Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп’ютерних наук

Допущено до захисту
Завідувач кафедри комп’ютерних наук
доктор технічних наук, професор
Андрій БОНДАРЧУК
« 01 » червня 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня «бакалавр»
Спеціальність 122 Комп’ютерні науки
Освітня програма 122.00.01 Інформатика

Тема: ІНФОРМАЦІЙНИЙ ДОДАТОК ДЛЯ УПРАВЛІННЯ РОЗКЛАДОМ ЗАНЯТЬ

Виконав
Студент групи ІНб-2-22-4.0д
Забродський Владислав Олексійович
_____
(підпис)

Науковий керівник
к.пед.н., доцент
Бойко М.А.
_____
(підпис)

Київ – 2026

Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп'ютерних наук

«Затверджую»
Завідувач кафедри комп’ютерних наук,
доктор технічних наук, професор
Андрій БОНДАРЧУК
31.жовтня 2025

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА
«Інформаційний додаток для управління розкладом занять»

Виконавець – студент групи ІНб-2-22-4.0д,
спеціальності 122 Комп’ютерні науки,
освітньої програми 122.00.01 Інформатика
Забродський Владислав Олексійович

1. Вихідні дані: нормативні та методичні матеріали щодо організації освітнього процесу; документація Microsoft .NET, C#, Windows Forms, ADO.NET, SQL Server та Azure SQL; наукові джерела з питань проєктування інформаційних систем та баз даних.
2. Основні завдання: проаналізувати предметну область та існуючі рішення; визначити вимоги до системи; спроєктувати архітектуру, базу даних та інтерфейс; реалізувати застосунок на C# Windows Forms з ADO.NET та SQL Server; виконати тестування та оцінити ефективність рішення.
3. Пояснювальна записка: обсяг 35–45 сторінок формату А4 з дотриманням вимог стандарту та методичних рекомендацій кафедри.
4. Графічні матеріали: UML-діаграми, ER-діаграма бази даних, схема архітектури, скріншоти інтерфейсу, презентація доповіді.
5. Додатки: Додаток А – фрагменти вихідного коду C#/Windows Forms; Додаток Б – SQL-структура бази даних.
6. Строк подання роботи на кафедру: 01 червня 2026 року

Науковий керівник
к.пед.н., доцент
_____Бойко М.А.

Виконавець:
_____ Забродський В.О.

КАЛЕНДАРНИЙ ПЛАН

№	Етапи виконання роботи	Термін виконання	Примітка
1	Затвердження теми кваліфікаційної роботи бакалавра	31.10.25	Виконано
2	Аналіз предметної області управління розкладом занять, вивчення існуючих програмних рішень, формування цілей і завдань	01.11.25 – 11.01.26	Виконано
3	Аналіз вимог, проєктування архітектури застосунку, структури бази даних SQL Server та моделі основних сутностей	26.01.26 – 15.03.26	Виконано
4	Дослідження та реалізація модулів застосунку засобами C# Windows Forms та ADO.NET	16.03.26 – 31.03.26	Виконано
5	Розробка модулів навчальних сесій, часових слотів, правил розміщення та генерації розкладу	01.04.26 – 15.04.26	Виконано
6	Тестування функціональних, валідаційних та сценарних сценаріїв, виправлення помилок	16.04.26 – 30.04.26	Виконано
7	Впровадження та оцінка ефективності розробленого рішення	01.05.26 – 15.05.26	Виконано
8	Підготовка пояснювальної записки, графічних матеріалів, додатків	25.05.25 – 12.06.26	Виконано
9	Захист кваліфікаційної роботи	16.06.26	

«Затверджую»
Науковий керівник
к.пед.н., доцент Бойко М.А.

Індивідуальний план склав
Забороцький В.О.

РЕФЕРАТ

Кваліфікаційна робота бакалавра: 55 с., 3 розділи, 12 рисунків, 18 таблиць, 49 джерел, 2 додатки.

Актуальність теми зумовлена потребою закладів освіти у впорядкованому цифровому інструменті для підготовки та супроводу розкладу занять. Формування розкладу охоплює значну кількість взаємопов'язаних обмежень: зайнятість викладачів, доступність аудиторій, часові слоти, навчальні групи, типи занять і тривалість сесій. Ручне або табличне ведення таких даних підвищує ризик дублювання, конфліктів і втрати актуальності інформації, тому розробка такого програмного продукту в межах бакалаврської роботи є доцільною як навчально-практична реалізація ключових принципів побудови систем управління розкладом: формування довідників, моделювання навчальних сесій, перевірка базових обмежень і збереження даних у реляційній базі.

Метою роботи є розробка та дослідження програмного забезпечення для автоматизації управління розкладом занять у закладі освіти на основі архітектурних рішень та методів комп'ютерних наук.

Об'єктом дослідження є процес автоматизованого управління розкладом занять у закладі освіти засобами програмного забезпечення.

Предметом дослідження є архітектура програмного забезпечення, алгоритми обробки даних та програмні компоненти, що реалізують функції автоматизованого формування, перевірки та ведення навчального розкладу.

Завдання:

1. проаналізувати предметну область автоматизації управління розкладом занять та існуючі програмні рішення на основі методів комп'ютерних наук;
2. визначити функціональні й нефункціональні вимоги до програмного забезпечення та обґрунтувати вибір технологічного стеку;

3. розробити архітектуру програмного забезпечення, структури даних і алгоритми обробки інформації для реалізації основних функцій системи;
4. реалізувати програмні модулі застосунку засобами C# та Windows Forms із використанням ADO.NET і SQL Server як системи управління базами даних;
5. провести функціональне та сценарне тестування програмного забезпечення й оцінити якість розробленого рішення відповідно до визначених вимог.

Основні результати. У роботі проаналізовано сучасні освітні інформаційні системи та системи управління розкладом, визначено вимоги до системи, спроєктовано архітектуру, модель даних та інтерфейс, описано реалізацію модулів Windows Forms-застосунку мовою C# із використанням ADO.NET та SQL Server/Azure SQL. Виконано функціональне та сценарне тестування основних модулів застосунку, за результатами якого підтверджено працездатність базових операцій із довідниками, навчальними сесіями, статистикою та формуванням розкладу, а також визначено напрями подальшого вдосконалення.

Практичне значення дослідження полягає в тому, що розроблений застосунок може бути використаний як навчально-практичний прототип локальної системи управління розкладом занять.

Він демонструє підхід до побудови довідників, моделювання навчальних сесій, збереження даних у реляційній базі, перевірки базових обмежень і формування аналітичних показників для навчального відділу.

Ключові слова: розклад занять, інформаційний додаток, Windows Forms, C#, SQL Server, ADO.NET, база даних, навчальна сесія, автоматизація, тестування.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ADO.NET	ActiveX Data Objects .NET – технологія доступу до даних платформи .NET Framework.
CRUD	Create, Read, Update, Delete – базові операції створення, читання, оновлення та видалення даних.
DAL	Data Access Layer – шар доступу до даних у багаторівневій архітектурі застосунку.
ДСТУ	Державний стандарт України.
ЄДЕБО	Єдина державна електронна база з питань освіти.
EMIS	Education Management Information System – інформаційна система управління освітою.
ER-модель	Entity-Relationship model – модель «сутність – зв'язок», що використовується для проєктування баз даних.
LMS	Learning Management System – система управління навчанням.
SIS	Student Information System – система управління студентськими даними.
SQL	Structured Query Language – структурована мова запитів для роботи з реляційними базами даних.
UI	User Interface – користувацький інтерфейс.
UML	Unified Modeling Language – уніфікована мова моделювання для проєктування програмних систем.
ЗВО	Заклад вищої освіти.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасних умовах цифровізації освіти інформаційні технології відіграють ключову роль у забезпеченні ефективного функціонування навчальних закладів. Зростання обсягів даних, ускладнення структури освітнього процесу та необхідність швидкого прийняття управлінських рішень зумовлюють потребу в автоматизації основних організаційних процесів, одним з яких є формування та управління розкладом занять [18; 20].

Розклад занять є одним із ключових елементів організації освітнього процесу. Через нього узгоджуються навчальні плани, доступність викладачів, місткість аудиторій, типи занять, часові обмеження, навантаження студентів та адміністративні рішення навчального закладу. Там, де розклад формується вручну або в розрізнених таблицях, зростає ймовірність виникнення помилок та конфліктів у розкладі.

Сучасний заклад освіти оперує великим обсягом взаємопов'язаних даних: відомості про викладачів, навчальні дисципліни, аудиторний фонд, групи, підгрупи, часові інтервали, особисті обмеження та побажання існують або окремо, або в таблицях без повної перевірки узгодженості. Якщо ці дані не об'єднані в єдину систему, кожна зміна потребує ручного коригування взаємопов'язаних даних.

Необхідність створення власного програмного забезпечення пояснюється навчально-практичною постановкою задачі. Такі комерційні рішення, як Untis або aSc Timetables, мають ширший функціонал, проте вони є готовими продуктами і не розкривають внутрішню логіку проєктування, структуру даних, обробку подій, валідацію й зв'язок між модулями. У межах бакалаврської роботи важливо не просто скористатися готовим інструментом, а дослідити принципи побудови такого програмного забезпечення та реалізувати його ключові компоненти власноруч.

Метою роботи є розробка та дослідження програмного забезпечення для автоматизації управління розкладом занять у закладі освіти на основі архітектурних рішень та методів комп'ютерних наук.

Для досягнення мети поставлено п'ять завдань:

1. проаналізувати предметну область автоматизації управління розкладом занять та існуючі програмні рішення на основі методів комп'ютерних наук;
2. визначити функціональні й нефункціональні вимоги до програмного забезпечення та обґрунтувати вибір технологічного стеку;
3. розробити архітектуру програмного забезпечення, структури даних і алгоритми обробки інформації для реалізації основних функцій системи;
4. реалізувати програмні модулі застосунку засобами C# та Windows Forms із використанням ADO.NET і SQL Server як системи управління базами даних;
5. провести функціональне та сценарне тестування програмного забезпечення й оцінити якість розробленого рішення відповідно до визначених вимог.

Об'єктом дослідження є процес автоматизованого управління розкладом занять у закладі освіти засобами програмного забезпечення.

Предметом дослідження є архітектура програмного забезпечення, алгоритми обробки даних та програмні компоненти, що реалізують функції автоматизованого формування, перевірки та ведення навчального розкладу.

Методи дослідження включають системний аналіз предметної області, порівняльний аналіз програмних рішень, моделювання структур даних, проєктування архітектури програмного забезпечення, аналіз вихідного коду та функціональне тестування.

Практичне значення роботи полягає в тому, що розроблений програмний застосунок може бути використаний як навчально-практичний прототип локальної системи управління розкладом занять. Рішення демонструє підхід до формування довідників, моделювання навчальних сесій, перевірки базових обмежень і збереження даних у реляційній базі.

Науково-практична новизна роботи полягає в адаптації принципів побудови програмного забезпечення для управління розкладом до компактного настільного застосунку, який поєднує довідникову структуру, модель навчальних сесій,

перевірку базових обмежень та локальне збереження даних у реляційній базі. На відміну від готових комерційних систем, у роботі акцент зроблено на прозорості архітектурних рішень, зрозумілості програмної реалізації та можливості подальшого розширення навчального прототипу.

Галузь застосування результатів роботи охоплює навчальні заклади, кафедри, навчальні відділи та малі освітні підрозділи, де необхідно організувати локальний облік викладачів, груп, аудиторій, навчальних сесій і часових обмежень без розгортання складної корпоративної платформи.

Апробація результатів роботи здійснена шляхом підготовки матеріалів за тематикою інформаційних технологій в освіті та аналізу практичної реалізації програмного забезпечення для управління розкладом занять. Окремі положення роботи можуть бути використані під час підготовки доповіді й презентаційних матеріалів до захисту.

