");
            Button buttonCancel = GetButton("Скасувати");
            datagrid2.Children.Clear();
            ShowAddingSection();

```

		П.О. Войтко			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        find_t = spc.id;
        DisplayTableHeader(datagrid2, new Specializations());
        InsertTextBoxIntoGrid(datagrid2, "D", rowNumber, columnNumber++,
false, spc.name);

        buttonAdd.Click += UpdateSpc_Click;
        buttonCancel.Click += Cancel_click;
        InsertButtonIntoGrid(datagrid2, buttonAdd, rowNumber,
columnNumber++);

        InsertButtonIntoGrid(datagrid2, buttonCancel, rowNumber,
columnNumber);

        break;
    }
    i++;
}
}
else if (current_table == Table.Departments)
{
    int i = 0;
    foreach (var dpr in db.Departments)
    {
        if (i == datagrid.SelectedIndex)
        {
            int columnNumber = 0;
            int rowNumber = 1;

            Button buttonAdd = GetButton("Оновити");
            Button buttonCancel = GetButton("Скасувати");

            datagrid2.Children.Clear();
            ShowAddingSection();

            find_t = dpr.id;
            DisplayTableHeader(datagrid2, new Departments());

            InsertTextBoxIntoGrid(datagrid2, "D", rowNumber, columnNumber++,
false, dpr.name);

            InsertTextBoxIntoGrid(datagrid2, "D", rowNumber, columnNumber++,
false, dpr.DoctorID.ToString());
            buttonAdd.Click += UpdateDep_Click;
            buttonCancel.Click += Cancel_click;
            InsertButtonIntoGrid(datagrid2, buttonAdd, rowNumber,
columnNumber++);

            InsertButtonIntoGrid(datagrid2, buttonCancel, rowNumber,
columnNumber);

            break;
        }
        i++;
    }
}
else if (current_table == Table.Sponsors)
{
    int i = 0;
    foreach (var sp in db.Sponsors)
    {
        if (i == datagrid.SelectedIndex)
        {
            int columnNumber = 0;
            int rowNumber = 1;
            Button buttonAdd = GetButton("Оновити");
            Button buttonCancel = GetButton("Скасувати");

```

		П.О. Войтко			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        datagrid2.Children.Clear();
        ShowAddingSection();
        find_t = sp.id;
        DisplayTableHeader(datagrid2, new Specializations());
        InsertTextBoxIntoGrid(datagrid2, "D", rowNumber, columnNumber++,
false, sp.name);

        buttonAdd.Click += UpdateSpons_Click;
        buttonCancel.Click += Cancel_click;
        InsertButtonIntoGrid(datagrid2, buttonAdd, rowNumber,
columnNumber++);

        InsertButtonIntoGrid(datagrid2, buttonCancel, rowNumber,
columnNumber);

        break;
    }
    i++;
}
}
//else if (current_table == Table.Wards)
//{
//    int i = 0;
//    foreach (var ward in db.Wards)
//    {
//        if (i == datagrid.SelectedIndex)
//        {
//            int columnNumber = 0;
//            int rowNumber = 1;

//            Button buttonAdd = GetButton("Update");
//            Button buttonCancel = GetButton("Cancel");

//            datagrid2.Children.Clear();
//            ShowAddingSection();

//            find_t = ward.id;
//            DisplayTableHeader(datagrid2, new Wards());

//            InsertTextBoxIntoGrid(datagrid2, "D", rowNumber, columnNumber++,
false, ward.name);
//            InsertTextBoxIntoGrid(datagrid2, "D", rowNumber, columnNumber++,
false, ward.DepartmentId.ToString());
//            buttonAdd.Click += UpdateWard_Click;
//            buttonCancel.Click += Cancel_click;
//            InsertButtonIntoGrid(datagrid2, buttonAdd, rowNumber,
columnNumber++);
//            InsertButtonIntoGrid(datagrid2, buttonCancel, rowNumber,
columnNumber);

