

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

ДЛЯ ОБЛІКУ ВИКЛАДАЦЬКОЇ РОБОТИ

SOFTWARE TO KEEP RECORD

OF TEACHING ACTIVITY

Виконав(ла): студент(ка) 4 курсу, групи

КНТ-112сп

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

НУЖНИЙ О.О.

(ПРІЗВИЩЕ та ініціали)

Керівник

ЗАЙКО Т.А.

(ПРІЗВИЩЕ та ініціали)

Рецензент

КОЦУР М.І.

(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
 “ ” 2025 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

НУЖНОГО Олександра Олександровича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення для обліку викладацької роботи. Software to Keep Record of Teaching Activity

керівник проєкту (роботи) к.т.н., доцент, ЗАЙКО Тетяна Анатоліївна,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “28” квітня 2025 року № 209

2. Строк подання студентом проєкту (роботи) 03 червня 2025 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ЗАЙКО Т.А., доцент		
Нормоконтроль	КАМІНСЬКА Ж.К., асистент		

7. Дата видачі завдання “ 28 ” квітня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Олександр НУЖНИЙ
(Імя ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Тетяна ЗАЙКО
(Імя ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
90 с., 3 табл., 34 рис., 3 дод., 18 джерел.

C#, VISUAL STUDIO, ВИКЛАДАЦЬКА ДІЯЛЬНІСТЬ, МОВА ПРОГРАМУВАННЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ПРОЄКТУВАННЯ.

Об'єкт дослідження – процес розробки програмного забезпечення для обліку викладацької роботи.

Предмет дослідження – програмні засоби для обліку викладацької роботи.

Мета роботи – розробка програмного забезпечення для обліку викладацької роботи.

Матеріали, методи та технічні засоби: мова програмування C#, середовище розробки Visual Studio.

Результати. Розроблено проєктні рішення для створення програмного забезпечення для обліку викладацької роботи. Створено програмне забезпечення для обліку викладацької роботи. Проведено тестування розробленої програмної системи для обліку викладацької роботи.

Висновки. Мету роботи досягнуто. Розроблено програмне забезпечення для обліку викладацької роботи за допомогою мови програмування C# та середовища розробки Visual Studio.

Галузь використання – освітня діяльність.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 90 pages, 3 tables, 34 figures, 3 appendixes, 18 sources.

C#, VISUAL STUDIO, TEACHING ACTIVITY, PROGRAMMING LANGUAGE, SOFTWARE, DESIGN.

The object of research is the process of developing software for accounting for teaching work.

The subject of the research is software for accounting for teaching work.

The purpose of this work is to develop the software for accounting for teaching work.

Materials, methods and technical tools: C# programming language, Visual Studio development environment.

Results. Project solutions for the creation of software for accounting for teaching work has been designed. The software for accounting for teaching work has been developed. The developed software system for accounting for teaching work has been tested.

Conclusions. The goal of the work has been achieved. The software tools for accounting for teaching work using the C# programming language and the Visual Studio development environment have been developed.

Scope of use – educational activities.

ЗМІСТ

	С.
Перелік скорочень та умовних познач	8
Вступ	9
1 Аналіз предметної області	11
1.1 Програмне забезпечення для обліку викладацької роботи	11
1.2 Аналіз програмного забезпечення для обліку викладацької роботи	23
1.3 Висновки за розділом 1	31
2 Матеріали і методи	33
2.1 Вибір мови програмування	33
2.2 Вибір середовища розробки для створення програмного забезпечення для обліку викладацької роботи	36
2.3 Висновки за розділом 2	39
3 Опис програми	41
3.1 Структура програмного забезпечення для обліку викладацької роботи	41
3.2 Функціонування програмного забезпечення для обліку викладацької роботи	43
3.3 Розробка бази даних програмного забезпечення для обліку викладацької роботи	45
3.4 Проєктування інтерфейсу програмного забезпечення для обліку викладацької роботи	47
3.5 Висновки за розділом 3	51
4 Експлуатація, тестування та експериментальне дослідження програми	53
4.1 Призначення й умови застосування програми	53
4.2 Характеристики програми для обліку викладацької роботи	54
4.3 Інструкція по експлуатації програми	55
4.3.1 Звернення до програми	55
4.3.2 Вхідні й вихідні дані	55
4.3.3 Повідомлення	55

4.4 Виконання програмного забезпечення для обліку викладацької роботи	56
4.5 Тестування програмного забезпечення для обліку викладацької роботи	61
4.6 Висновки за розділом 4	61
Висновки	62
Перелік джерел посилань	65
Додаток А Технічне завдання	67
Додаток Б Фрагмент тексту програми	71
Додаток В Слайди презентації	83

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

LMS – Learning Management System;

MVC – Model-View-Controller;

БД – база даних;

ПЗ – програмне забезпечення.

Вступ

Програмне забезпечення для обліку викладацької роботи є надзвичайно актуальним у сучасних умовах розвитку освіти. Його впровадження дозволяє автоматизувати рутинні процеси, пов'язані з розрахунком педагогічного навантаження, плануванням та звітуванням. Завдяки цьому значно зменшується кількість помилок, знижуються часові витрати на підготовку документів і підвищується загальна ефективність роботи як окремих викладачів, так і адміністрації закладу [1]-[3].

Сучасні освітні установи потребують прозорого механізму контролю викладацької діяльності. Програмне забезпечення для обліку викладацької роботи дає можливість оперативно відстежувати обсяг виконаної роботи, аналізувати завантаженість персоналу, приймати обґрунтовані управлінські рішення. Особливо актуальним є використання таких програмних систем в умовах дистанційного та змішаного навчання, де традиційні методи обліку стають менш ефективними або взагалі непридатними [4], [5].

Ще однією важливою перевагою програмного забезпечення (ПЗ) для обліку викладацької роботи є можливість швидкого формування звітів, діаграм і статистичних даних, що необхідні для внутрішнього моніторингу, акредитацій, перевірок з боку органів освіти, а також для самоаналізу закладу. Багато програм підтримують інтеграцію з іншими освітніми платформами, такими як електронні журнали чи системи дистанційного навчання, що дозволяє створити єдиний інформаційний простір [6], [7].