Структура роботи відповідає поставленій меті та складається зі вступу, трьох розділів, висновків, списку використаних джерел і двох додатків. У першому розділі розглянуто теоретичні основи автоматизації управління розкладом, у другому — проєктування архітектури програмного забезпечення, структур даних та інтерфейсу, у третьому — реалізацію програмних модулів, тестування, оцінку якості та напрями подальшого розвитку системи.

РОЗДІЛ 1

ТЕОРЕТИЧНІ ОСНОВИ УПРАВЛІННЯ РОЗКЛАДОМ

1.1 Роль інформаційних систем в освітньому процесі

Інформаційні системи давно перестали бути допоміжною зручністю для освіти. У сучасному закладі вони виконують роль інфраструктурної основи, через яку збираються, обробляються та передаються дані між адміністрацією, викладачами й здобувачами освіти. Класичний ручний документообіг був прийнятним тоді, коли кількість груп, дисциплін і викладачів була відносно невеликою, проте за умов постійного збільшення обсягів навчальної інформації та зміни освітніх траєкторій такий підхід уже не забезпечує належного рівня точності [18; 23].

Освітня інформаційна система об’єднує дані про контингент студентів, кадровий склад, навчальні плани, аудиторний фонд, оцінювання, відвідування та розклад. Її головне завдання полягає не лише у зберіганні записів, а у підтримці узгодженості між ними: зміна дисципліни або кількості годин впливає на навантаження викладачів, структуру сесій, аудиторний фонд і кінцевий розклад. Саме тому довідникова база системи є не формальністю, а активним елементом управління навчальним процесом.

У контексті управління розкладом інформаційна система виконує три базові функції. По-перше, вона є довідковою базою, де зберігаються викладачі, предмети, аудиторії, групи, типи занять і часові інтервали. По-друге, вона реалізує правила перевірки, що знижують ризик некоректних записів. По-третє, вона забезпечує аналітичний контроль: облік зареєстрованих сутностей, виявлення дублювань і формування статистики по навантаженню. Перевага цифрового підходу полягає у повторюваності й контрольованості процесу: правила і дані фіксуються явно, їх можна перевіряти, удосконалювати та передавати іншому користувачу без втрати логіки [18].

Інформаційні системи також підсилюють прозорість освітнього процесу. Коли розклад, зміни, аудиторії та навчальні сесії ведуться централізовано, адміністрація отримує єдину картину, студенти бачать актуальні дані, викладачі розуміють власне навантаження, а навчальний відділ може оперативно реагувати на зміни. Для невеликих закладів або окремих підрозділів достатньо локального настільного застосунку з реляційною базою даних, тоді як для великих університетів доцільні комплексні SIS/EMIS-рішення. Проте логіка в обох випадках однакова: система має бути надійною, зрозумілою, контрольованою та здатною працювати з реальними обмеженнями навчального процесу.

У наданому проєкті ця ідея реалізована через набір модулів Windows Forms: робочі дні та години, викладачі, предмети, студентські групи, аудиторії, часові слоти, сесії, статистика та генерація розкладу. Настільний додаток швидко запускається, добре підходить для локального використання й дозволяє методисту працювати без складної веб-інфраструктури.

У таблиці 1.1 подано узагальнення відповідних характеристик, що використовуються для подальшого проєктування системи.

Таблиця 1.1.

Роль інформаційних систем в освітньому процесі

Напрям	**Зміст**	**Практичний ефект**
Централізація даних	Єдине зберігання відомостей про студентів, викладачів, аудиторії та дисципліни	Зменшення дублювання і помилок
Автоматизація операцій	Підтримка CRUD-операцій, пошуку, фільтрації, звітів	Економія часу навчального відділу
Контроль конфліктів	Перевірка зайнятості викладачів, груп, аудиторій і слотів	Підвищення коректності розкладу
Аналітика	Статистичні показники щодо довідників і навантаження	Обґрунтовані управлінські рішення
Прозорість	Доступ до актуальних даних відповідно до ролі користувача	Краща координація між учасниками

1.2 Особливості формування розкладу занять

Формування розкладу занять є складною організаційно-інформаційною задачею, що вимагає одночасного врахування значної кількості параметрів і правил. Необхідно не просто розставити заняття по днях і годинах, а узгодити викладача, дисципліну, студентську групу або підгрупу, тип заняття, аудиторію, місткість, тривалість, часовий слот, недоступні години та організаційні побажання. У теорії такі задачі розглядаються як задачі розкладу та задоволення обмежень, де рішення вважається прийнятним лише за умови дотримання заданого набору правил [39; 40].

Обмеження поділяються на жорсткі та м’які. Жорсткі обмеження не можна порушувати: зайнятість викладача, групи й аудиторії, відповідність типу приміщення типу заняття, доступність часу. М’які обмеження бажано враховувати: рівномірність навантаження протягом тижня, мінімізація перерв між заняттями, зручність початку занять, уникнення складних дисциплін наприкінці дня. Якісна система має явно підтримувати жорсткі обмеження і створювати основу для подальшої оптимізації м’яких [41; 42]. Проєктна специфікація Timetable Management System, використана як вихідний матеріал у роботі, передбачає введення робочих днів, годин, часових слотів тривалістю одна година або тридцять хвилин, а також окреме управління недоступним часом викладачів, груп, підгруп, сесій і аудиторій [46].

У практичному застосунку задача розкладу розкладається на кілька послідовних етапів. Спочатку наповнюються довідники: викладачі, предмети, групи, теги, аудиторії. Потім створюються сесії, які пов’язують ці довідники в конкретну навчальну подію. Далі визначаються спеціальні правила: послідовні заняття, паралельні заняття, заняття, що не повинні накладатися. Після цього система може призначати аудиторії, враховувати недоступні часові інтервали та генерувати підсумковий варіант розкладу. З педагогічної точки зору автоматизація

розкладу має і методичну цінність: вона допомагає зробити навантаження більш збалансованим і передбачуваним, що безпосередньо впливає на якість навчання.

У таблиці 1.2 подано узагальнення відповідних характеристик, що використовуються для подальшого проєктування системи.

Таблиця 1.2.

Класифікація обмежень під час формування розкладу

Тип	**Приклад**	**Наслідок порушення**	**Спосіб контролю**
Жорстке	Викладач не може проводити дві пари одночасно	Розклад некоректний	Перевірка перетину за викладачем і часом
Жорстке	Аудиторія не може бути зайнята двома сесіями	Конфлікт приміщення	Перевірка Room + TimeSlot
Жорстке	Лабораторна робота потребує лабораторії	Неможливість проведення	Відповідність Tag та RoomType
Жорстке	Кількість студентів не перевищує місткість	Неможливість розміщення	Порівняння StudentCount і Capacity
М’яке	Бажано уникати перерв між заняттями	Незручний розклад	Оптимізаційний критерій
М’яке	Рівномірний розподіл навантаження	Перевтома студентів	Аналітичний контроль

Отже, першочерговими для програмної реалізації є жорсткі обмеження, оскільки їх порушення робить розклад непридатним для практичного використання. М’які обмеження можуть бути враховані на наступних етапах розвитку системи як критерії оптимізації.

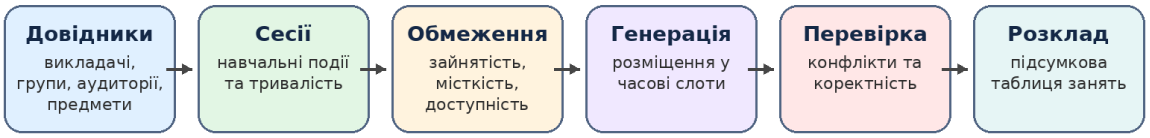

Рисунок 1.1. Узагальнений процес формування та перевірки навчального розкладу

1.3 Аналіз існуючих програмних рішень

Сучасні програмні рішення для цифровізації освітнього процесу сформувалися як відповідь на різні функціональні потреби закладів освіти. Їх

доцільно поділяти на системи управління навчанням (LMS), системи управління студентськими даними (SIS), спеціалізовані системи формування розкладу та аналітичні системи управління освітою (EMIS). Кожен клас має власну логіку функціонування, набір переваг і обмежень.

LMS-системи орієнтовані передусім на організацію навчального контенту, комунікацію між викладачем і студентом, виконання завдань і контроль знань. Moodle є одним із найпоширеніших представників цього класу: відкритий код, гнучка структура курсів, підтримка ролей, плагінів і зовнішніх інтеграцій [1; 2]. Проте Moodle не є спеціалізованою системою формування розкладу: він може відображати навчальні активності, але не виконує повноцінну оптимізацію часових слотів, аудиторій і доступності викладачів. Google Classroom реалізує інший підхід: простий запуск, інтеграція з Google Workspace, зрозуміле створення курсів і завдань [4; 5]. Через простоту система має обмежену адміністративну глибину. Canvas LMS є хмарним комерційним рішенням із сучасним інтерфейсом [6; 7], але не замінює SIS і не вирішує задачу автоматичного складання розкладу.

Системи класу SIS спрямовані на централізований облік студентських даних, відвідуваності, оцінювання та частково розкладу. openSIS підтримує роботу з даними студентів, персоналу, курсів, оцінок і звітності [8; 9]. Українські рішення, зокрема АС «Деканат» та ЄДЕБО, орієнтовані на документообіг, звітність, облік студентів і взаємодію з державними реєстрами [41; 42], але не є повноцінними системами автоматичного формування розкладу. Найбільш релевантними до теми є спеціалізовані системи розкладу: Untis/WebUntis і aSc Timetables [39; 40]. Вони підтримують автоматичну генерацію, перевірку обмежень, роботу із замінами та публікацію онлайн-розкладу, однак є комерційними й не завжди відповідають навчальній меті бакалаврського проєкту, де важливо дослідити внутрішню логіку побудови системи.

Отже, розробка власного застосунку є доцільною не як спроба конкурувати з промисловими продуктами, а як навчально-практична реалізація ключових

принципів системи управління розкладом. Власна розробка дає змогу показати структуру сутностей, алгоритмічні обмеження, роботу з базою даних, інтерфейсні сценарії та напрями рефакторингу.

У таблиці 1.3 подано узагальнення відповідних характеристик, що використовуються для подальшого проєктування системи.

Таблиця 1.3.

Критерії порівняння існуючих програмних рішень

Критерій	**Зміст критерію**	**Значення для теми**
Автоматизація розкладу	Наявність механізмів створення, перевірки та коригування розкладу	Визначає придатність системи до планування
Централізація даних	Зберігання довідників, груп, викладачів, аудиторій і дисциплін	Зменшує дублювання і забезпечує цілісність
Контроль конфліктів	Виявлення перетинів викладача, групи, аудиторії та часу	Формує базову коректність розкладу
Вартість і доступність	Ліцензійна модель, складність впровадження	Впливає на практичну доцільність
Навчальна прозорість	Можливість дослідити внутрішню логіку реалізації	Важлива для кваліфікаційної роботи

У таблиці 1.4 подано узагальнення відповідних характеристик, що використовуються для подальшого проєктування системи.

Таблиця 1.4.

Порівняння існуючих програмних рішень

Рішення	Клас	Сильні сторони	Обмеження для задачі розкладу	Висновок для проєктування
Moodle	LMS	Відкритий код, курси, ролі, плагіни	Не є спеціалізованою системою розкладу	Використати ідею відкритості й модульності, але не як базу розкладу.
Google Classroom	LMS	Простота, інтеграція з Google Workspace	Недостатньо адміністративної глибини	Врахувати простоту інтерфейсу, однак не орієнтувати систему лише на курси.
Canvas	LMS	Хмарність, сучасний UI, інтеграції	Комерційність, не замінює SIS	Використати принцип сучасного UI, але реалізувати локальний прототип.
openSIS	SIS	Студентські дані, розклад, аналітика	Потребує адаптації під локальні вимоги	Врахувати централізацію даних, але без надмірної складності SIS.
АС «Деканат»	SIS/адміністративна	Орієнтація на ЗВО і українську звітність	Менш сучасний інтерфейс	Застосувати досвід українського освітнього обліку для термінології.
Untis/WebUntis	Timetable	Заміни, онлайн-розклад, мобільний доступ	Вартість і складність впровадження	Орієнтир для правил замін і публікації, але не для повного копіювання.
aSc Timetables	Timetable	Автоматична генерація, перевірка конфліктів	Менше управлінської аналітики	Орієнтир для автоматичної генерації й перевірки конфліктів.