//            break;
//        }
//        i++;
//    }
//}
else if (current_table == Table.Donations)
{
    int i = 0;
    foreach (var don in db.Donations)
    {
        if (i == datagrid.SelectedIndex)
        {
            int columnNumber = 0;
            int rowNumber = 1;

```

		П.О. Войтко			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						64
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        Button buttonAdd = GetButton("Оновити");
        Button buttonCancel = GetButton("Скасувати");

        datagrid2.Children.Clear();
        ShowAddingSection();

        find_t = don.id;
        DisplayTableHeader(datagrid2, new Donations());

        InsertTextBoxIntoGrid(datagrid2, "D", rowNumber, columnNumber++,
false, don.amount.ToString());
        InsertTextBoxIntoGrid(datagrid2, "D", rowNumber, columnNumber++,
false, don.date.ToString());
        InsertTextBoxIntoGrid(datagrid2, "D", rowNumber, columnNumber++,
false, don.departmentId.ToString());
        InsertTextBoxIntoGrid(datagrid2, "D", rowNumber, columnNumber++,
false, don.SponsorId.ToString());
        buttonAdd.Click += UpdateDonat_Click;
        buttonCancel.Click += Cancel_click;
        InsertButtonIntoGrid(datagrid2, buttonAdd, rowNumber,
columnNumber++);
        InsertButtonIntoGrid(datagrid2, buttonCancel, rowNumber,
columnNumber);

        break;
    }
    i++;
}
}
db.SaveChanges();
update();
}

private void UpdateDonat_Click(object sender, RoutedEventArgs e)
{
    var a1 = ((TextBox)datagrid2.Children[6]).Text.Trim();
    var a2 = ((TextBox)datagrid2.Children[7]).Text.Trim();
    var a3 = ((TextBox)datagrid2.Children[8]).Text.Trim();
    var a4 = ((TextBox)datagrid2.Children[9]).Text.Trim();
    var don = db.Donations.Find(find_t);

    don.amount = Convert.ToInt32(a1);
    don.date = Convert.ToDateTime(a2);
    don.departmentId = Convert.ToInt32(a3);
    don.SponsorId = Convert.ToInt32(a4);
    don.Departments = db.Departments.Where(r => r.id ==
don.departmentId).FirstOrDefault();
    don.Sponsors = db.Sponsors.Where(r => r.id == don.SponsorId).FirstOrDefault();
    db.SaveChanges();
    update();
    datagrid2.Children.Clear();
    datagrid.IsEnabled = true;
}

//private void UpdateWard_Click(object sender, RoutedEventArgs e)
//{
//    var a1 = ((TextBox)datagrid2.Children[4]).Text.Trim();
//    var a2 = ((TextBox)datagrid2.Children[5]).Text.Trim();
//    var ward = db.Wards.Find(find_t);

//    ward.name = a1;

```

		П.О. Войтко			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						65
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        // ward.DepartmentId = Convert.ToInt32(a2);
        // ward.Departments = db.Departments.Where(r => r.id ==
ward.DepartmentId).FirstOrDefault();
        // db.SaveChanges();
        // update();
        // datagrid2.Children.Clear();
        // datagrid.IsEnabled = true;
    //}

private void UpdateSpons_Click(object sender, RoutedEventArgs e)
{
    var a1 = ((TextBox)datagrid2.Children[3]).Text.Trim();
    var sp = db.Sponsors.Find(find_t);
    sp.name = a1;
    db.SaveChanges();
    update();
    datagrid2.Children.Clear();
    datagrid.IsEnabled = true;
}

private void UpdateDep_Click(object sender, RoutedEventArgs e)
{
    var a1 = ((TextBox)datagrid2.Children[4]).Text.Trim();
    var a2 = ((TextBox)datagrid2.Children[5]).Text.Trim();
    var dp = db.Departments.Find(find_t);

    dp.name = a1;
    dp.DoctorID = Convert.ToInt32(a2);
    dp.Doctors = db.Doctors.Where(r => r.id == dp.DoctorID).FirstOrDefault();

    db.SaveChanges();
    update();
    datagrid2.Children.Clear();
    datagrid.IsEnabled = true;
}

private void UpdateSpc_Click(object sender, RoutedEventArgs e)
{
    var a1 = ((TextBox)datagrid2.Children[3]).Text.Trim();
    var vac = db.Specializations.Find(find_t);
    vac.name = a1;
    db.SaveChanges();
    update();
    datagrid2.Children.Clear();
    datagrid.IsEnabled = true;
}

private void UpdateVac_Click(object sender, RoutedEventArgs e)
{
    var a1 = ((TextBox)datagrid2.Children[5]).Text.Trim();
    var a2 = ((TextBox)datagrid2.Children[6]).Text.Trim();
    var a3 = ((TextBox)datagrid2.Children[7]).Text.Trim();
    var vac = db.Vacations.Find(find_t);

    vac.Enddate = Convert.ToDateTime(a1);
    vac.StartDate = Convert.ToDateTime(a2);
    vac.DoctorID = Convert.ToInt32(a3);
    vac.Doctors = db.Doctors.Where(r => r.id == vac.DoctorID).FirstOrDefault();

    db.SaveChanges();
    update();
    datagrid2.Children.Clear();
    datagrid.IsEnabled = true;
}

```

		П.О. Войтко			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						66
Змн.	Арк.	№ докум.	Підпис	Дата		