Проте деякі програми для обліку викладацької роботи можуть бути досить складними для налаштування та адаптації системи до специфіки конкретного навчального закладу. Кожен заклад має власні вимоги до обліку видів діяльності та звітності, і далеко не кожне програмне забезпечення здатне гнучко налаштуватися під ці особливості без залучення фахівців або значних витрат часу. Ще одним викликом є потреба в навчанні персоналу, крім того,

інтерфейс деяких програм для обліку викладацької роботи може бути незручним або перевантаженим, що ускладнює роботу користувачів і знижує загальну ефективність використання. З технічного боку, програмне забезпечення для обліку викладацької роботи може мати проблеми з надійністю або сумісністю. У випадку локального встановлення ПЗ для обліку викладацької роботи додатковою складністю є потреба в обслуговуванні та резервному копіюванні, що вимагає технічного персоналу або залучення сторонніх спеціалістів.

Тому актуальною є розробка програмного забезпечення для обліку викладацької роботи. У дипломній кваліфікаційній роботі бакалавра розв'язується актуальне завдання розробки програмного забезпечення для обліку викладацької роботи.

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для обліку викладацької роботи;
- здійснити проєктування програмного забезпечення для обліку викладацької роботи;
- створити програмне забезпечення для обліку викладацької роботи;
- виконати тестування розробленого програмного забезпечення для обліку викладацької роботи.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Програмне забезпечення для обліку викладацької роботи

Розробка програмного забезпечення для освітньої діяльності та обліку викладацької роботи активно змінює традиційні підходи до навчального процесу, забезпечуючи більш індивідуалізоване, інтерактивне та динамічне навчання. Впровадження новітніх технологій сприяє створенню ефективних інструментів для онлайн освіти, а також оптимізації адміністративних функцій. Сучасні програмні засоби дозволяють удосконалювати методи навчання і значно полегшують облік викладацької роботи, завдяки чому вчителі можуть більше зосереджуватись на навчальному процесі, а адміністрація – на керуванні даними [1].

Важливу роль у цьому відіграють програмні інструменти, які допомагають створювати адаптивні навчальні плани та інтегрувати їх у різноманітні навчальні середовища. Сучасні розробники прагнуть створювати програмне забезпечення, яке дозволяє викладачам здійснювати облік результатів студентів, планувати та організовувати заняття, а також здійснювати спостереження за навчальним процесом у реальному часі [1].

Застосування таких технологій дозволяє впроваджувати гнучкі методи навчання, що можуть поєднувати онлайн та традиційні формати, надаючи викладачам можливість забезпечувати якісне навчання для учнів, незалежно від їхнього місцезнаходження. Це дозволяє не лише покращити ефективність навчального процесу, але й забезпечити зручний доступ до навчальних матеріалів [1].

Враховуючи зростаючі вимоги до автоматизації освітніх процесів, програмне забезпечення для обліку викладацької роботи стає важливою складовою для будь-якої освітньої установи. Воно сприяє інтеграції оцінок, відгуків, зворотного зв'язку з учнями та батьками, що допомагає зробити навчання більш прозорим і доступним [1].

Розробка програмного забезпечення для освіти дозволяє навчальним закладам ефективно адаптувати процес онлайн освіти, забезпечуючи вищий рівень залученості учнів. Такий підхід дає змогу значно покращити доступність навчальних матеріалів та сприяє більш гнучкому підходу до самостійного навчання. Системи освітнього програмного забезпечення надають безліч переваг освітнім організаціям, допомагаючи вдосконалювати процеси навчання та управління [1].

Однією з головних переваг є забезпечення доступності всіх навчальних ресурсів. Освітні програмні системи розробляються таким чином, щоб бути зручними для всіх учнів, незалежно від їхнього місцезнаходження. Це дозволяє зробити онлайн навчання доступним навіть для тих студентів, які перебувають в ізольованих або віддалених регіонах, надаючи їм рівний доступ до навчальних матеріалів та інструментів [1].

Завдяки використанню таких технологій створюються можливості для впровадження нових методів навчання, включаючи гібридні курси та персоналізовані програми для учнів. Це сприяє підвищенню мотивації до навчання, дозволяючи кожному учню обирати найзручніші для нього способи здобуття знань. Крім того, в освітніх програмних рішеннях інтегруються новітні технології, які дозволяють застосовувати креативні стратегії навчання та інтерактивні методи [1].

Програмне забезпечення розробляється з урахуванням потреб конкретної освітньої установи, що дозволяє налаштувати систему відповідно до індивідуальних вимог. Такий підхід також забезпечує можливість масштабування системи у міру зростання організації. У результаті навчальний процес стає більш ефективним, а комунікація між учнями, викладачами та батьками значно полегшується завдяки використанню додаткових програмних інструментів [1].

Крім того, освітні програмні рішення підтримують розвиток самостійного навчання. Студенти можуть працювати в своєму темпі,

налаштовувати графік занять та швидко отримувати необхідну інформацію. Це допомагає учням більш ефективно засвоювати матеріал та покращувати свої навички управління часом [1], [2].

Не менш важливим є технічне обслуговування та постійна підтримка програмного забезпечення. Розробники забезпечують регулярні оновлення та виправлення помилок, що дозволяє зберігати стабільність і актуальність системи, а також підвищувати її продуктивність. Це важливо для забезпечення безперервної роботи освітнього процесу та ефективного використання всіх функціональних можливостей програмних рішень [1], [2].

Розвиток програмного забезпечення для освітніх потреб має величезний потенціал і продовжує розвиватися з новими можливостями для учнів і студентів. Технології, що інтегруються в освітні системи, відкривають шлях до інноваційного підходу до навчання, значно змінюючи традиційні процеси [1], [2].

Однією з найважливіших змін є впровадження штучного інтелекту та машинного навчання в освітні програмні платформи. Ці технології не тільки дозволяють створювати персоналізовані підходи до навчання, а й забезпечують аналітичні інструменти для прогнозування успіхів студентів. Системи, що використовують ці технології, можуть адаптуватися до потреб кожного учня, що підвищує ефективність навчання та дозволяє досягати кращих результатів [1], [2].