1.4 Постановка задачі та технічне завдання

На основі аналізу предметної області, існуючих програмних рішень і проєктної специфікації визначено, що система має забезпечувати повний цикл підготовки даних для формування розкладу занять. Вона повинна дозволяти адміністратору вводити робочі дні, часові слоти, викладачів, предмети, студентські групи, теги занять, аудиторії, сесії та правила доступності, причому всі ці сутності мають бути пов’язані між собою відповідно до логіки навчального процесу.

Функціональні вимоги охоплюють CRUD-операції для основних довідників, створення навчальних сесій, фільтрацію, перегляд статистики, призначення

аудиторій, визначення недоступних часових інтервалів, підтримку послідовних, паралельних і неперетинних сесій, а також генерацію підсумкового розкладу. Особливе значення має валідація: система повинна контролювати формати ідентифікаторів, обов’язковість полів, числові значення, місткість аудиторій і логіку поєднання тегів із приміщеннями [21; 22]. Нефункціональні вимоги включають зручність інтерфейсу, швидкість виконання основних операцій, надійність збереження даних, можливість резервного копіювання та простоту супроводу [33; 34]. Технічна реалізація базується на мові C#, платформі .NET Framework, Windows Forms для інтерфейсу та SQL Server/Azure SQL для зберігання даних [10; 14; 15].

Таблиця 1.5

Технічне завдання на розробку застосунку

Компонент	**Вимога**	**Критерій приймання**
Довідник викладачів	CRUD, ранг, факультет, кафедра	EmployeeID відповідає формату, запис зберігається
Довідник предметів	Код, назва, семестр, години	Система відхиляє порожній код або від’ємні години
Студентські групи	Формування GroupID і SubGroupID	Ідентифікатор відповідає шаблону Year.Semester.Program.Group
Аудиторії	Будівля, кімната, тип, місткість	Лабораторні лише до лабораторій, місткість перевіряється
Сесії	Викладач + предмет + тег + група + тривалість	Сесія формується і доступна для фільтрації
Статистика	Підрахунок сутностей і графіки	Показники оновлюються за даними бази
Генерація	Формування розкладу з урахуванням слотів	Відсутність базових конфліктів зайнятості

Висновки до розділу 1

У першому розділі проаналізовано роль інформаційних систем в освітньому процесі та встановлено, що управління розкладом занять є складною інформаційною задачею, яка потребує централізованого зберігання даних, формалізації обмежень і підтримки аналітичного контролю. Визначено, що розклад має розглядатися не як проста таблиця, а як система взаємопов'язаних навчальних подій, де кожна подія залежить від викладача, групи, аудиторії, предмета, типу заняття, тривалості та часового слоту.

Досліджено існуючі програмні рішення класів LMS, SIS, EMIS і timetable-систем. Показано, що готові системи мають сильні сторони, однак для навчально-практичної мети доцільною є розробка власного застосунку, який демонструє внутрішню логіку побудови розкладу. Запропоновано технічне завдання, що охоплює довідники, сесії, аудиторії, часові слоти, недоступність, статистику, генерацію та валідацію даних.

РОЗДІЛ 2

ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО ЗАСТОСУНКУ

2.1 Аналіз вимог і специфікації

Надана проєктна специфікація описує систему для ABC institute, який потребує настільного застосунку для управління розкладом. Вимоги подано поетапно: спочатку базові довідники, потім сесії, далі правила взаємного розміщення занять, призначення аудиторій, недоступний час і формування підсумкового розкладу. Така структура є логічною, адже неможливо якісно генерувати розклад, якщо спочатку не визначено, хто, що, де і коли може проводити [46].

Перший блок специфікації включає робочі дні та години, викладачів, предмети, студентів, теги та локації. Другий блок стосується сесій: сесія є центральною одиницею планування, вона поєднує викладача, предмет, тег, групу або підгрупу, кількість студентів і тривалість. Третій блок вводить правила: послідовні заняття, паралельні заняття та заняття, що не повинні накладатися. Четвертий блок присвячений аудиторіям: система дозволяє призначати приміщення для предмета, тегу, викладача, групи, підгрупи, сесії та послідовних занять. П’ятий блок визначає недоступний час для викладачів, груп, підгруп, сесій і кімнат, що є практичною вимогою, адже розклад без механізму недоступності стає нежиттєздатним [46].

Таблиця 2.1.

Витяг ключових вимог зі специфікації

Блок	Сутності	Основні операції	Призначення
Working days/hours	Робочі дні, години, слоти	Додавання, редагування	Формування часової сітки
Lecturers	Викладачі, рівні, ранги	CRUD, активні години	Облік кадрового ресурсу
Subjects	Предмети, коди, години	CRUD, перегляд	Опис навчальних дисциплін
Students	Рік, семестр, програма, група	Генерація ID, CRUD	Формування контингенту
Locations	Будівлі, кімнати, тип, місткість	CRUD, перегляд	Керування аудиторним фондом
Sessions	Викладач + предмет + група + тег	Додавання, фільтри	Центральна одиниця розкладу
Constraints	Consecutive, parallel, not overlapping	Призначення правил	Контроль логіки занять
Not available time	Викладачі, групи, кімнати	Встановлення недоступності	Запобігання конфліктам

2.2 Архітектура системи та компоненти

Архітектура розробленого застосунку є типовою для настільних інформаційних систем на базі Windows Forms. Вона складається з інтерфейсного шару, шару обробки подій і логіки форм, шару доступу до даних та реляційної бази даних. Такий підхід є достатнім для навчального проєкту й дозволяє швидко реалізувати робочий функціонал.

Інтерфейсний шар представлений формами та користувацькими елементами. У структурі архіву наявні модулі Homepage, SplashScreen, DaysHours, Lectures, Subjects, Students, Tags, Locations, Rooms, Sessions, Statistics, Generate, що свідчить про модульний поділ за функціональними напрямами. Шар логіки реалізовано переважно безпосередньо у класах форм: це поширений підхід для WinForms-проєктів, де обробники подій кнопок відкривають підключення до бази, виконують

SQL-запити й оновлюють DataGridView. Перевага такого підходу – простота; недолік – змішування UI та доступу до даних, що ускладнює тестування і супровід [30; 31]. Шар доступу до даних використовує ADO.NET: SqlConnection, SqlCommand, SqlDataAdapter, DataTable [12; 13]. У частині класів помітно винесення рядка підключення до App.config через ConfigurationManager, що є правильною практикою з точки зору безпеки й супроводу [37; 38].

Рисунок 2.1. Логічна архітектура застосунку управління розкладом

Таблиця 2.2.

Архітектурні компоненти застосунку

Компонент	Реалізація в проєкті	Відповідальність
Presentation Layer	Windows Forms, UserControl, DataGridView	Відображення форм і взаємодія з користувачем
Event Logic	Обробники Click, Load, Paint	Реакція на дії користувача
Data Access	ADO.NET, SqlConnection, SqlCommand	Виконання SQL-запитів
Database	SQL Server / Azure SQL	Зберігання даних довідників і сесій
Reporting	Charting	Побудова статистичних графіків
Navigation	Homepage та переходи між формами	Організація сценаріїв роботи

2.3 Проєктування бази даних

Модель даних є фундаментом системи розкладу. Якщо сутності визначені неточно, жоден інтерфейс не врятує систему від плутанини. Ключовими сутностями є Lecturers, Subjects, StudentGroups, Tags, Rooms, TimeSlots, Sessions, NotAvailableTime, ConsecutiveSessions, ParallelSessions та SessionRooms. Вони відображають реальні об’єкти освітнього процесу й забезпечують основу для контролю конфліктів.

Таблиця викладачів містить ідентифікатор, ім’я, EmployeeID, факультет, кафедру, кампус, будівлю, рівень і ранг. Таблиця предметів описує навчальну дисципліну: рік, семестр, назву, код і кількість годин різних типів занять. Студентські групи мають складні ідентифікатори: GroupID формується за шаблоном Year.Semester.Programme.GroupNumber, а SubGroupID додає номер підгрупи, що з одного рядка дозволяє визначити курс, семестр, програму і групу. Аудиторії описуються будівлею, номером, типом і місткістю, причому тип є важливим для відповідності тегу заняття [26; 27]. Сесія є центральною таблицею, яка перетворює

довідники на конкретні навчальні події, пов’язуючи викладача, предмет, тег, групу або підгрупу, кількість студентів і тривалість.

Таблиця 2.3.

Проєкт таблиць бази даних

Таблиця	Ключові поля	Призначення
Lecturers	Id, Name, EmployeeId, Faculty, Department, Level, Rank	Зберігання даних про викладачів
Subjects	Id, OfferedYear, Semester, SubjectName, SubjectCode, LectureHours, TutorialHours, LabHours	Опис навчальних дисциплін
StudentGroups	Id, AcademicYearSemester, Programme, GroupNo, SubGroupNo, GroupId, SubGroupId	Облік груп і підгруп
Tags	Id, TagName, TagCode, RelatedTag	Типізація занять
Rooms	Id, Building, RoomName, RoomType, Capacity	Аудиторний фонд
TimeSlots	Id, Day, StartTime, EndTime	Часова сітка розкладу
Sessions	Id, LecturerId, SubjectId, TagId, GroupId, StudentCount, Duration	Навчальні події
SessionRooms	Id, SessionId, RoomId	Призначення аудиторій
NotAvailableTime	Id, EntityType, EntityId, Day, TimeSlotId	Облік недоступності

2.4 Проєктування користувацького інтерфейсу

Користувацький інтерфейс для системи розкладу має бути не декоративним, а зрозумілим і дисциплінованим. Основний користувач – адміністратор, методист або працівник навчального відділу. Його завдання полягає у швидкому внесенні даних, перевірці таблиці, пошуку помилки та формуванні розкладу, тому інтерфейс має бути простим, передбачуваним і послідовним [33; 34].

У наданій специфікації всі форми мають подібну логіку: область введення даних, кнопки Save/Update/Delete/Clear, таблиця для перегляду записів, інколи фільтри або вкладки. Це правильний шаблон, бо користувач швидко звикає до

однакових сценаріїв, що зменшує кількість помилок і час навчання персоналу. Проєкт WinForms використовує окремі форми для модулів із закругленням панелей через CreateRoundRectRgn, що робить інтерфейс більш сучасним. Використання DataGridView дозволяє швидко переглядати записи й обирати їх для редагування [16]. Особлива увага приділяється формам сесій і генерації: тут інтерфейс підвантажує списки викладачів, тегів, груп і предметів із бази, а не змушує користувача вводити їх вручну, що знижує ризик орфографічних помилок і дублювання.

Таблиця 2.4.