```

private void UpdateD_s_Click(object sender, RoutedEventArgs e)
{
    var a1 = ((TextBox)datagrid2.Children[4]).Text.Trim();
    var a2 = ((TextBox)datagrid2.Children[5]).Text.Trim();
    int ddd = 0;

    if (a1 == "" || a2 == "" )
    {
        MessageBox.Show("Заповніть всі текстові поля");
        return;
    }
    if (!Int32.TryParse(a1, out ddd) || !Int32.TryParse(a2, out ddd))
    {
        MessageBox.Show("Дані введено невірно!");
        return;
    }
    var d_s = db.DoctorsSpecializations.Find(find_t);

    d_s.DoctorID = Convert.ToInt32(a1);
    d_s.SpecializationId = Convert.ToInt32(a2);

    d_s.Doctors = db.Doctors.Where(r => r.id == d_s.DoctorID).FirstOrDefault();
    d_s.Specializations = db.Specializations.Where(r => r.id ==
d_s.SpecializationId).FirstOrDefault();

    db.SaveChanges();
    update();
    datagrid2.Children.Clear();
    datagrid.IsEnabled = true;
    MessageBox.Show("Дані успішно оновлено!");
}

private void UpdateDoctor_Click(object sender, RoutedEventArgs e)
{
    decimal ddd = 0;

    //MessageBox.Show(datagrid2.Children[6].GetType().ToString());
    var a1 = ((TextBox)datagrid2.Children[10]).Text.Trim();
    var a2 = ((TextBox)datagrid2.Children[11]).Text.Trim();
    var a3 = ((TextBox)datagrid2.Children[12]).Text.Trim();
    var a4 = ((TextBox)datagrid2.Children[13]).Text.Trim();
    var a5 = ((TextBox)datagrid2.Children[14]).Text.Trim();
    var a6 = ((TextBox)datagrid2.Children[15]).Text.Trim();
    var a7 = ((TextBox)datagrid2.Children[16]).Text.Trim();
    var a8 = ((TextBox)datagrid2.Children[17]).Text.Trim();
    if (a1 == "" || a2 == "" || a3 == "" || a4 == "" || a5 == "" || a6 == "" || a7
== "" || a8 == "")
    {
        MessageBox.Show("Заповніть всі текстові поля");
        return;
    }
    if (!decimal.TryParse(a3, out ddd) || !decimal.TryParse(a4, out ddd))
    {
        MessageBox.Show("Дані введено невірно!");
        return;
    }
    var doctor = db.Doctors.Find(find_t);
    doctor.name = a1;
    doctor.surname = a2;
    doctor.salary = Convert.ToDecimal(a3);
    doctor.premium = Convert.ToDecimal(a4);
    doctor.Adress = a5;
    doctor.PhoneNumber = a6;
    doctor.Education = a7;
}

```

		П.О. Войтко			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        doctor.DoctorCategory = a8;
        db.SaveChanges();
        update();
        datagrid2.Children.Clear();
        datagrid.IsEnabled = true;
        MessageBox.Show("Дані успішно оновлено!!");
    }

    private void Button_Click_2(object sender, RoutedEventArgs e)
    {
        if (datagrid.SelectedIndex == -1)
        {
            MessageBox.Show("Виберіть елемент, який ви хочете видалити!");
            return;
        }

        if (current_table == Table.Doctors)
        {
            var result = MessageBox.Show("Ви впевнені, що хочете видалити це?",
"Deleting Data",
                MessageBoxButton.YesNo);
            if (result == MessageBoxResult.Yes)
            {
                int i = 0;
                foreach (var doctor in db.Doctors)
                {
                    if (i == datagrid.SelectedIndex)
                    {
                        if(!db.DoctorsSpecializations.Where(d=>d.id==doctor.id).Any())
                        {
                            db.Doctors.Remove(doctor);
                            MessageBox.Show("Лікар успішно видалений");
                        }
                        else
                        {
                            MessageBox.Show("Видалення не можливе!Цей лікар пов'язаний з
іншою таблицею!");
                        }
                    }

                    break;
                }
                i++;
            }
            db.SaveChanges();
            update();
        }

        else if (current_table == Table.DoctorsSpecializations)
        {
            var result = MessageBox.Show("Ви впевнені, що хочете видалити це?",
"Deleting Data",
                MessageBoxButton.YesNo);
            if (result == MessageBoxResult.Yes)
            {
                int i = 0;
                foreach (var d_s in db.DoctorsSpecializations)
                {
                    if (i == datagrid.SelectedIndex)
                    {
                        db.DoctorsSpecializations.Remove(d_s);
                    }
                }
            }
        }
    }

```

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
						68
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        break;
    }
    i++;
}
db.SaveChanges();
update();

}

}
else if (current_table == Table.Vacations)
{
    var result = MessageBox.Show("Ви впевнені, що хочете видалити це?",
"Deleting Data",
    MessageBoxButton.YesNo);
    if (result == DialogResult.Yes)
    {
        int i = 0;
        foreach (var vac in db.Vacations)
        {
            if (i == datagrid.SelectedIndex)
            {
                db.Vacations.Remove(vac);

                break;
            }
            i++;
        }
        db.SaveChanges();
        update();
    }