Інтеграція доповненої та віртуальної реальності також змінює спосіб навчання. Такі інструменти, як віртуальні лабораторії, 3D-симуляції та екскурсії, дозволяють учням занурюватися в навчальний процес, створюючи враження реального досвіду. Це значно підвищує інтерес до матеріалу та мотивацію студентів [1], [2].

Використання технології блокчейн для зберігання навчальних даних, таких як сертифікати та оцінки, робить систему більш прозорою та безпечною.

Така технологія дозволяє уникнути фальсифікацій та забезпечує швидку перевірку документів, що полегшує процес верифікації кваліфікацій [1], [2].

Мобільні платформи для навчання також набирають популярності, надаючи студентам можливість навчатися в будь-який час і з будь-якого місця. Це створює умови для гнучкого та зручного навчання, яке відповідає індивідуальним потребам кожного користувача [1], [2].

Що стосується основних функцій освітніх додатків, важливою є наявність відео уроків, що дозволяє студентам вивчати матеріал у зручний для себе час. Оцінювання в реальному часі дає змогу відстежувати прогрес і вчасно коригувати навчальний процес. Додаткові ресурси, такі як статті та посібники, збагачують досвід навчання, забезпечуючи доступ до додаткової інформації. Спілкування з викладачем у режимі реального часу також є важливим для збереження підтримки та мотивації учнів [1], [2].

Функція відстеження прогресу допомагає користувачам бачити свої досягнення, що стимулює подальше навчання та зростання. Організація матеріалу в зрозумілі модулі допомагає зосередитися на одному етапі, покращуючи управління навчальним процесом і надаючи студентам відчуття контролю над своїм навчанням [1], [2].

Вартість створення комплексної освітньої програмної системи залежить від кількох важливих факторів, які визначають кінцеву суму витрат. Це можуть бути такі аспекти, як рівень складності розробки, обрані функції, технологічні рішення, розмір команди розробників та багато інших умов. Для різних типів програмних рішень витрати можуть значно варіюватися [1], [2].

Програмне забезпечення для освітніх цілей може бути класифіковано за рівнем складності. Наприклад, для створення простих додатків, які реалізують базові функції, як-то тести чи картки, витрати будуть меншими. Складніші системи, що включають мультимедійні елементи, базову аналітику або інтерактивний контент, потребують більших інвестицій. У випадку розробки високотехнологічних платформ, які включають інноваційні функції, як

доповнена реальність або інтеграція з штучним інтелектом, ціна може значно зрости [1], [2].

Функціональні можливості програмного забезпечення також мають великий вплив на його загальну вартість. Наприклад, інвестиції в дизайн інтерфейсу користувача можуть суттєво покращити взаємодію з продуктом, але вимагають додаткових витрат на розробку. Якщо система включає налаштований навчальний контент, інтерактивні модулі або мультимедійні елементи, це також може збільшити витрати на створення. Крім того, додавання адаптивного навчання за допомогою технологій штучного інтелекту чи машинного навчання ще більше підвищує вартість розробки. Для створення інструментів для співпраці, таких як чат, форуми або віртуальні класи, також необхідні додаткові ресурси [1], [2].

Технології, які використовуються для розробки програмного забезпечення, можуть впливати на його вартість. Вибір конкретних технологічних стеків, таких як бази даних, фреймворки чи мови програмування, а також інтеграція з іншими зовнішніми сервісами, такими як системи для управління навчанням чи хмарні платформи, може додатково збільшити загальні витрати на проект [1], [2].

Вартість розробки програмного забезпечення для освітніх цілей та обліку викладацької роботи визначається цілим рядом чинників, що залежать від різних аспектів проекту. Одним із важливих моментів є місце розташування розробників. Компанії, що займаються розробкою такого програмного забезпечення, можуть перебувати в різних частинах світу, і вартість їхніх послуг значно відрізняється в залежності від регіону. Наприклад, на розробників з Північної Америки та Західної Європи зазвичай витрачається більше, ніж на фахівців з країн Східної Європи або Азії. Різниця в оплаті праці також впливає на загальний бюджет проекту [1], [2].

З іншого боку, вибір методології розробки також має важливе значення. Гнучкі підходи дозволяють здійснювати проект поетапно, з постійними

коригуваннями на основі зворотного зв'язку. Це може потребувати великих початкових витрат, але така методика зазвичай виявляється ефективнішою в довгостроковій перспективі. Методи з фіксованими етапами, наприклад водоспадна модель, можуть бути менш дорогими, але вони обмежують можливості для адаптації протягом процесу [1], [2].

Додатково, вартість залежить від складності програмного забезпечення. Якщо програма включає лише базові функції, наприклад, тести або навчальні модулі, витрати будуть значно меншими. Однак, якщо в проект включено складні функціональні можливості, такі як інтерактивні елементи, потокове відео або персоналізація навчального процесу на основі штучного інтелекту, це суттєво підвищує вартість розробки. Такі складні елементи вимагають більше часу для реалізації, а також більш висококваліфікованих фахівців для тестування та коригування програмного продукту [1], [2].

Що стосується дизайну та взаємодії з користувачем, то ці аспекти теж не можна ігнорувати. Гарний інтерфейс забезпечує не лише привабливість, але й зручність користування програмним забезпеченням. Чітке розташування елементів, простота навігації та інтуїтивно зрозумілий дизайн значно покращують досвід користувачів. Це, безумовно, вимагає додаткових витрат на розробку, проте інвестиції у високу якість інтерфейсу окупаються завдяки підвищеному задоволенню користувачів і збереженню їхньої зацікавленості у використанні програми [1], [2].

Крім того, важливими факторами є термін розробки та технічне обслуговування. Наймання досвідчених фахівців може вимагати більших витрат, але це гарантує високу якість роботи, що зменшує ризики помилок у майбутньому. Підтримка та оновлення програмного забезпечення після запуску також є важливими аспектами, які необхідно враховувати при плануванні бюджету проекту [1], [2].

Вибір платформи для розробки програмного забезпечення має суттєвий вплив на витрати, час і ресурси, необхідні для створення та запуску продукту.

Вибір між Android та iOS визначатиме специфіку роботи програми на різних типах пристроїв. Програма для Android може бути адаптована під широкий спектр пристроїв, від дешевих моделей до моделей дуже дорогих, що збільшує обсяг роботи, необхідний для забезпечення бездоганної роботи на всіх пристроях. iOS, в свою чергу, працює лише на продуктах Apple, що забезпечує менше варіантів для тестування і полегшує процес розробки. Проте програми для iOS можуть бути дорожчими через більшу купівельну спроможність користувачів [1], [2].