Вимоги до інтерфейсу користувача

Вимога	Пояснення	Очікуваний результат
Єдність форм	Однакова логіка Save/Update/Delete/Clear	Менше помилок користувача
Валідація	Перевірка обов'язкових полів і форматів	Чистіші дані у БД
Підвантаження довідників	ComboBox замість ручного введення	Менше дублювання
Табличний перегляд	DataGridView для списків	Швидке редагування
Статистика	Картки й графіки	Оперативна аналітика
Навігація	Homepage як центральне меню	Зрозумілий сценарій роботи

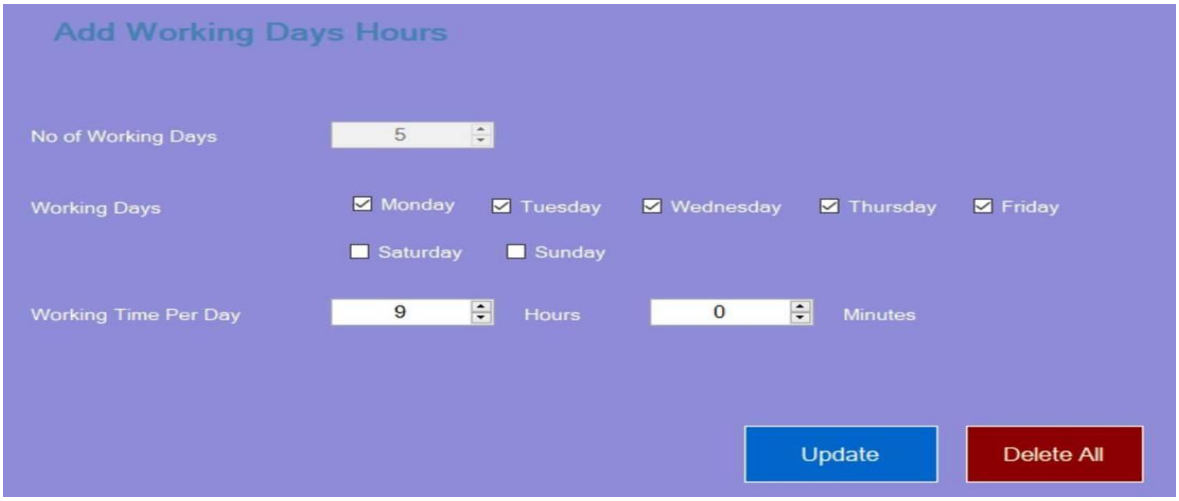

Рисунок 2.3. Приклад інтерфейсного екрана зі специфікації системи

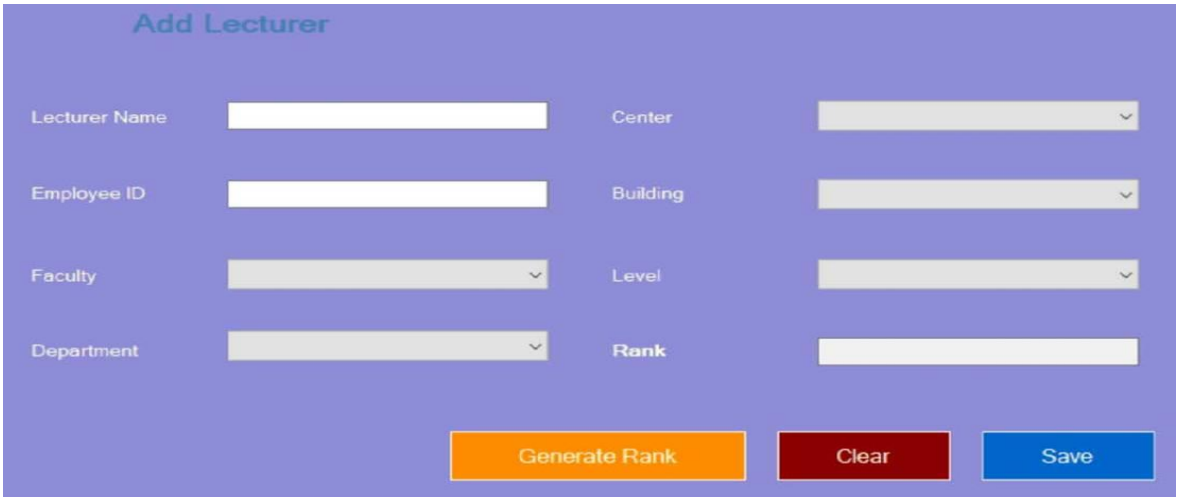

Рисунок 2.4. Приклад інтерфейсного екрана зі специфікації системи

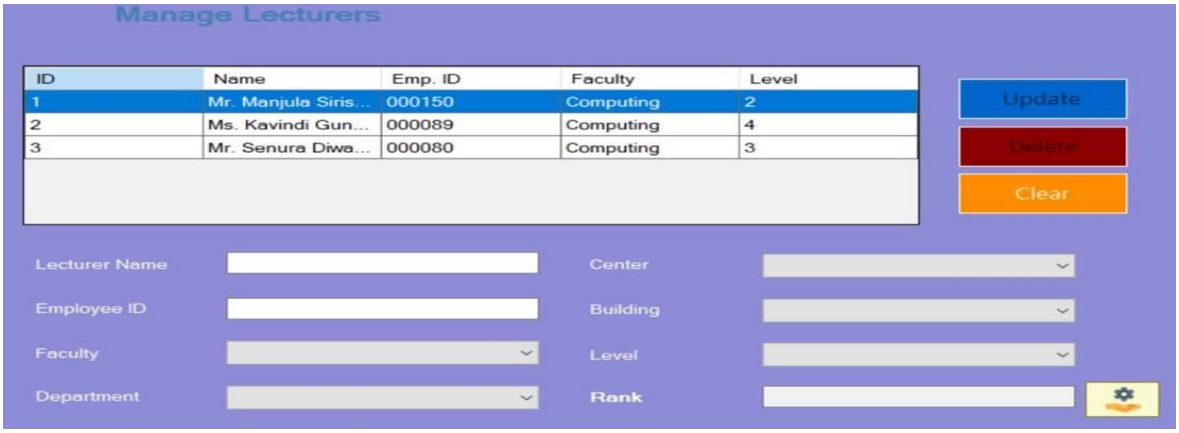

Рисунок 2.5. Приклад інтерфейсного екрана зі специфікації системи

Рисунок 2.6. Приклад інтерфейсного екрана зі специфікації системи

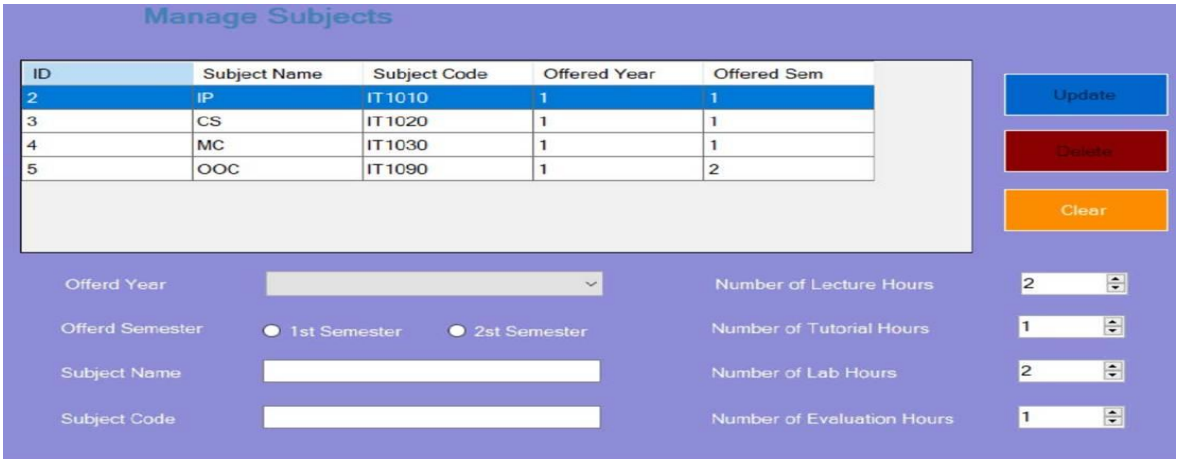

Рисунок 2.7. Приклад інтерфейсного екрана зі специфікації системи

Узагальнену структуру головного вікна та навігації між функціональними модулями показано на рисунку 2.8. Такий підхід забезпечує єдиний сценарій роботи користувача з довідниками, сесіями, статистикою і генерацією розкладу.

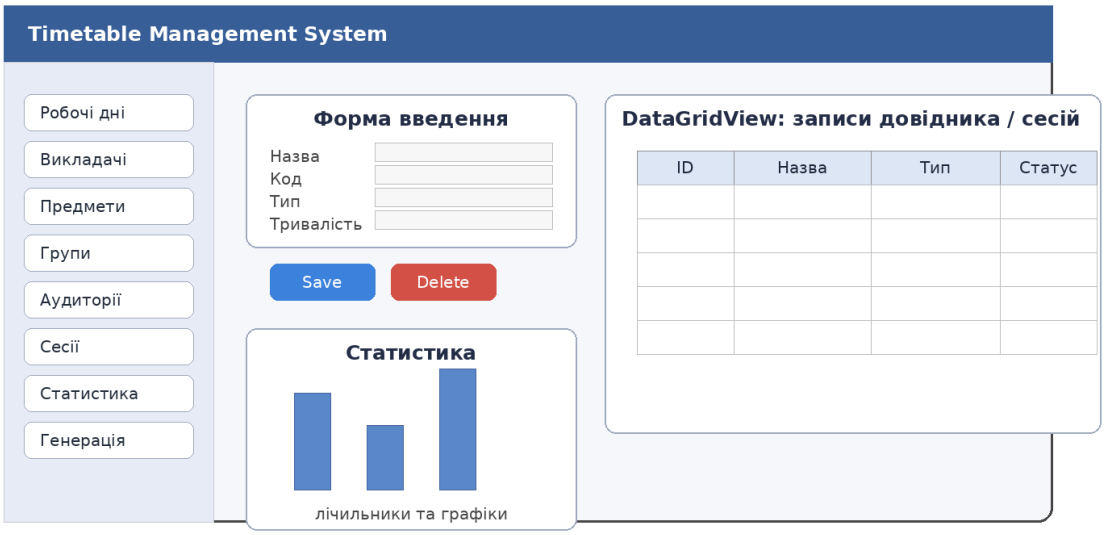

Рисунок 2.8. Структура головного вікна та навігації настільного застосунку

Для перевірки повноти проєктування доцільно зіставити функціональні вимоги з модулями застосунку. Така матриця показує, які частини системи відповідають за окремі сценарії роботи користувача, і допомагає уникнути ситуації, коли вимога описана теоретично, але не має програмної реалізації.

Таблиця 2.5.

Матриця відповідності вимог і модулів застосунку

Функціональна вимога	**Модуль застосунку**	**Дані / таблиці**	**Очікуваний результат**
Налаштування робочого часу	DaysHours, TimeSlot	TimeSlots	Сформована часова сітка для розкладу
Облік викладачів	Lectures	Lecturers	Доступний довідник викладачів із параметрами
Облік дисциплін	Subjects	Subjects	Структурований перелік предметів і годин
Облік груп і підгруп	Students	StudentGroups	Уніфіковані GroupID та SubGroupID
Створення сесій	Sessions	Sessions	Навчальна подія пов’язує викладача, предмет, групу й тег
Призначення аудиторій	Rooms, Locations	Rooms, SessionRooms	Сесія має допустиме приміщення
Аналітика	Statistics	агреговані запити SQL	Користувач бачить кількість сутностей і стан бази
Генерація розкладу	Generate	Sessions, TimeSlots, Rooms	Побудова підсумкового варіанта розкладу

Матриця підтверджує, що основні вимоги мають відповідні модулі реалізації. Найбільш критичними для практичного використання є модулі сесій, аудиторій і генерації, оскільки саме вони забезпечують узгодження довідникових даних у реальний розклад.