}
else if (current_table == Table.Specializations)
{
    var result = MessageBox.Show("Ви впевнені, що хочете видалити це??",
"Deleting Data",
    MessageBoxButton.YesNo);
    if (result == DialogResult.Yes)
    {
        int i = 0;
        foreach (var spc in db.Specializations)
        {
            if (i == datagrid.SelectedIndex)
            {
                db.Specializations.Remove(spc);

                break;
            }
            i++;
        }
        db.SaveChanges();
        update();
    }

}
else if (current_table == Table.Departments)
{
    var result = MessageBox.Show("Ви впевнені, що хочете видалити це??",
"Deleting Data",

```

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
						69
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        MessageBoxButton.YesNo);
    if (result == MessageBoxResult.Yes)
    {
        int i = 0;
        foreach (var dp in db.Departments)
        {
            if (i == datagrid.SelectedIndex)
            {
                db.Departments.Remove(dp);

                break;
            }
            i++;
        }
        db.SaveChanges();
        update();
    }
}
else if (current_table == Table.Sponsors)
{
    var result = MessageBox.Show("Ви впевнені, що хочете видалити це??",
"Deleting Data",
        MessageBoxButton.YesNo);
    if (result == MessageBoxResult.Yes)
    {
        int i = 0;
        foreach (var sp in db.Sponsors)
        {
            if (i == datagrid.SelectedIndex)
            {
                db.Sponsors.Remove(sp);

                break;
            }
            i++;
        }
        db.SaveChanges();
        update();
    }
}
//else if (current_table == Table.Wards)
//{
//    var result = MessageBox.Show("Are u sure, that u want to delete this?",
"Deleting Data",
//        MessageBoxButton.YesNo);
//    if (result == MessageBoxResult.Yes)
//    {
//        int i = 0;
//        foreach (var ward in db.Wards)
//        {
//            if (i == datagrid.SelectedIndex)
//            {
//                db.Wards.Remove(ward);

//                break;
//            }
//            i++;
//        }
//        db.SaveChanges();

```

		П.О. Войтко			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						70
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        //      update();

        //    }

    //}
    else if (current_table == Table.Donations)
    {
        var result = MessageBox.Show("Ви впевнені, що хочете видалити це??",
"Deleting Data",
        MessageBoxButton.YesNo);
        if (result == MessageBoxResult.Yes)
        {
            int i = 0;
            foreach (var don in db.Donations)
            {
                if (i == datagrid.SelectedIndex)
                {
                    db.Donations.Remove(don);

                    break;
                }
                i++;
            }
            db.SaveChanges();
            update();
        }
    }
}
private Button GetButton(string content)
{
    return new Button()
    {
        VerticalAlignment = VerticalAlignment.Center,
        Margin = new Thickness(5),
        Width = 95,
        Content = content,
        Height = 20,
        FontSize = 14
    };
}
private void ShowAddingSection()
{
    datagrid.IsEnabled = false;
    datagrid2.ColumnDefinitions.Add(new ColumnDefinition() { Width = GridLength.Auto
});
    datagrid2.ColumnDefinitions.Add(new ColumnDefinition() { Width = GridLength.Auto
});
    datagrid2.ColumnDefinitions.Add(new ColumnDefinition() { Width = GridLength.Auto
});
    datagrid2.ColumnDefinitions.Add(new ColumnDefinition() { Width = GridLength.Auto
});
    datagrid2.ColumnDefinitions.Add(new ColumnDefinition() { Width = GridLength.Auto
});

    datagrid2.RowDefinitions.Add(new RowDefinition() { Height = GridLength.Auto });
    datagrid2.RowDefinitions.Add(new RowDefinition() { Height = GridLength.Auto });
}
private void InsertTextBoxIntoGrid(Grid grid, string value, int row, int column,
bool isTextblock = true, string data = "")
{
    if (isTextblock == true)

```

		П.О. Войтко			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						71
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        {
            var cellElement = new TextBlock()
            {
                VerticalAlignment = VerticalAlignment.Center,
                Margin = new Thickness(5),
                Text = value,
                FontSize = 14
            };

            cellElement.SetValue(Grid.ColumnProperty, column);
            cellElement.SetValue(Grid.RowProperty, row);
            grid.Children.Add(cellElement);
        }
        else
        {
            //it is textbox

            var cellElement = new TextBox()
            {
                VerticalAlignment = VerticalAlignment.Center,
                Margin = new Thickness(5),
                Width = 100,
                //Name = value,
                Text = data,
                Height = 20,
                FontSize = 14
            };

            cellElement.SetValue(Grid.ColumnProperty, column);
            cellElement.SetValue(Grid.RowProperty, row);
            grid.Children.Add(cellElement);
        }
    }

    private void DisplayTableHeader<Type>(Grid grid, Type type)
    {
        int counter = 0;
        foreach (var prop in typeof(Type).GetProperties())
        {
            if (_skipHeaderNamesList.Contains(prop.Name))
                continue;

            var textBlock = new TextBlock()
            {
                VerticalAlignment = VerticalAlignment.Center,
                Margin = new Thickness(5),
                Text = prop.Name,
                FontWeight = FontWeights.Bold,
                FontSize = 16,
            };

            textBlock.SetValue(Grid.ColumnProperty, counter);

            grid.Children.Add(textBlock);
            grid.ColumnDefinitions.Add(new ColumnDefinition() { Width = GridLength.Auto,
        });

            counter++;
        }

        grid.ColumnDefinitions.Add(new ColumnDefinition() { Width = GridLength.Auto });
        InsertTextBoxIntoGrid(datagrid2, "Refactor", 0, counter);
    }