У разі потреби у розробці програмного забезпечення для обох платформ, процес зазвичай потребує більше ресурсів, оскільки для кожної платформи потрібен окремий код [1], [2].

Часові рамки також мають безпосередній вплив на загальні витрати. Зміна термінів чи прискорення процесу за рахунок додаткових годин роботи чи більшої кількості розробників підвищує вартість проекту. Тому важливо правильно розрахувати час для розробки, щоб забезпечити високий рівень якості без додаткових витрат [1], [2].

Після запуску програмне забезпечення для підтримки освітньої діяльності потребує постійного технічного обслуговування та оновлень, що може включати виправлення помилок, додавання нових функцій, оновлення для підтримки нових пристроїв та операційних систем, а також покращення захисту даних. Регулярне оновлення і обслуговування є важливими для підтримки належного функціонування програмного забезпечення і уникнення негативних відгуків користувачів, що можуть виникнути через збої або відсутність актуальності продукту [2], [3].

Розробка програмного забезпечення для освітньої діяльності часто потребує інтеграції з іншими інструментами для забезпечення зручності та ефективності. Підключення до таких платформ, як Microsoft Teams, Zoom або Google Classroom, значно розширює можливості програми. Це дозволяє користувачам зручно взаємодіяти між собою, ділитися матеріалами,

відстежувати прогрес і навіть реєструватися для участі в онлайн заняттях. Однак, на додаток до значної користі, інтеграція вимагає додаткових витрат часу та ресурсів розробників, які повинні створити відповідний код для підключення до кожної з платформ. Це завдання потребує уважної перевірки, щоб забезпечити належну функціональність без збоїв. Крім того, деякі платформи можуть вимагати окремих ліцензій чи додаткових витрат, що також варто враховувати в процесі планування [3], [4].

Безпека даних є ще одним критично важливим аспектом при створенні освітнього програмного забезпечення. Оскільки програму використовуватимуть учні та викладачі, в ній зберігаються чутливі дані, такі як оцінки, домашні завдання, особисті контактні відомості. Захист цієї інформації вимагає інтеграції складних систем шифрування, ідентифікації користувачів та контролю доступу. Для забезпечення безпеки також необхідно регулярно оновлювати програму, щоб відповідати вимогам законодавства та технічним стандартам. Це збільшує вартість розробки такого програмного забезпечення та потребує додаткових ресурсів, проте є важливим для довгострокового функціонування додатку [3], [4].

Розробка програмного забезпечення для освітніх цілей повинна базуватися на розумінні потреб користувачів. Першим кроком є детальне вивчення вимог як учнів, так і викладачів. Це дозволяє створити програму, яка відповідає їхнім очікуванням і вирішує конкретні проблеми. Крім того, необхідно розробити чітко структурований навчальний контент, що відповідатиме освітнім стандартам і цілям. Вибір правильних інструментів для розробки також має важливе значення для забезпечення масштабованості, безпеки та сумісності з різними пристроями. Інтерфейс програми повинен бути інтуїтивно зрозумілим і легким у використанні, щоб забезпечити комфортне навчання без зайвих труднощів. Успіх також залежить від регулярних відгуків та можливості тестування програмного забезпечення на

ранніх етапах, що дозволить виявити недоліки та покращити функціональність продукту [3], [4].

Різноманітні освітні програми створюються з метою підтримки процесу навчання та управління діяльністю викладачів. Різні типи такого програмного забезпечення розробляються відповідно до конкретних завдань та потреб. Важливими є платформи для електронного навчання, які дають можливість проходити численні онлайн курси, що охоплюють безліч дисциплін. Такі програми дозволяють користувачам самостійно освоювати нові теми, а також проводити тести та виконувати завдання для більш глибокого засвоєння матеріалу [3], [4].

Інтерактивні електронні книги додають додаткову цінність до традиційного процесу читання. Вони містять мультимедійні елементи, такі як анімації або відео, що дозволяють зробити навчання більш динамічним і інтуїтивно зрозумілим. Користувачі мають можливість взаємодіяти з матеріалом, що полегшує процес засвоєння знань [3], [4].

Програми для індивідуального навчання, на кшталт репетиторів, забезпечують можливість отримати додаткові знання або уточнити проблеми в різних галузях, що особливо корисно для студентів, які потребують індивідуальної допомоги або хочуть навчатись у власному темпі [3], [4].

Для деяких освітніх потреб розробляються спеціалізовані програми, що надають можливості для удосконалення в конкретних галузях, таких як математика чи мовні навички. Ці програми допомагають вирішити конкретні завдання, покращуючи знання користувачів у певній області [3], [4].

Рішення щодо того, чи здійснювати розробку програмного забезпечення для освіти власними силами чи скористатися сторонніми ресурсами, залежить від кількох факторів, включаючи бюджет, терміни та складність проекту. Власна розробка дає більше контролю та гнучкості, дозволяючи створити продукт, який відповідає специфічним вимогам. Водночас використання сторонніх ресурсів може бути більш економічним, надаючи доступ до

спеціалізованих знань і забезпечуючи швидший процес розробки. Залежно від потреб проекту, може бути доцільним комбінувати обидва варіанти. Гібридний підхід дозволяє скористатися сильними сторонами як внутрішньої команди, так і зовнішніх постачальників послуг, що допомагає досягти бажаного результату з максимальною ефективністю [3], [4].

Розробка програмного забезпечення для освітніх потреб має значний потенціал для вдосконалення як навчального процесу, так і управління викладацькою діяльністю. Інвестування в створення таких систем може значно підвищити якість освітніх послуг та забезпечити більш ефективну організацію навчання [3], [4].