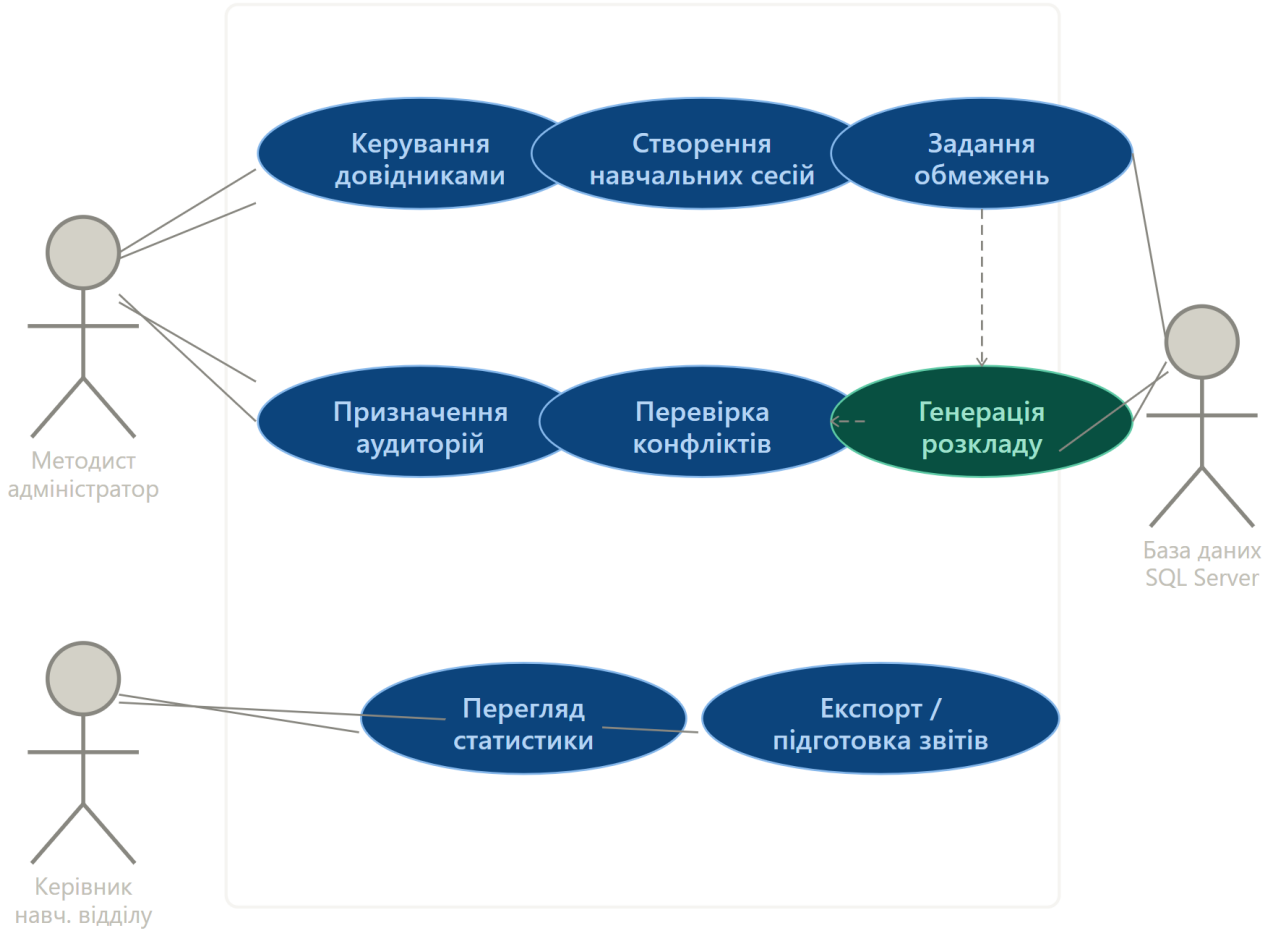

Рисунок 2.9. UML-діаграма прецедентів використання інформаційного застосунку

На UML-діаграмі показано основні сценарії роботи адміністратора або методиста: керування довідниками, створення навчальних сесій, задання обмежень, перевірка конфліктів, генерація розкладу та перегляд статистики. База даних виступає зовнішнім компонентом, з яким взаємодіють усі модулі, що виконують збереження або отримання даних.

2.5 Забезпечення якості, безпеки та супроводу

Якість інформаційної системи залежить не лише від кількості реалізованих форм. Для застосунку управління розкладом важливими є стабільність, захищеність доступу до даних, коректна обробка помилок і можливість подальшого супроводу. Безпека в розглянутому проєкті має окреме значення через використання

підключення до SQL Server/Azure SQL: у вихідному коді виявлено рядки підключення з чутливими даними, тому рекомендовано їх перенесення у захищену конфігурацію, змінні середовища або окремий механізм керування секретами [37; 38]. Супровід програмного забезпечення потребує розділення відповідальностей: доцільно відокремити UI-шар, бізнес-логіку, сервіси валідації та репозиторії доступу до даних, що відповідає принципам чистої архітектури й спрощує майбутню міграцію на веб-платформу [31; 32]. Окремо слід враховувати резервне копіювання, адже втрата бази даних для системи розкладу означає реальний організаційний збій.

Висновки до розділу 2

У другому розділі виконано проєктування інформаційного застосунку на основі наданої специфікації та архіву вихідного коду. Визначено основні модулі системи: довідники, сесії, аудиторії, недоступний час, статистика та генерація розкладу. Центральною одиницею предметної області є навчальна сесія, яка пов’язує викладача, предмет, тег, групу, кількість студентів і тривалість. Запропонована архітектура відповідає настільному застосунку Windows Forms та включає UI-шар, логіку подій, ADO.NET-доступ до даних і реляційну базу. Для подальшого розвитку рекомендовано відокремити бізнес-логіку й доступ до даних від форм, винести рядки підключення у захищену конфігурацію та розширити модель перевірки конфліктів.

РОЗДІЛ 3

РОЗРОБКА ТА ДОСЛІДЖЕННЯ РОБОТИ ЗАСТОСУНКУ

3.1 Реалізація основного функціоналу

Практична реалізація застосунку виконана мовою C# у середовищі Windows Forms. Точка входу розміщена у класі Program, де створюється головний потік STA та запускається SplashScreen. Це стандартна модель для WinForms-застосунків, оскільки інтерфейс працює в одному UI-потоці, а події користувача обробляються через механізм Windows Forms [10; 11].

Функціональність розподілена між окремими каталогами: DaysHours, Lectures, Subjects, Students, Tags, Locations, Rooms, Sessions, Statistics, Generate. Такий поділ відображає предметну область і спрощує навігацію в коді. Для роботи з базою даних використовується ADO.NET: SqlConnection, SqlCommand, SqlDataAdapter, DataTable. Клас TimeSlot, наприклад, містить методи Select та Insert, які виконують SQL-запити до таблиці Time_slots. Це типовий CRUD-патерн: вибрати записи, додати новий, за потреби оновити або видалити.

Лістинг 3.1. Точка входу WinForms-застосунку

```
[STAThread]
static void Main()
{
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new SplashScreen());
}
```

Таблиця 3.1.

Відповідність модулів вихідного коду функціональним вимогам

Модуль	Файли / каталог	Реалізована роль
Запуск системи	Program.cs, SplashScreen.cs	Ініціалізація і старт застосунку
Головне меню	Homepage.cs	Навігація між модулями
Робочі дні та слоти	DaysHours, DayTimeAdpt	Формування часової сітки
Викладачі	Lectures.cs	CRUD і перегляд викладачів
Предмети	Subjects.cs	CRUD дисциплін
Студенти	students.cs	Керування групами та підгрупами
Теги	tags.cs	Типізація занять
Локації та аудиторії	Locations, Rooms	Керування аудиторним фондом
Сесії	Sessions.cs	Зв’язування довідників у навчальні події
Статистика	Statistics.cs	Підрахунок і графіки
Генерація	Generate.cs	Побудова підсумкового розкладу

3.2 Реалізація модулів довідників, сесій і статистики

Модулі довідників реалізують базову адміністративну логіку: додавання, перегляд, редагування і видалення записів. Незважаючи на відносну простоту реалізації, саме вони визначають якість подальшого формування розкладу, адже некоректно заповнені довідники роблять генератор ненадійним.

Модуль студентських груп реалізує генерацію GroupID і SubGroupID. Стандартизований ідентифікатор дозволяє уникати дублювання й неоднозначності, а самі групи та підгрупи використовуються у сесіях, лабораторних заняттях і правилах недоступності. Модуль статистики використовує System.Windows.Forms.DataVisualization.Charting [17]: у конструкторі викликаються методи підрахунку кількості викладачів, студентських груп, предметів, а також будуються діаграми. Це свідчить, що система не лише зберігає дані, а й надає користувачу управлінський огляд стану бази.

3.3 Аналіз фрагментів вихідного коду

Аналіз вихідного коду показав, що проєкт реалізує типовий для Windows Forms підхід: форма відповідає за відображення інтерфейсу, обробку подій і виконання SQL-запитів. Така модель є зрозумілою для навчального проєкту, однак у промисловій розробці потребує рефакторингу. Позитивним аспектом є використання параметризованих SQL-запитів у фрагментах додавання даних, що знижує ризик помилок і частково захищає від SQL-ін’єкцій. Водночас у коді трапляються порожні блоки catch, які приховують помилки, тому для реального використання необхідно додати журналювання та зрозумілі повідомлення користувачу.

Зберігання секретів у коді є порушенням безпечної практики розробки. У дипломному тексті облікові дані підключення не відтворюються, а замість них в рекомендаціях запропоновано перенесення таких параметрів до конфігураційного файлу або захищеного сховища [37; 38].

Лістинг 3.2. Приклад вибірки часових слотів

```
public DataTable Select()
{
    SqlConnection conn = new SqlConnection(mconnstrng);
    DataTable dt2 = new DataTable();
    try
    {
        string sql = "SELECT * FROM Time_slots";
        SqlCommand cmd = new SqlCommand(sql, conn);
        SqlDataAdapter adapter = new SqlDataAdapter(cmd);
        conn.Open();
        adapter.Fill(dt2);
    }
    finally { conn.Close(); }
    return dt2;
}
```

Лістинг 3.3. Приклад додавання часового слоту

```
public bool Insert(TimeSlot s)
{
    bool isSuccess = false;
    SqlConnection conn = new SqlConnection(mconnstrng);
    try
    {
        string sql = "INSERT INTO Time_slots(datepicker, start_time, end_time)
" +
                     "VALUES(@datepicker, @start_time, @end_time)";
        SqlCommand cmd = new SqlCommand(sql, conn);
        cmd.Parameters.AddWithValue("@datepicker", s.datepicker);
        cmd.Parameters.AddWithValue("@start_time", s.start_time);
        cmd.Parameters.AddWithValue("@end_time", s.end_time);
        conn.Open();
        int rows = cmd.ExecuteNonQuery();
        if (rows > 0) isSuccess = true;
    }
    finally { conn.Close(); }
    return isSuccess;
}
```

Лістинг 3.4. Безпечний підхід до отримання рядка підключення

```
static string mconnstrng =
    ConfigurationManager.ConnectionStrings["connstrng"].ConnectionString;
```

3.4 Тестування коректності роботи застосунку

Тестування інформаційного застосунку для розкладу охоплює не тільки перевірку кнопок, а й логіку предметної області. Перший рівень тестування – модульний і функціональний: для кожного довідника перевіряється повний сценарій додавання запису, перегляду у таблиці, вибору для редагування, зміни поля, збереження та видалення. Якщо хоча б один крок не працює, модуль не можна вважати готовим. Другий рівень – тестування валідації: EmployeeID має бути шестизначним числом, години предмета не можуть бути від’ємними, місткість

аудиторії – числом, GroupID – відповідати шаблону. Без цих перевірок база швидко заповниться некоректними даними.