```

		П.О. Войтко			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						72
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        counter++;

        grid.ColumnDefinitions.Add(new ColumnDefinition() { Width = GridLength.Auto });
        InsertTextBoxIntoGrid(datagrid2, "Delete", 0, counter);
    }
    private void InsertButtonIntoGrid(Grid grid, Button button, int row, int column)
    {
        button.SetValue(Grid.ColumnProperty, column);
        button.SetValue(Grid.RowProperty, row);
        grid.Children.Add(button);
    }
    private void Button_Click_3(object sender, RoutedEventArgs e)
    {
        int columnNumber = 0;
        int rowNumber = 1;
        Button buttonAdd = GetButton("Add");
        Button buttonCancel = GetButton("Cancel");

        datagrid2.Children.Clear();
        ShowAddingSection();

        if (current_table == Table.Doctors)
        {
            DisplayTableHeader(datagrid2, new Doctors());
            InsertTextBoxIntoGrid(datagrid2, "doc_name", rowNumber, columnNumber++,
false);
            InsertTextBoxIntoGrid(datagrid2, "doc_surname", rowNumber, columnNumber++,
false);
            InsertTextBoxIntoGrid(datagrid2, "doc_salary", rowNumber, columnNumber++,
false);
            InsertTextBoxIntoGrid(datagrid2, "doc_premium", rowNumber, columnNumber++,
false);
            InsertTextBoxIntoGrid(datagrid2, "doc_adress", rowNumber, columnNumber++,
false);
            InsertTextBoxIntoGrid(datagrid2, "doc_phonenumber", rowNumber,
columnNumber++, false);
            InsertTextBoxIntoGrid(datagrid2, "doc_education", rowNumber, columnNumber++,
false);
            InsertTextBoxIntoGrid(datagrid2, "doc_category", rowNumber, columnNumber++,
false);
            buttonAdd.Click += AddDoctor_Click;
        }

        if (current_table == Table.DoctorsSpecializations)
        {
            DisplayTableHeader(datagrid2, new DoctorsSpecializations());

            InsertTextBoxIntoGrid(datagrid2, "Doctor_surname", rowNumber,
columnNumber++, false);
            InsertTextBoxIntoGrid(datagrid2, "Specialization_name", rowNumber,
columnNumber++, false);

            buttonAdd.Click += AddD_S_Click;
        }
        if (current_table == Table.Vacations)
        {
            DisplayTableHeader(datagrid2, new Vacations());

            InsertTextBoxIntoGrid(datagrid2, "Doc_Name", rowNumber, columnNumber++,
false);
            InsertTextBoxIntoGrid(datagrid2, "End_date", rowNumber, columnNumber++,
false);
            InsertTextBoxIntoGrid(datagrid2, "Strar_date", rowNumber, columnNumber++,
false);
        }
    }

```

```

        buttonAdd.Click += AddVac_Click;
    }
    if (current_table == Table.Specializations)
    {
        DisplayTableHeader(datagrid2, new Specializations());

        InsertTextBoxIntoGrid(datagrid2, "Spec_Name", rowNumber, columnNumber++,
false);
        buttonAdd.Click += AddSpec_Click;
    }
    if (current_table == Table.Departments)
    {
        DisplayTableHeader(datagrid2, new Departments());

        InsertTextBoxIntoGrid(datagrid2, "Dep_Name", rowNumber, columnNumber++,
false);
        InsertTextBoxIntoGrid(datagrid2, "Doctor_ID", rowNumber, columnNumber++,
false);
        buttonAdd.Click += AddDepar_Click;
    }
    if (current_table == Table.Sponsors)
    {
        DisplayTableHeader(datagrid2, new Sponsors());

        InsertTextBoxIntoGrid(datagrid2, "Spon_Name", rowNumber, columnNumber++,
false);
        buttonAdd.Click += AddSpon_Click;
    }
    //if (current_table == Table.Wards)
    //{
    //    DisplayTableHeader(datagrid2, new Wards());