Сучасне програмне забезпечення для освітніх установ і навчальних закладів відіграє важливу роль у процесах навчання та оцінювання результатів, допомагаючи викладачам ефективно управляти своїми обов'язками та обліковими операціями. Використання таких інструментів дозволяє значно спростити роботу з документами, зменшуючи залежність від паперових носіїв і забезпечуючи високий рівень автоматизації. Впровадження програмних рішень для оцінювання і підрахунку результатів допомагає школам та університетам скоротити час, необхідний для обробки тестових робіт та складання звітів. Подібні програми дозволяють не тільки полегшити процес проведення іспитів, але й надають викладачам зручні інструменти для ведення обліку студентів протягом всього навчального року [4], [5].

Завдяки доступності різноманітних безкоштовних рішень, таких як просте програмне забезпечення для оцінювання, ці інструменти є економічно вигідними для освітніх установ різних рівнів. Це дозволяє освітянам зосередитися на навчальному процесі та вдосконаленні програм, не витрачаючи значних коштів на дорогі рішення. Програмне забезпечення для оцінювання забезпечує точність, оперативність і зручність у роботі з великими обсягами інформації [4], [5].

У свою чергу, програми для створення друкованих матеріалів, такі як програмне забезпечення для настільних публікацій, відкривають широкі можливості для викладачів і студентів. За допомогою таких інструментів можна легко створювати різноманітні навчальні матеріали та брошури, що є важливою складовою частиною освітнього процесу. Такі програми дозволяють створювати наочні, візуально привабливі матеріали, які значно підвищують ефективність подачі навчальної інформації [4], [5].

Популяризація онлайн ресурсів також змінила підхід до довідкового програмного забезпечення, яке було замінено на сучасні вебплатформи, такі як енциклопедії та бази даних. Інтернет став основним джерелом знань, що надає миттєвий доступ до великої кількості матеріалів, необхідних для навчання та досліджень. Тим не менш, в певних сферах, таких як медицина та право, довідкові програми все ще знаходять своє місце, забезпечуючи фахівців зручними глосаріями та іншими ресурсами [4], [5].

Програмне забезпечення для тренувальних тестів і вправ є незамінним для підготовки студентів до іспитів і сертифікаційних процедур. Це дає змогу створювати спеціалізовані тести, які допомагають у закріпленні знань і навичок, перевіряючи прогрес студентів перед основними випробуваннями. Технології для дистанційного навчання, зокрема інструменти для створення навчальних посібників, стали важливими компонентами в умовах пандемії, забезпечивши безперервність навчального процесу у віртуальному середовищі. Вчителі мають змогу надавати студентам потрібні матеріали та організовувати навчальні заняття, використовуючи доступні онлайн рішення для максимальної гнучкості та ефективності навчання [4], [5].

Технології, які забезпечують симуляцію реального світу, відкривають нові горизонти для навчання. Завдяки віртуальній та доповненій реальності студенти мають змогу практично освоювати різні навички в режимі реального часу. Використання таких технологій дозволяє створити навчальний досвід, наближений до реальних умов, що значно підвищує ефективність навчання.

Наприклад, вивчення процесу польоту чи механізмів роботи техніки стає можливим через спеціалізоване програмне забезпечення, що робить знання більш доступними та зрозумілими для учнів [4], [5].

Окреме програмне забезпечення розробляється для дітей з особливими потребами, яке забезпечує доступ до навчального матеріалу за допомогою адаптаційних технологій. Використання таких систем разом з допоміжними засобами, такими як синтезатори мовлення, робить навчання доступним і ефективним для всіх учнів, незалежно від їхніх потреб. Ці інструменти дозволяють створити інклюзивне середовище, де кожен учень має рівні можливості для навчання [4], [5].

Математичне програмне забезпечення є потужним інструментом для розвитку аналітичних та критичних здібностей учнів. Завдяки таким програмам студенти можуть покращити свої навички розв'язування проблем, отримуючи не лише теоретичні знання, а й практичний досвід вирішення складних задач. Ці програми пропонують різноманітні методи вирішення завдань, що допомагає учням краще розуміти матеріал і знаходити оптимальні способи досягнення результатів [4], [5].

Електронні курси та навчальні програми створюють нові можливості для здобуття знань, надаючи доступ до різноманітних матеріалів і завдань. Така система дозволяє не лише пройти курс, але й поглибити розуміння теми через інтерактивні елементи, тести та завдання, що супроводжують навчальний процес. Завдяки доступу до навчальних матеріалів у цифровому форматі, студенти можуть вчитися в зручний час, що сприяє ефективнішому освоєнню матеріалу [4], [5].

Отже, розвиток технологій у галузі освіти дозволяє створювати різноманітні програмні інструменти для підтримки вчителів та студентів у процесі навчання. Використання таких програмних засобів забезпечує зручний доступ до навчальних матеріалів і полегшує організацію освітнього процесу,

що, в свою чергу, сприяє покращенню якості освіти та забезпечує інтерактивний підхід до навчання [4], [5].

Визначено, що розробка програмного забезпечення для освітньої діяльності та обліку викладацької роботи активно змінює традиційні підходи до навчального процесу, забезпечуючи більш індивідуалізоване, інтерактивне та динамічне навчання. Впровадження новітніх технологій сприяє створенню ефективних інструментів для онлайн освіти, а також оптимізації адміністративних функцій. Сучасні програмні засоби дозволяють удосконалювати методи навчання і значно полегшують облік викладацької роботи, завдяки чому вчителі можуть більше зосереджуватись на навчальному процесі, а адміністрація – на керуванні даними.

1.2 Аналіз програмного забезпечення для обліку викладацької роботи

Проаналізуємо програмне забезпечення для обліку викладацької роботи [6]-[14].

Програмне забезпечення Schoology (рис. 1.1) виступає як багатофункціональна система підтримки викладацької діяльності, орієнтована на повноцінний облік і організацію всіх етапів освітнього процесу. Завдяки широким можливостям цього цифрового середовища забезпечується створення, зберігання й упорядкування даних, пов'язаних із проведенням занять, взаємодією з учнями, плануванням навчальних активностей і оцінюванням. Програмне забезпечення дає змогу викладачам вести облік своєї роботи в межах кожного курсу, фіксувати результати діяльності здобувачів освіти, слідкувати за виконанням завдань, організовувати навчальні події та вчасно оновлювати навчальний контент [6], [7], [12].