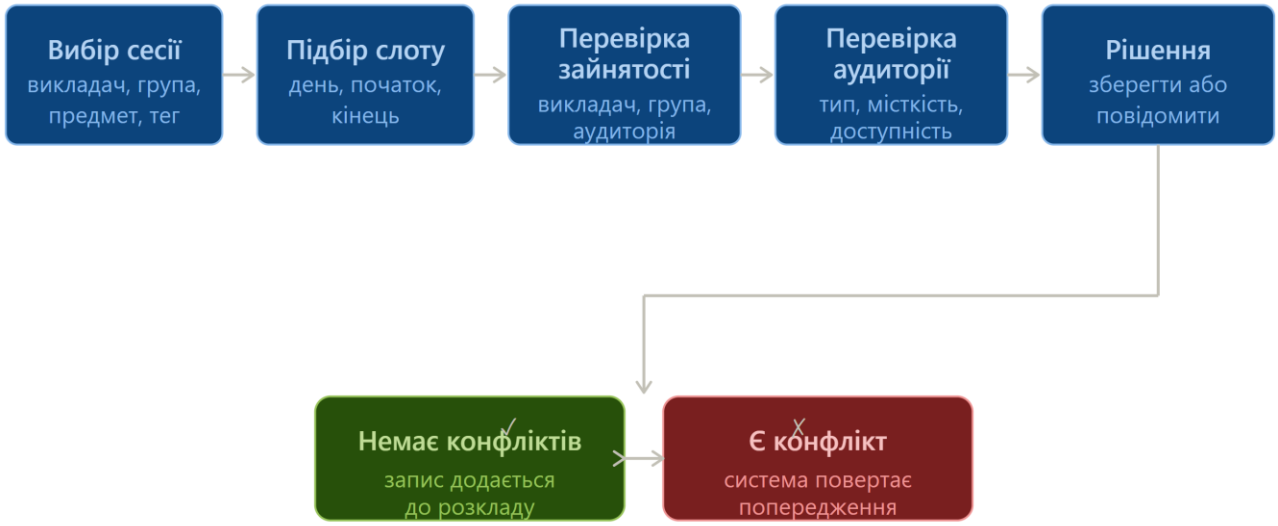

Рисунок 3.1. Алгоритм базової перевірки конфліктів у розкладі

Третій рівень – інтеграційне тестування: сесія має коректно підтягувати викладачів, предмети, теги й групи; видалення предмета не повинно залишати «висячих» посилань; тип і місткість аудиторії при призначенні мають відповідати заняттю. Четвертий рівень – сценарне тестування розкладу: лекція для великої групи, лабораторна для підгрупи, послідовні лекція і практичне, паралельні вибіркові дисципліни, недоступний час викладача та аудиторії. Система має або коректно сформувати запис, або попередити про проблему.

Таблиця 3.2.

Тестові сценарії функціональної перевірки

№	Сценарій	Вхідні дані	Очікуваний результат	Статус
1	Додавання викладача	Name, EmployeeID=000150, Faculty=Computing	Запис з’являється в таблиці	успішно
2	Некоректний EmployeeID	EmployeeID=150	Повідомлення про неправильний формат	потребує валідації
3	Додавання предмета	IT2030, OOC, лекції=2, практики=1	Предмет збережено	успішно
4	Генерація GroupID	Y1, S1, IT, 01	Отримано Y1.S1.IT.01	успішно
5	Створення сесії	Lecturer + Subject + Tag + Group + Duration	Сесія відображається у списку	успішно
6	Призначення лабораторії	Tag=Lab, RoomType=Laboratory	Аудиторія приймається	успішно
7	Лекція у малій кімнаті	120 студентів, capacity=40	Попередження про нестачу місць	потребує доопрацювання
8	Недоступний час викладача	Lecturer unavailable Monday 10:00	Слот не призначається	потребує контролю
9	Оновлення статистики	Додано новий предмет	Лічильник збільшується	успішно
10	Видалення запису	Обраний рядок у DataGridView	Запис видалено після підтвердження	потребує підтвердження

Узагальнення результатів тестових сценаріїв наведено на рисунку 3.2. Діаграма показує, що базові CRUD-операції працюють коректно, а подальшого посилення потребують валідаційні та конфліктні перевірки.

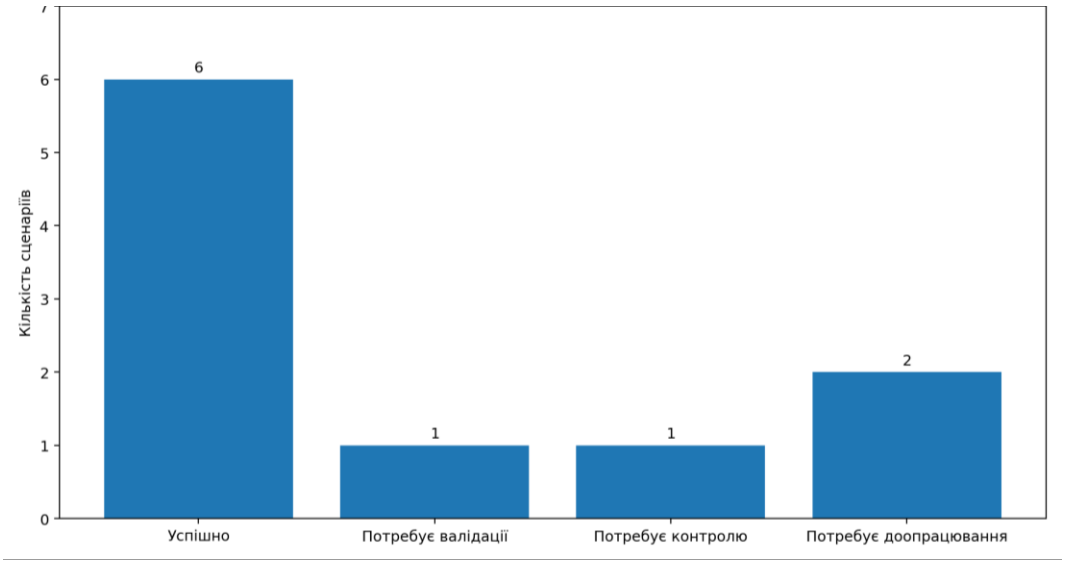

Рисунок 3.2. Результати функціональних тестових сценаріїв

Для додаткової наочності результати тестування доцільно подати не лише у вигляді таблиці сценаріїв, а й як оцінку покриття ключових модулів. Це дозволяє побачити, які частини системи вже перевірені достатньо, а які залишаються пріоритетними для подальшого доопрацювання.

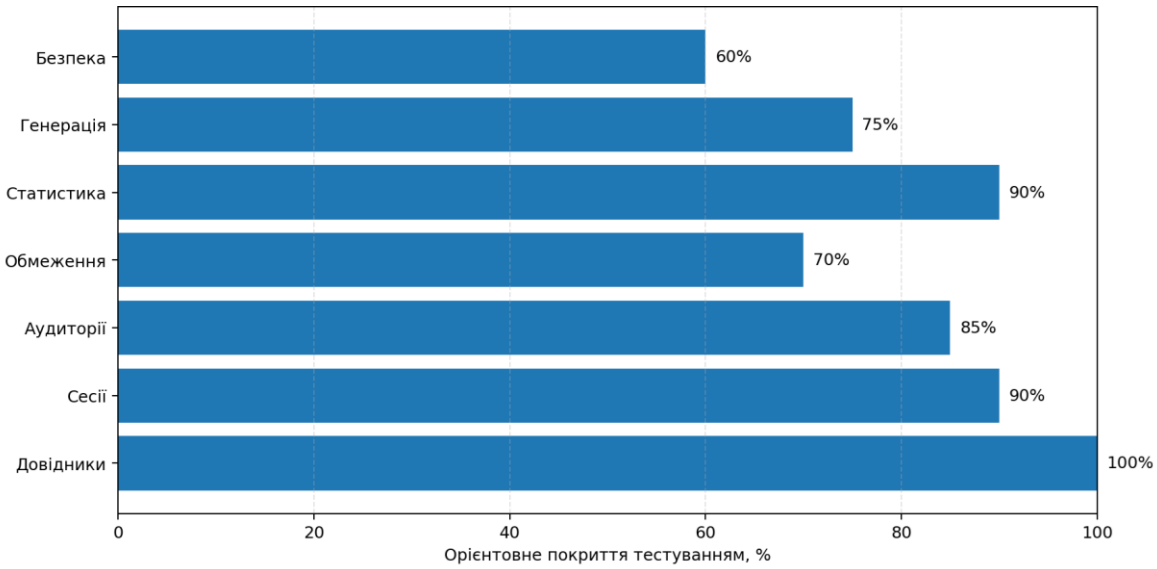

Рисунок 3.3. Орієнтовне покриття функціональних модулів тестуванням

З діаграми видно, що найкраще перевірені довідникові модулі та статистика, оскільки вони мають прямі сценарії додавання, перегляду й оновлення даних. Менш

повним залишається покриття складних сценаріїв генерації, недоступного часу та безпеки, тому саме ці напрями варто посилити у наступних версіях системи.

3.5 Аналіз готовності системи до практичного впровадження

Готовність інформаційного застосунку до практичного використання не можна оцінювати лише за наявністю працездатних форм і підключення до бази даних. У системах управління розкладом важливою є здатність витримувати реальні адміністративні сценарії: часті зміни даних, помилки введення, коригування аудиторій, перенесення занять, відсутність викладачів і потреба оперативного оновлення інформації. Тому оцінювання готовності має враховувати не тільки програмну реалізацію, а й організаційну логіку використання системи.

Повнота довідникової бази є першим критерієм готовності: кожна довідкова сутність має подальше призначення у сесіях, тому поверхнева форма без перевірки ідентифікатора та ключових атрибутів не гарантує якісних даних. Керованість змін також є критичною – у реальному процесі викладач може змінити доступний час, аудиторія стати тимчасово недоступною, група перейти на інший формат навчання. Система має дозволяти такі зміни без ручного втручання у базу і без порушення зв’язків між таблицями. Зрозумілість сценарію роботи для нетехнічного користувача, якість повідомлень про помилки та захист від випадкового або несанкціонованого редагування завершують картину вимог до реальної експлуатації.

Супровід і тестованість також потребують уваги: бізнес-логіка, вбудована у кнопки форм, складно перевіряється автоматизованими тестами. Після її винесення у сервіси можна окремо тестувати перевірку конфліктів викладача, аудиторії, групи, недоступного часу і місткості приміщення. Питання інтеграції із зовнішніми системами та методичної якості розкладу (м’які обмеження) виходять за рамки навчального прототипу, але мають бути врахованими у реальному продукті. Загалом готовність системи можна охарактеризувати як базову навчально-

практичну: вона вже має основні модулі, працює з базою даних і відображає логіку предметної області, але потребує підсилення на рівні валідації, безпеки, архітектурного розділення та контролю конфліктів.

Таблиця 3.3.

Критерії готовності системи до впровадження

Критерій	Поточний стан	Необхідне доопрацювання
Довідникова база	Реалізовано основні сутності	Посилити перевірку форматів і дублювання
Керування змінами	Є CRUD-операції	Додати журналювання змін
Валідація	Частково реалізована	Уніфікувати правила для всіх форм
Безпека	Є підключення до БД	Винести секрети з коду
Супровід	Модулі розділені каталогами	Винести DAL і сервіси
Тестування	Описано сценарії	Додати автоматизовані тести
Масштабованість	Локальний WinForms-додаток	Передбачити API або веб-версію

3.6 Обґрунтування проєктних рішень

Вибір настільної архітектури для цього проєкту є виправданим з огляду на навчальну мету, обсяг функціоналу та очікуваний сценарій використання. Університетські підрозділи часто потребують не складної хмарної системи, а інструменту, який швидко запускається, працює у знайомому середовищі Windows і дозволяє адміністратору виконувати базові операції без залучення команди DevOps. Windows Forms не є новою технологією, однак дає стабільний результат при розробці великої кількості форм, таблиць і полів введення. Вона дозволяє швидко підключити DataGridView, ComboBox, Chart та інші стандартні компоненти. Використання C# обґрунтовано сильною типізацією, зручною інтеграцією з .NET і базами даних Microsoft, а також поширеністю мови у навчальних програмах спеціальності [10; 11].

Вибір реляційної бази даних SQL Server/Azure SQL пов’язаний із природою предметної області: розклад складається з таблиць і зв’язків, де реляційна модель

дозволяє чітко визначати первинні ключі, зовнішні ключі й обмеження цілісності [26; 27]. Розділення на модулі відповідає реальному порядку роботи та дозволяє поступово перевіряти правильність даних. Взаємна узгодженість проєктних рішень щодо таблиць і форм є ключовою: якщо для аудиторії в базі передбачено тип приміщення, форма має надавати вибір зі списку, а GroupID краще генерувати автоматично. Оцінюючи проєкт загалом, його сильна сторона полягає у предметній повноті, а головний резерв покращення – архітектурне впорядкування.

Таблиця 3.4.