    //    InsertTextBoxIntoGrid(datagrid2, "Ward_name", rowNumber, columnNumber++,
false);
    //    InsertTextBoxIntoGrid(datagrid2, "Dep_ID", rowNumber, columnNumber++,
false);
    //    buttonAdd.Click += AddWard_Click;
    //}
    if (current_table == Table.Donations)
    {
        DisplayTableHeader(datagrid2, new Donations());

        InsertTextBoxIntoGrid(datagrid2, "Donat_amount", rowNumber, columnNumber++,
false);
        InsertTextBoxIntoGrid(datagrid2, "Donat_date", rowNumber, columnNumber++,
false);
        InsertTextBoxIntoGrid(datagrid2, "don_dep_id", rowNumber, columnNumber++,
false);
        InsertTextBoxIntoGrid(datagrid2, "don_spon_is", rowNumber, columnNumber++,
false);
        buttonAdd.Click += AddDonat_Click;
    }
    buttonCancel.Click += Cancel_click;
    InsertButtonIntoGrid(datagrid2, buttonAdd, rowNumber, columnNumber++);
    InsertButtonIntoGrid(datagrid2, buttonCancel, rowNumber, columnNumber);
}

private void AddDonat_Click(object sender, RoutedEventArgs e)
{
    var t1 = ((TextBox)datagrid2.Children[6]).Text;
    var t2 = ((TextBox)datagrid2.Children[7]).Text;
    var t3 = ((TextBox)datagrid2.Children[8]).Text;
    var t4 = ((TextBox)datagrid2.Children[9]).Text;
    var don = new Donations()

```

		П.О. Войтко			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						74
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        {
            amount = Convert.ToInt32(t1),
            date = Convert.ToDateTime(t2),
            departmentId = Convert.ToInt32(t3),
            SponsorId = Convert.ToInt32(t4),

        };
        db.Donations.Add(don);
        db.SaveChanges();
        update();
        datagrid.IsEnabled = true;
        datagrid2.Children.Clear();
    }

    //private void AddWard_Click(object sender, RoutedEventArgs e)
    //{
    //    var t2 = ((TextBox)datagrid2.Children[4]).Text;
    //    var t3 = ((TextBox)datagrid2.Children[5]).Text;
    //    var ward = new Wards()
    //    {

    //        name = t2,
    //        DepartmentId = Convert.ToInt32(t3)

    //    };
    //    db.Wards.Add(ward);
    //    db.SaveChanges();
    //    update();
    //    datagrid.IsEnabled = true;
    //    datagrid2.Children.Clear();
    //}

    private void AddSpon_Click(object sender, RoutedEventArgs e)
    {
        var t2 = ((TextBox)datagrid2.Children[3]).Text;
        var sp = new Sponsors()
        {
            name = t2

        };

        db.Sponsors.Add(sp);
        db.SaveChanges();
        update();
        datagrid.IsEnabled = true;
        datagrid2.Children.Clear();
    }

    private void AddDepar_Click(object sender, RoutedEventArgs e)
    {
        var t2 = ((TextBox)datagrid2.Children[4]).Text;
        var t3 = ((TextBox)datagrid2.Children[5]).Text;
        var dep = new Departments()
        {

            name = t2,
            DoctorID = Convert.ToInt32(t3)

        };
        db.Departments.Add(dep);
        db.SaveChanges();
        update();
        datagrid.IsEnabled = true;
        datagrid2.Children.Clear();
    }
}

```

		П.О. Войтко			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						75
Змн.	Арк.	№ докум.	Підпис	Дата		

```

private void AddSpec_Click(object sender, RoutedEventArgs e)
{
    var t2 = ((TextBox)datagrid2.Children[3]).Text;
    var spc = new Specializations()
    {
        name = t2
    };

    db.Specializations.Add(spc);
    db.SaveChanges();
    update();
    datagrid.IsEnabled = true;
    datagrid2.Children.Clear();
}

private void AddVac_Click(object sender, RoutedEventArgs e)
{
    var t2 = ((TextBox)datagrid2.Children[5]).Text;
    var t3 = ((TextBox)datagrid2.Children[6]).Text;
    var t4 = ((TextBox)datagrid2.Children[7]).Text;

    var vac = new Vacations()
    {
        Enddate = Convert.ToDateTime(t2),
        StartDate = Convert.ToDateTime(t3),
        DoctorID = Convert.ToInt32(t4)
    };

    db.Vacations.Add(vac);
    db.SaveChanges();
    update();
    datagrid.IsEnabled = true;
    datagrid2.Children.Clear();
}

private void AddD_S_Click(object sender, RoutedEventArgs e)
{
    int ddd = 0;
    var t2 = ((TextBox)datagrid2.Children[4]).Text;
    var t3 = ((TextBox)datagrid2.Children[5]).Text;
    if (t2 == "" || t3 == "")
    {
        MessageBox.Show("Заповніть всі текстові поля!!");
        return;
    }
    if (!Int32.TryParse(t2, out ddd) || !Int32.TryParse(t3, out ddd))
    {
        MessageBox.Show("Дані невірні! Спробуйте ще раз!");
        return;
    }

    var d_s = new DoctorsSpecializations()
    {
        SpecializationId = Convert.ToInt32(t2),
        DoctorID = Convert.ToInt32(t3),
    };
}