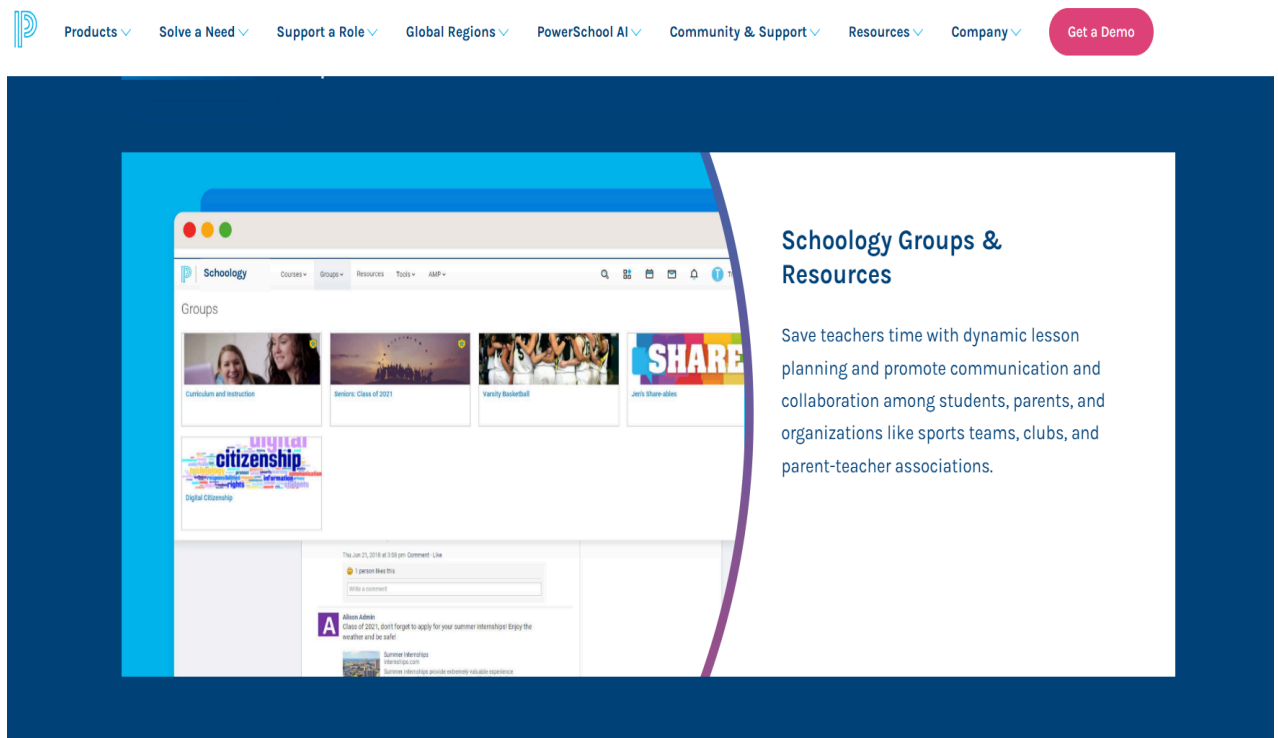


Рисунок 1.1 – Програмне забезпечення Schoology [12]

Інтеграція програмного забезпечення Schoology із системами адміністрування навчальних закладів дозволяє не лише зберігати інформацію, а й виконувати аналітику викладацької активності, що значно розширює функції контролю за педагогічним навантаженням. У середовищі платформи передбачено зручні інструменти для відображення розкладу, фіксації відвідуваності, надання зворотного зв'язку та спілкування з учнями, батьками і колегами, що сприяє чіткому документуванню всієї роботи викладача.

Підтримка обліку викладацької роботи в програмному забезпеченні Schoology реалізується через можливість створення структурованих освітніх курсів із прив'язкою до термінів, завдань і навчальних груп. Здійснюється постійне оновлення класних списків, облік оцінок і участі студентів у навчальному процесі. Усі ці дії фіксуються в системі та дозволяють викладачу повертатися до історії навчання, аналізувати результативність та планувати подальші освітні заходи на основі зібраних даних. Schoology не лише

автоматизує, а й упорядковує повсякденну викладацьку діяльність, забезпечуючи її прозорість і зручність контролю [6], [7], [12].

Основними характеристиками програмного забезпечення Schoology є такі [6], [7], [12]:

- система для обліку викладацької роботи: програмне забезпечення Schoology надає зручні інструменти для фіксації, організації та аналізу діяльності викладачів, що сприяє ефективному плануванню навчального процесу [6], [7], [12];

- засіб управління навчанням: програмне забезпечення Schoology підтримує створення курсів, завдань, опитувань, тестів, що дозволяє викладачам повноцінно керувати навчальними активностями [6], [7], [12];

- інтеграція з іншими платформами: програмне забезпечення Schoology може взаємодіяти з Google Workspace, Microsoft Office, а також понад двома сотнями інших інструментів для освіти [6], [7], [12];

- підтримка комунікації: програмне забезпечення Schoology надає можливість надсилати повідомлення, створювати обговорення та взаємодіяти з учнями та колегами в межах одного середовища [6], [7], [12];

- інструменти для звітності: програмне забезпечення Schoology дозволяє отримувати аналітичні звіти про успішність здобувачів освіти, участь у заняттях, виконання завдань, що сприяє кращому обліку викладацької активності [6], [7], [12];

- можливість взаємодії з батьками: програмне забезпечення Schoology дає змогу залучати батьків до освітнього процесу через окремі профілі з доступом до оцінок, розкладу та комунікації [6], [7], [12];

- централізоване управління навчальними матеріалами: програмне забезпечення Schoology дозволяє зберігати, оновлювати та поширювати навчальний контент у межах одного середовища [6], [7], [12];

- адаптивність до навчального процесу: програмне забезпечення Schoology підтримує різні формати занять, очні, дистанційні та змішані, що робить платформу універсальною для викладацької діяльності [6], [7], [12];
- інструменти для планування: програмне забезпечення Schoology дає змогу створювати календарі, розклади, графіки уроків і контролювати навчальний ритм відповідно до потреб курсу [6], [7], [12];
- підтримка професійного розвитку викладачів: програмне забезпечення Schoology дозволяє організовувати тренінги, курси та обмін методичними матеріалами між освітянами [6], [7], [12].