Обґрунтування ключових проєктних рішень

Рішення	**Причина вибору**	**Ризик**	**Компенсаційний захід**
Windows Forms	Швидка розробка настільного UI	Обмежена масштабованість	Надалі винести логіку в окремі сервіси
C#	Сильна типізація, підтримка .NET	Залежність від Windows-екосистеми	Передбачити майбутній API
SQL Server/Azure SQL	Реляційна структура предметної області	Потреба у налаштуванні доступу	Використати конфігурацію та резервні копії
ADO.NET	Прямий і зрозумілий доступ до БД	Дублювання SQL у формах	Створити репозиторії
DataGridView	Зручний табличний перегляд	Можливе перевантаження даними	Додати фільтри та пошук
Charting	Оперативна статистика	Потреба у підписах і легендах	Стандартизувати графіки

3.7 Програма тестування та супроводу системи

Розширена програма тестування має охоплювати не лише стандартні дії користувача, а й помилкові, граничні та конфліктні ситуації. Тестування довідників передбачає набір позитивних і негативних сценаріїв: додавання коректного запису, спроба додати порожній, введення дубліката, редагування, видалення запису, що

вже використовується у сесії. Особливу увагу приділяють формату ідентифікаторів: EmployeeID є шестизначним числом, GroupID і SubGroupID формуються за визначеним шаблоном. Тестування сесій охоплює коректність підтягування списків викладачів, предметів, тегів і груп; неможливість створити сесію без обов'язкових полів; правильне формування текстового формату сесії.

Тестування аудиторного фонду перевіряє відповідність типу заняття типу приміщення і місткості, тоді як часові конфлікти – перетин викладача, студентської групи, аудиторії та використання недоступного слоту. Сценарне тестування моделює реальний навчальний тиждень: кілька викладачів, груп, аудиторій різного типу, лекції, практичні, лабораторні, недоступний час і послідовні заняття. Тестування з позиції користувацького досвіду перевіряє, чи може методист швидко знайти потрібну кнопку, створити сесію та повернутися на головний екран. Тестування стабільності охоплює роботу при відсутності підключення до бази, некоректному рядку підключення та великій кількості записів. Статистичні модулі перевіряються на коректне оновлення після змін у довідниках і коректну поведінку при порожній базі.

Таблиця 3.5.

Програма тестування системи

Етап	**Що перевіряється**	**Очікуваний результат**
Довідники	Додавання, редагування, видалення, дублікати	Дані зберігаються коректно, помилки блокуються
Ідентифікатори	EmployeeID, GroupID, SubjectCode	Формати відповідають правилам
Сесії	Зв'язок викладача, предмета, тегу, групи	Сесія формується без втрати даних
Аудиторії	Тип і місткість приміщення	Неможливі непридатні призначення
Часові конфлікти	Викладач, група, аудиторія, слот	Система виявляє перетини
Статистика	Лічильники і графіки	Показники оновлюються після змін
Безпека	Підключення, помилки, секрети	Службові дані не розкриваються
Супровід	Структура коду і повторюваність логіки	Модулі можна змінювати без хаосу

3.8 Ризики, обмеження та напрями подальшого розвитку системи

Будь-яка інформаційна система має не лише функціональні можливості, а й обмеження. Для дипломного проєкту важливо не приховувати ці обмеження, а чітко їх визначити, оскільки це демонструє зрілість інженерного аналізу. Розроблений додаток виконує основні операції з даними розкладу, але як навчально-практичний прототип не є повністю завершеною промисловою системою: його цінність полягає у правильній постановці предметної області, дослідженні структури даних та демонстрації ключових модулів.

Ключовим ризиком є некоректні вхідні дані: помилка у коді предмета, дублювання групи або невірна місткість аудиторії переходять до наступних етапів і виявляються вже під час генерації розкладу. Головним напрямом розвитку тому є посилення валідації на рівні кожної форми. Архітектурна залежність логіки від форм ускладнює підтримку: зміна структури таблиці чи правила перевірки вимагає редагування кількох форм, тому подальший розвиток передбачає створення окремого шару доступу до даних і шару бізнес-логіки. Недостатній контроль конфліктів потребує програмної реалізації перевірок зайнятості, місткості аудиторії та відповідності типу приміщення. Безпека підключення до бази даних є найбільш гострим ризиком: прямі рядки підключення у вихідному коді – типова, але небезпечна практика, яку необхідно усунути. Відсутність розмежування прав доступу та обмежена масштабованість настільного підходу є додатковими напрямами розвитку.

Перспективними напрямами є також автоматизована генерація розкладу з урахуванням пріоритетів (евристичний алгоритм, що спочатку розміщує найбільш обмежені сесії), покращення інтерфейсу (пошук, фільтри, кольорове позначення конфліктів, підтвердження видалення), імпорт і експорт даних у форматі Excel та CSV, веб-публікування розкладу.

Таблиця 3.6.

Основні ризики та напрями їх усунення

Ризик	Можливий наслідок	Напрям усунення
Некоректні вхідні дані	Помилки у сесіях і розкладі	Посилити валідацію
Логіка у формах	Складність супроводу	Винести бізнес-логіку в сервіси
Недостатній контроль конфліктів	Накладання занять	Реалізувати алгоритми перевірки
Секрети у коді	Ризик доступу до БД	Використати захищену конфігурацію
Відсутність ролей	Випадкове редагування даних	Додати рольову модель
Локальна архітектура	Складність масштабування	Розглянути API або веб-версію
Обмежена аналітика	Менше управлінської користі	Додати звіти та показники

3.9 Практичні рекомендації щодо використання результатів роботи

Результати дипломної роботи можуть використовуватись не лише як опис конкретного Windows Forms-застосунку, а й як методична основа для створення подібних локальних систем у навчальних підрозділах. Насамперед це стосується закладів, де розклад досі формується в електронних таблицях без будь-якої перевірки узгодженості. Навіть базова централізація даних здатна суттєво зменшити кількість організаційних помилок.

Впровадження рекомендовано виконувати поетапно: спочатку один підрозділ або один навчальний семестр, перевірка правильності довідників, тестування створення сесій, і лише після цього масштабування рішення. Перед внесенням інформації до системи варто провести нормалізацію даних: узгодити назви дисциплін, коди предметів, ідентифікатори груп, перелік аудиторій, типи занять і правила скорочень. Без цього система зберігатиме різні варіанти одного об’єкта, що ускладнить фільтрацію, статистику і генерацію. Призначення відповідальної особи за якість довідників, регулярна перевірка статистики та документування правил роботи є організаційними передумовами стабільного використання.

Результати роботи можуть стати основою для подальших навчальних проєктів: на її базі можна реалізувати автоматичний генератор розкладу, веб-панель для студентів, API для публікації змін, модуль повідомлень, імпорт із Excel або систему ролей. Практична значущість роботи полягає у демонстрації повного циклу створення інформаційної системи: від аналізу проблеми до проєктування, реалізації, тестування й оцінки ефективності.

На основі проведеного аналізу сформовано перелік функціональних і технічних елементів, які доцільно додати або посилити перед практичним впровадженням системи. Ці пункти не знижують цінності навчального прототипу, але визначають дорожню карту його перетворення на більш зріле прикладне рішення.

Таблиця 3.7

Елементи, які потребують додаткового доопрацювання

Напрям	**Поточний стан**	**Що потрібно додати**	**Очікуваний ефект**
Валідація	Частково реалізована у формах	Єдині правила перевірки для всіх довідників і сесій	Менше помилок у базі даних
Перевірка конфліктів	Описана як базовий сценарій	Автоматичне виявлення перетину викладача, групи, аудиторії й часу	Розклад стане практично коректнішим
Журналювання	Помилки частково обробляються	Лог-файл або таблиця SystemLog	Простіше виявляти причини збоїв
Безпека підключення	Рядок підключення винесено частково	Захищене зберігання параметрів і мінімальні права користувача БД	Зменшення ризику витоку доступів
Імпорт / експорт	Не є центральною функцією	Експорт розкладу в Excel/PDF та імпорт довідників	Зручніше впровадження в реальному підрозділі
Ролі користувачів	Орієнтація на адміністратора	Ролі адміністратор, методист, перегляд	Контроль доступу до редагування
Резервне копіювання	Описано як рекомендацію	Автоматичне створення резервної копії БД	Захист від втрати даних

Запропоновані доопрацювання формують логічну послідовність розвитку: спочатку варто посилити валідацію та перевірку конфліктів, після цього реалізувати журналювання і резервне копіювання, а вже потім розширювати систему імпортом, експортом і ролями користувачів.

Висновки до розділу 3

У третьому розділі розглянуто практичну реалізацію програмного продукту для формування та управління розкладом занять. Встановлено, що застосунок реалізовано у середовищі Windows Forms мовою C# із використанням ADO.NET для взаємодії з базою даних SQL Server/Azure SQL, що забезпечує можливість створення настільного рішення з графічним інтерфейсом і збереженням даних у реляційній базі. Структура програмного продукту охоплює всі основні предметні напрями: робочі дні, викладачі, навчальні дисципліни, академічні групи, теги, аудиторії, сесії, статистика та генерація розкладу. Практична реалізація підтверджує, що додаток виконує основні функції, передбачені темою роботи, і може розглядатися як завершений навчально-практичний прототип. Разом із тим визначено напрями подальшого вдосконалення: посилення валідації, покращення обробки помилок, підвищення рівня безпеки, архітектурне розділення відповідальності та удосконалення механізмів контролю конфліктів.

ВИСНОВКИ

Проаналізовано предметну область управління розкладом занять у закладі освіти та встановлено, що розклад є центральним організаційним механізмом освітнього процесу. Він поєднує навчальні плани, викладачів, студентські групи, аудиторії, часові слоти, типи занять і правила доступності. Ручне складання або використання розрізнених таблиць призводить до дублювання даних, конфліктів і зростання навантаження на адміністративний персонал.

Визначено, що для ефективного управління розкладом необхідна централізована інформаційна система, яка підтримує довідники, навчальні сесії, аудиторний фонд, часові інтервали, недоступний час, статистику та перевірку базових обмежень. На основі аналізу специфікації сформовано технічне завдання, уточнено функціональні й нефункціональні вимоги та приведено завдання роботи до чотирьох логічно узгоджених пунктів.

Досліджено існуючі програмні рішення для цифровізації освіти: LMS-платформи, SIS-системи, спеціалізовані timetable-системи та EMIS-рішення. Показано, що Moodle, Google Classroom і Canvas зручні для організації навчального контенту, openSIS та АС «Деканат» орієнтовані на адміністративний облік, а Untis і aSc Timetables найкраще відповідають задачі розкладу. Разом із тим власна розробка є доцільною як навчально-практичний інструмент, що дозволяє дослідити внутрішню логіку побудови системи.

Запропоновано архітектуру інформаційного застосунку, модель бази даних, структуру основних модулів і вимоги до інтерфейсу користувача. Центральною сутністю визначено навчальну сесію, яка пов’язує викладача, предмет, тег, групу або підгрупу, кількість студентів і тривалість заняття. На цій основі побудовано логіку подальшого призначення аудиторій, контролю доступності та формування розкладу.

Окремі фрагменти коду використано для демонстрації реалізації базових операцій роботи з даними.

Система забезпечує виконання основних функцій управління даними розкладу, однак потребує подальшого вдосконалення механізмів автоматичної перевірки конфліктів, журналювання помилок та захищеного зберігання параметрів підключення.

Отримані результати відповідають предметній області спеціальності 122 «Комп’ютерні науки», оскільки поєднують аналіз предметної області, проєктування бази даних, програмну реалізацію, тестування та оцінювання якості програмного рішення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Moodle Documentation. MoodleDocs. URL: https://docs.moodle.org/ (дата звернення: 01.05.2026).

2. Moodle LMS. Moodle. URL: https://moodle.com/products/lms/ (дата звернення: 01.05.2026).

3. Moodle Open Source. Moodle. URL: https://moodle.com/about/open-source/ (дата звернення: 01.05.2026).

4. Google Classroom. Google for Education. URL: https://edu.google.com/workspace-for-education/products/classroom/ (дата звернення: 01.05.2026).