```

		П.О. Войтко			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						76
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        db.DoctorsSpecializations.Add(d_s);
        db.SaveChanges();
        update();
        datagrid.IsEnabled = true;
        datagrid2.Children.Clear();
        MessageBox.Show("Нові дані успішно додані!");
    }

    private void Cancel_click(object sender, RoutedEventArgs e)
    {
        datagrid2.Children.Clear();
        datagrid.IsEnabled = true;
    }

    private void AddDoctor_Click(object sender, RoutedEventArgs e)
    {
        decimal ddd = 0;
        var t2 = ((TextBox)datagrid2.Children[10]).Text;
        var t3 = ((TextBox)datagrid2.Children[11]).Text;
        var t4 = ((TextBox)datagrid2.Children[12]).Text;
        var t5 = ((TextBox)datagrid2.Children[13]).Text;
        var t6 = ((TextBox)datagrid2.Children[14]).Text;
        var t7 = ((TextBox)datagrid2.Children[15]).Text;
        var t8 = ((TextBox)datagrid2.Children[16]).Text;
        var t9 = ((TextBox)datagrid2.Children[17]).Text;
        if (t2=="" || t3 == "" || t4 == "" || t5 == "" || t6 == "" || t7 == "" || t8 == ""
|| t9 == "")
        {
            MessageBox.Show("Заповніть всі текстові поля");

            return;
        }
        if(!decimal.TryParse(t4,out ddd)|| !decimal.TryParse(t5, out ddd))
        {
            MessageBox.Show("Дані невірні!Спробуйте ще раз");

            return;
        }
        var doctor = new Doctors()
        {
            name = t2,
            surname = t3,
            salary = Convert.ToDecimal(t4),
            premium = Convert.ToDecimal(t5),
            Adress = t6,
            PhoneNumber = t7,
            Education=t8,
            DoctorCategory=t9,

        };

        db.Doctors.Add(doctor);
        db.SaveChanges();
        update();
        datagrid.IsEnabled = true;
        datagrid2.Children.Clear();
        MessageBox.Show("Нові дані успішно додані!");
    }

    private void Button_Click_4(object sender, RoutedEventArgs e)
    {
        current_table = Table.DoctorsSpecializations;
        label_table.Content = "DoctorsSpecializations table";
        update();
    }

```

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
						77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    private void Button_Click_5(object sender, RoutedEventArgs e)
    {
        current_table = Table.Vacations;
        label_table.Content = "Vacations table";
        update();
    }

    private void Button_Click_6(object sender, RoutedEventArgs e)
    {
        current_table = Table.Specializations;
        label_table.Content = "Specializations table";
        update();
    }

    private void Button_Click_7(object sender, RoutedEventArgs e)
    {
        current_table = Table.Departments;
        label_table.Content = "Departments table";
        update();
    }

    private void Button_Click_8(object sender, RoutedEventArgs e)
    {
        current_table = Table.Sponsors;
        label_table.Content = "Sponsors table";
        update();
    }

    private void Button_Click_9(object sender, RoutedEventArgs e)
    {
        current_table = Table.Wards;
        label_table.Content = "Wards table";
        update();
    }

    private void Button_Click_10(object sender, RoutedEventArgs e)
    {
        label_table.Content = "Donations table";

        current_table = Table.Donations;

        update();
    }

    private void Button_Click_11(object sender, RoutedEventArgs e)
    {
        if (text_find.Text == "")
        {
            MessageBox.Show("Заповніть текстове поле, якщо ви хочете знайти щось!");
            return;
        }

        if (current_table == Table.Doctors)
        {
            var query =
                from doctor in db.Doctors
                where doctor.name == text_find.Text || doctor.surname == text_find.Text
                || doctor.salary.ToString() == text_find.Text || doctor.premium.ToString()
                == text_find.Text ||
                doctor.Adress == text_find.Text || doctor.PhoneNumber == text_find.Text ||
                doctor.Education == text_find.Text || doctor.DoctorCategory ==
                text_find.Text