Серед недоліків програмного забезпечення Schoology можна відзначити недостатню інтуїтивну навігацію, що ускладнює швидкий доступ до окремих функцій, особливо для нових користувачів. Також при використанні програмного забезпечення Schoology іноді виникають труднощі зі стабільністю роботи системи, що може впливати на своєчасне внесення або збереження даних, пов'язаних з викладацькою діяльністю. Крім того, умови ліцензування і вартість програмного продукту часто залишаються недостатньо прозорими, що може створювати бар'єри для впровадження платформи в окремих навчальних закладах [6], [7], [12].

Програмне забезпечення Learnidy (рис. 1.2) розроблено для підтримки викладацької діяльності та забезпечення педагогів усім необхідним для ефективної організації навчального процесу. Його функціональність дозволяє з будь-якого місця готувати матеріали до занять, створювати структуровані плани уроків та аналізувати хід викладання. Завдяки наявності вбудованих шаблонів уроків, якими можна користуватися як індивідуально, так і ділитися з колегами, здійснюється не лише планування навчального процесу, а й контроль за реалізацією викладацьких завдань. У програмі реалізовано підтримку обліку діяльності викладача, що дозволяє зберігати історію викладання, узгоджувати дії в межах колективу, а також відстежувати ефективність застосованих методик [7], [13].

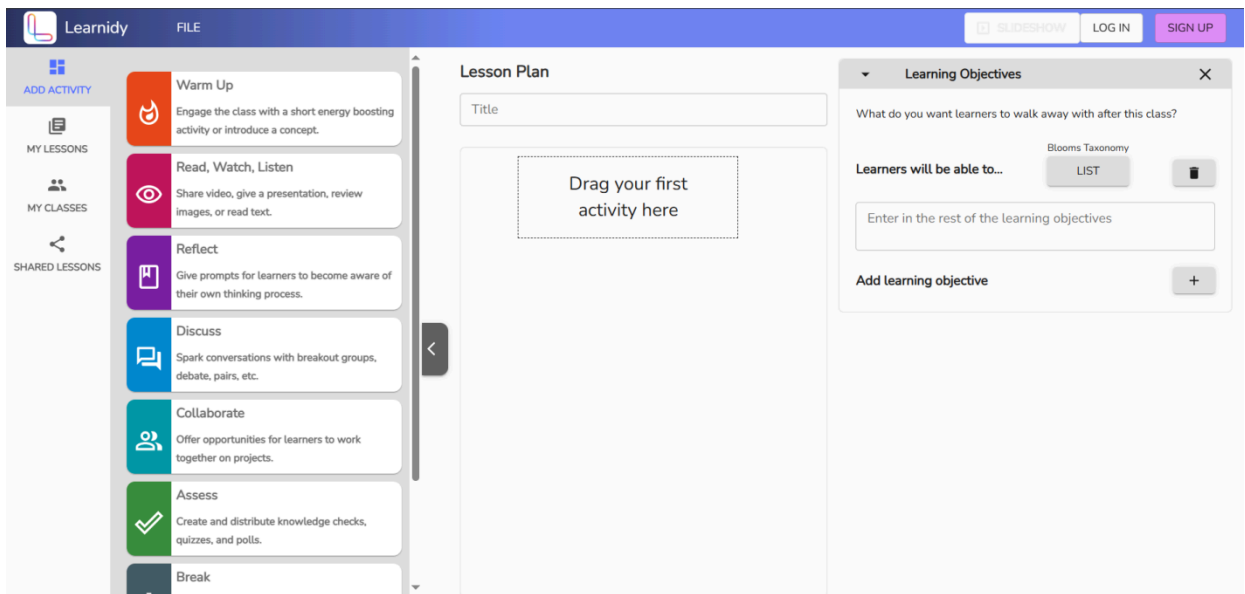


Рисунок 1.2 – Програмне забезпечення Learnidy [13]

Програмне забезпечення Learnidy є зручною платформою, яка не лише автоматизує щоденну рутину викладача, а й дає змогу зосередитися на змісті навчання. Програмне забезпечення дозволяє вести повноцінний облік роботи викладача — від створення навчального матеріалу до його практичної реалізації в класі. Воно спрямоване на те, щоб зробити викладання більш змістовним і прозорим, а сам процес навчання — контрольованим, організованим і підзвітним. Розроблене безпосередньо вчителями, воно враховує реальні потреби користувачів, що дозволяє говорити про Learnidy як про ефективний інструмент для фіксації, організації й удосконалення викладацької діяльності [7], [13].

Програмне забезпечення Learnidy має такі особливості [7], [13]:

- розробка з урахуванням потреб викладачів: програмне забезпечення Learnidy створене викладачами, що забезпечує практичну орієнтацію на реальні завдання викладацької діяльності [7], [13];

- підтримка планування уроків: програмне забезпечення Learnidy надає широкий вибір шаблонів для побудови структурованих і послідовних занять, що полегшує облік навчального контенту [7], [13];

- можливість спільного використання матеріалів: програмне забезпечення Learnidy надає зручне середовище для обміну планами між педагогами в інтегрованій освітній мережі [7], [13];

- інструменти для контролю викладацької діяльності: програмне забезпечення Learnidy надає вбудовані засоби дозволяють фіксувати, переглядати та оновлювати інформацію про проведені заняття та педагогічну активність [7], [13];

- функції аналітики: програмне забезпечення Learnidy надає систему аналізу даних, яка дозволяє вивчати ефективність уроків і краще координувати роботу вчителя з освітніми цілями [7], [13];

- орієнтація на активне навчання: програмне забезпечення Learnidy надає шаблони, які спрямовані на створення залученого освітнього середовища, що підтримує системний облік активної взаємодії викладача з учнями [7], [13].

Серед недоліків програмного забезпечення Learnidy можна відзначити обмежену функціональність через те, що платформа перебуває на стадії розробки, що може призводити до нестабільної роботи або браку деяких важливих можливостей для повноцінного обліку викладацької діяльності. Також інтерфейс ще не до кінця оптимізовано, що іноді ускладнює швидкий доступ до потрібних інструментів або функцій. Крім того, відсутність повної інтеграції з іншими популярними освітніми системами поки що обмежує можливості синхронізації даних і ускладнює комплексне ведення обліку навчальної активності в рамках ширших освітніх платформ [7], [13].