5. Classroom Help. Google Support. URL: https://support.google.com/edu/classroom/ (дата звернення: 01.05.2026).

6. Canvas LMS. Instructure. URL: https://www.instructure.com/canvas (дата звернення: 01.05.2026).

7. What is Canvas? Instructure Community. URL: https://community.instructure.com/ (дата звернення: 01.05.2026).

8. openSIS Student Information System Features. URL: https://www.opensis.com/features (дата звернення: 01.05.2026).

9. openSIS Student Information & School Management Software. URL: https://www.opensis.com/ (дата звернення: 01.05.2026).

10. Microsoft Learn. Windows Forms documentation. URL: https://learn.microsoft.com/dotnet/desktop/winforms/ (дата звернення: 01.05.2026).

11. Microsoft Learn. C# documentation. URL: https://learn.microsoft.com/dotnet/csharp/ (дата звернення: 01.05.2026).

12. Microsoft Learn. ADO.NET overview. URL: https://learn.microsoft.com/dotnet/framework/data/adonet/ (дата звернення: 01.05.2026).

13. Microsoft Learn. SqlConnection Class. URL: https://learn.microsoft.com/dotnet/api/system.data.sqlclient.sqlconnection (дата звернення: 01.05.2026).

14. Microsoft Learn. SQL Server documentation. URL: https://learn.microsoft.com/sql/sql-server/ (дата звернення: 01.05.2026).

15. Microsoft Learn. Azure SQL Database documentation. URL: https://learn.microsoft.com/azure/azure-sql/database/ (дата звернення: 01.05.2026).

16. Microsoft Learn. DataGridView Control. URL: https://learn.microsoft.com/dotnet/desktop/winforms/controls/datagridview-control-windows-forms (дата звернення: 01.05.2026).

17. Microsoft Learn. Chart Controls in Windows Forms. URL: https://learn.microsoft.com/dotnet/api/system.windows.forms.datavisualization.charting.chart (дата звернення: 01.05.2026).

18. UNESCO. Education Management Information Systems (EMIS). URL: https://www.unesco.org/ (дата звернення: 01.05.2026).

19. UNESCO Institute for Statistics. Education data and indicators. URL: https://uis.unesco.org/ (дата звернення: 01.05.2026).

20. European Commission. Digital Education Action Plan. URL: https://education.ec.europa.eu/focus-topics/digital-education/action-plan (дата звернення: 01.05.2026).

21. ISO/IEC/IEEE 29148:2018. Systems and software engineering – Life cycle processes – Requirements engineering. ISO, 2018.

22. ISO/IEC 25010:2011. Systems and software engineering – Systems and software Quality Requirements and Evaluation. ISO, 2011.

23. Sommerville I. Software Engineering. 10th ed. Pearson, 2016.

24. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner’s Approach. 9th ed. McGraw-Hill, 2019.

25. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed. Addison-Wesley, 2003.

26. Date C. J. An Introduction to Database Systems. 8th ed. Pearson, 2003.

27. Connolly T., Begg C. Database Systems: A Practical Approach to Design, Implementation, and Management. 6th ed. Pearson, 2015.

28. Elmasri R., Navathe S. Fundamentals of Database Systems. 7th ed. Pearson, 2016.

29. Silberschatz A., Korth H., Sudarshan S. Database System Concepts. 7th ed. McGraw-Hill, 2019.

30. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994.

31. Martin R. C. Clean Architecture: A Craftsman’s Guide to Software Structure and Design. Pearson, 2017.

32. Martin R. C. Clean Code: A Handbook of Agile Software Craftsmanship. Pearson, 2008.

33. Nielsen J. Usability Engineering. Morgan Kaufmann, 1993.

34. Shneiderman B. Designing the User Interface. 6th ed. Pearson, 2016.

35. Krug S. Don’t Make Me Think, Revisited. New Riders, 2014.

36. OWASP. Top 10 Web Application Security Risks. URL: https://owasp.org/www-project-top-ten/ (дата звернення: 01.05.2026).

37. OWASP. Secrets Management Cheat Sheet. URL: https://cheatsheetseries.owasp.org/cheatsheets/Secrets_Management_Cheat_Sheet.html (дата звернення: 01.05.2026).

38. Microsoft Learn. Connection strings and configuration files. URL: https://learn.microsoft.com/dotnet/framework/data/adonet/connection-strings-and-configuration-files (дата звернення: 01.05.2026).

39. aSc Timetables. Official website. URL: https://www.asctimetables.com/ (дата звернення: 01.05.2026).

40. Untis. Timetable software and WebUntis. URL: https://www.untis.at/ (дата звернення: 01.05.2026).

41. Печериця, В. В., Бондарчук, А. П., & Замрій, І. В. (2021). Розробка методики прототипування об'єктів інформаційної системи на базі технології Java Script, Node. JS. Телекомунікаційні та інформаційні технології, (4), 12-19.

42. Bondarchuk, A., Kornaga, Y., Bazaliy, M., Serhiyenko, P., & Ilin, O. (2020). Method of protecting software code from analysis by obfuscation means. Telecommunication and informative technologies, 69(4).

43. Придибайло, О. Б., & Бондарчук, А. П. (2018). Розробка інформаційних технологій для сервісно-орієнтовного проектування інформаційних систем. Сучасний захист інформації, (1), 6-10.

44. Єдина державна електронна база з питань освіти. URL: https://info.edbo.gov.ua/ (дата звернення: 01.05.2026).

45. Міністерство освіти і науки України. URL: https://mon.gov.ua/ (дата звернення: 01.05.2026).

46. Закон України «Про освіту». URL: https://zakon.rada.gov.ua/laws/show/2145-19 (дата звернення: 01.05.2026).

47. Закон України «Про захист персональних даних». URL: https://zakon.rada.gov.ua/laws/show/2297-17 (дата звернення: 01.05.2026).

48. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Київ: ДП «УкрНДНЦ», 2016.

49. Project Specification: Timetable Management System: внутрішня проєктна специфікація ABC institute, використана як вихідний матеріал для аналізу вимог і структури даних інформаційного застосунку.

ДОДАТОК А

Фрагменти вихідного коду застосунку

Лістинги А.1 та А.2 наведено для демонстрації повної реалізації методів класу TimeSlot, що забезпечують базові операції вибірки та вставки даних. Лістинг А.3 ілюструє рекомендований спосіб отримання рядка підключення – з конфігураційного файлу App.config, що є безпечнішою практикою порівняно зі збереженням рядка у відкритому вигляді у вихідному коді [37; 38].

Лістинг А.1. Повна реалізація класу TimeSlot: вибірка записів

```
public DataTable Select()
{
    SqlConnection conn = new SqlConnection(mconnstrng);
    DataTable dt = new DataTable();
    try
    {
        string sql = "SELECT * FROM Time_slots";
        SqlCommand cmd = new SqlCommand(sql, conn);
        SqlDataAdapter adapter = new SqlDataAdapter(cmd);
        conn.Open();
        adapter.Fill(dt);
    }
    finally { conn.Close(); }
    return dt;
}
```

Лістинг А.2. Повна реалізація методу Insert класу TimeSlot

```
public bool Insert(TimeSlot s)
{
    bool isSuccess = false;
    SqlConnection conn = new SqlConnection(mconnstrng);
    try
    {
        string sql = "INSERT INTO Time_slots(datepicker,start_time,end_time)"
+
                     " VALUES(@datepicker,@start_time,@end_time)";
```

```
            SqlCommand cmd = new SqlCommand(sql, conn);
            cmd.Parameters.AddWithValue("@datepicker", s.datepicker);
            cmd.Parameters.AddWithValue("@start_time", s.start_time);
            cmd.Parameters.AddWithValue("@end_time", s.end_time);
            conn.Open();
            int rows = cmd.ExecuteNonQuery();
            if (rows > 0) isSuccess = true;
        }
        finally { conn.Close(); }
        return isSuccess;
    }
```

Лістинг А.3. Отримання рядка підключення з конфігурації

```
    static string mconnstrng =
        ConfigurationManager.ConnectionStrings["connstrng"].ConnectionString;
```

ДОДАТОК Б

SQL-структура основних таблиць бази даних

У застосунку наведено приклад SQL-структури бази даних, яка може використовуватися для збереження основних сутностей системи управління розкладом занять. Структура є узагальненою і може бути адаптована під конкретну

Лістинг Б.1. реалізацію SQL Server або Azure SQL.

```
CREATE TABLE Lecturers (
    Id INT IDENTITY(1,1) PRIMARY KEY,
    FullName NVARCHAR(150) NOT NULL,
    EmployeeId NVARCHAR(20) NOT NULL UNIQUE,
    Faculty NVARCHAR(100) NULL,
    Department NVARCHAR(100) NULL,
    LevelName NVARCHAR(50) NULL,
    RankName NVARCHAR(50) NULL
);

CREATE TABLE Subjects (
    Id INT IDENTITY(1,1) PRIMARY KEY,
    SubjectCode NVARCHAR(30) NOT NULL UNIQUE,
    SubjectName NVARCHAR(150) NOT NULL,
    OfferedYear INT NOT NULL,
    Semester INT NOT NULL,
    LectureHours INT DEFAULT 0,
    TutorialHours INT DEFAULT 0,
    LabHours INT DEFAULT 0
);

CREATE TABLE StudentGroups (
    Id INT IDENTITY(1,1) PRIMARY KEY,
    AcademicYearSemester NVARCHAR(20) NOT NULL,
    Programme NVARCHAR(50) NOT NULL,
    GroupNo NVARCHAR(20) NOT NULL,
    SubGroupNo NVARCHAR(20) NULL,
    GroupId NVARCHAR(80) NOT NULL,
    SubGroupId NVARCHAR(100) NULL
);

CREATE TABLE Tags (
    Id INT IDENTITY(1,1) PRIMARY KEY,
    TagName NVARCHAR(50) NOT NULL,
    TagCode NVARCHAR(20) NOT NULL,
    RelatedTag NVARCHAR(50) NULL
);

CREATE TABLE Rooms (
    Id INT IDENTITY(1,1) PRIMARY KEY,
    Building NVARCHAR(80) NOT NULL,
    RoomName NVARCHAR(50) NOT NULL,
    RoomType NVARCHAR(50) NOT NULL,
    Capacity INT NOT NULL CHECK (Capacity > 0)
```

```
);

CREATE TABLE TimeSlots (
    Id INT IDENTITY(1,1) PRIMARY KEY,
    DayName NVARCHAR(20) NOT NULL,
    StartTime TIME NOT NULL,
    EndTime TIME NOT NULL
);

CREATE TABLE Sessions (
    Id INT IDENTITY(1,1) PRIMARY KEY,
    LecturerId INT NOT NULL FOREIGN KEY REFERENCES Lecturers(Id),
    SubjectId INT NOT NULL FOREIGN KEY REFERENCES Subjects(Id),
    TagId INT NOT NULL FOREIGN KEY REFERENCES Tags(Id),
    GroupId INT NOT NULL FOREIGN KEY REFERENCES StudentGroups(Id),
    StudentCount INT NOT NULL CHECK (StudentCount > 0),
    DurationMinutes INT NOT NULL CHECK (DurationMinutes IN (30, 60, 90, 120))
);

CREATE TABLE SessionRooms (
    Id INT IDENTITY(1,1) PRIMARY KEY,
    SessionId INT NOT NULL FOREIGN KEY REFERENCES Sessions(Id),
    RoomId INT NOT NULL FOREIGN KEY REFERENCES Rooms(Id)
);

CREATE TABLE NotAvailableTime (
    Id INT IDENTITY(1,1) PRIMARY KEY,
    EntityType NVARCHAR(30) NOT NULL,
    EntityId INT NOT NULL,
    DayName NVARCHAR(20) NOT NULL,
          TimeSlotId INT NOT NULL FOREIGN KEY REFERENCES TimeSlots(Id)
);
```