```

		П.О. Войтко			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						78
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        select new { doctor.id, doctor.name, doctor.premium, doctor.salary,
doctor.surname,
                doctor.Address,
                doctor.PhoneNumber,
                doctor.Education,
                doctor.DoctorCategory
        };
        if (query.ToList().Count != 0)
        {
            datagrid.ItemsSource = query.ToList();
        }
        else
        {
            MessageBox.Show("Нічого не знайдено!");
        }
    }
    else if (current_table == Table.DoctorsSpecializations)
    {
        var query =
            from d_c in db.DoctorsSpecializations
            where d_c.DoctorID.ToString() == text_find.Text
            select new { d_c.id, d_c.Specializations.name,
d_c.Doctors.surname };
        if (query.ToList().Count != 0)
        {
            datagrid.ItemsSource = query.ToList();
        }
        else
        {
            MessageBox.Show("Нічого не знайдено!");
        }
    }
    else if (current_table == Table.Vacations)
    {
        var query =
            from vac in db.Vacations
            where vac.DoctorID.ToString() == text_find.Text ||
vac.Enddate.ToString() == text_find.Text || vac.StartDate.ToString() == text_find.Text
            select new { vac.id, vac.Doctors.name, vac.Enddate,
vac.StartDate, };
        if (query.ToList().Count != 0)
        {
            datagrid.ItemsSource = query.ToList();
        }
        else
        {
            MessageBox.Show("Нічого не знайдено!");
        }
    }
    else if (current_table == Table.Specializations)
    {
        var query =
            from spc in db.Specializations
            where spc.name == text_find.Text
            select new { spc.id, spc.name };
        if (query.ToList().Count != 0)
        {
            datagrid.ItemsSource = query.ToList();
        }
    }

```

		П.О. Войтко			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						79
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
    else
    {
        MessageBox.Show("Нічого не знайдено!");
    }
}
else if (current_table == Table.Departments)
{
    var query =
        from dp in db.Departments
        where dp.name == text_find.Text
        select new { dp.id, dp.name, dp.Doctors.surname };
    if (query.ToList().Count != 0)
    {
        datagrid.ItemsSource = query.ToList();
    }
    else
    {
        MessageBox.Show("Нічого не знайдено!");
    }
}
else if (current_table == Table.Sponsors)
{
    var query =
        from sp in db.Sponsors
        where sp.name == text_find.Text
        select new { sp.id, sp.name };
    if (query.ToList().Count != 0)
    {
        datagrid.ItemsSource = query.ToList();
    }
    else
    {
        MessageBox.Show("Нічого не знайдено!");
    }
}
//else if (current_table == Table.Wards)
//{
//    var query =
//        from ward in db.Wards
//        where ward.name == text_find.Text
//        select new { ward.id, ward.name, ward.DepartmentId };
//    if (query.ToList().Count != 0)
//    {
//        datagrid.ItemsSource = query.ToList();
//    }
//    else
//    {
//        MessageBox.Show("Nothing is find!");
//    }
//}
else if (current_table == Table.Donations)
{
    var query =
        from donat in db.Donations
        where donat.amount.ToString() == text_find.Text ||
donat.date.ToString() == text_find.Text
        select new { donat.id, donat.amount, donat.date,
donat.Departments.name, donat.SponsorId };
    if (query.ToList().Count != 0)
    {

```

		П.О. Войтко			ІІЗ.КР.Б – 121 – 23 - ІІЗ	Арк.
						80
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        datagrid.ItemsSource = query.ToList();
    }
    else
    {
        MessageBox.Show("Нічого не знайдено!");
    }
}
else
{
    MessageBox.Show("Виберіть таблицю, з якою хочете працювати!!");
}
}

private void Button_Click_12(object sender, RoutedEventArgs e)
{
    try
    {
        string dbname = db.Database.Connection.Database;
        string sqlCommand = @"

BACKUP DATABASE hosp
TO DISK = 'C:\Program Files\Microsoft SQL
Server\MSSQL15.SQLEXPRESS\MSSQL\Backup\hosp_FullDbBkup.bak'
WITH INIT, NAME = 'Hospital Full Db backup',
DESCRIPTION = 'Hospital Full Database Backup';

db.Database.ExecuteSqlCommand(System.Data.Entity.TransactionalBehavior.DoNotEnsureTransactio
n, string.Format(sqlCommand, dbname, ""));
    }
    catch
    {
        MessageBox.Show("БЕКАП БД ПРОЙШОВ НЕВДАЛО!Сробуйте ще раз");
        return;
    }
    MessageBox.Show("БЕКАП ПРОЙШОВ УСПІШНО!");
}

private void button_print_Click(object sender, RoutedEventArgs e)
{
    if(datagrid.Items.Count ==0)
    {
        MessageBox.Show("Datagrid is null");
        return;
    }

    PrintDialog printDialog = new PrintDialog();

    if (printDialog.ShowDialog() == true)
    {
        printDialog.PrintVisual(datagrid, "Print data grid");
    }
}

private void Button_Click_14(object sender, RoutedEventArgs e)
{
    Window1 w = new Window1();
    this.Close();
    w.Show();
}
}

```

		П.О. Войтко			ІПЗ.КР.Б – 121 – 23 - ІПЗ	Арк.
						81
Змн.	Арк.	№ докум.	Підпис	Дата		