Програмне забезпечення NetSupport School (рис. 1.3) надає повноцінну підтримку викладацької діяльності шляхом реалізації численних інструментів, що дозволяють вести облік і контроль навчального процесу в цифровому середовищі. Його функціональність спрямована на постійний моніторинг роботи учнів, фіксацію взаємодії та ефективне управління навчальними ресурсами. Завдяки можливості доступу до пристроїв студентів,

забезпечується безперервне спостереження за їхньою активністю, що сприяє точному обліку участі у навчальному процесі [11], [14].

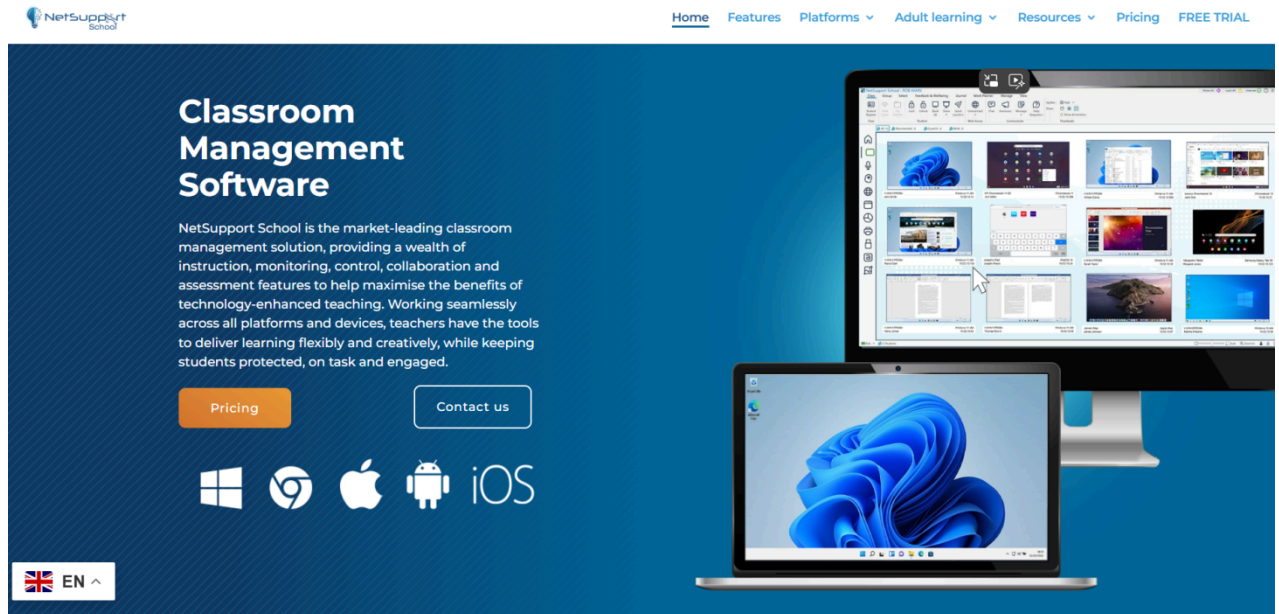


Рисунок 1.3 – Програмне забезпечення NetSupport School [14]

Застосування аудіо та відео засобів дозволяє створювати інтерактивне навчальне середовище з фіксацією педагогічної активності. За допомогою віртуальної дошки програмного забезпечення NetSupport School, яка оснащена інструментами для створення анотацій і пояснень, викладач може зручно демонструвати матеріал і при цьому реєструвати свої дії як частину навчального плану. У програмі NetSupport School реалізовано функції опитування та миттєвого зворотного зв'язку, що дозволяє відслідковувати реакцію учнів та враховувати її при подальшому плануванні занять [11], [14].

Завдяки кросплатформенності забезпечується гнучкий облік викладацької роботи незалежно від типу пристроїв, які використовуються в класі. Весь функціонал програмного забезпечення NetSupport School орієнтований на те, щоб дати змогу вчителю не тільки організовувати навчання, а й ретельно фіксувати хід його реалізації [11], [14].

Програмне забезпечення NetSupport School має такі характеристики та особливості [11], [14]:

- підтримка різних платформ: програмне забезпечення NetSupport School підтримує сумісність із Windows, macOS, Android, iOS та Chromebook, що дозволяє використовувати програмне забезпечення у змішаному середовищі навчання [11], [14];

- функції моніторингу студентів: програмне забезпечення NetSupport School надає викладачам можливість переглядати активність учнів у реальному часі, включаючи перегляд екранів, використання клавіатури та миші, а також доступ до мікрофонів [11], [14];

- інтерактивні інструменти викладання: у програмному забезпеченні NetSupport School реалізовано віртуальну дошку, опитування, відео- та аудіозв'язок, що сприяє ефективній взаємодії викладача з учнями під час занять [11], [14];

- засоби оцінювання знань: програмне забезпечення NetSupport School надає функції миттєвого зворотного зв'язку, тести, вікторини й опитування, що дозволяє контролювати успішність учнів та фіксувати результати викладацької діяльності [11], [14];

- централізоване керування класом: викладач може блокувати екрани, запускати або зупиняти програми та контролювати активність у браузері, що спрощує організацію навчального процесу [11], [14].

Серед недоліків програмного забезпечення NetSupport School можна відзначити певну складність у налаштуванні на початкових етапах, що може ускладнювати його впровадження в освітній процес без відповідної технічної підтримки. Крім того, інтерфейс не завжди є інтуїтивно зрозумілим для нових користувачів, що може впливати на швидкість адаптації викладачів. У деяких випадках спостерігаються проблеми з продуктивністю на слабших пристроях, що обмежує можливість повноцінного використання всіх функцій програмного забезпечення [11], [14].

Порівняльну характеристику програмного забезпечення для обліку викладацької роботи наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння програмного забезпечення для обліку викладацької роботи

Критерій порівняння	Schoology [12]	Learnidy [13]	NetSupport School [14]
Зручність інтерфейсу	+—	+	+—
Підтримка онлайн комунікації	+	+—	+
Наявність шаблонів для занять	+—	+	—
Функції моніторингу студентів	+—	—	+
Інтеграція з іншими сервісами	+	+—	+—
Прозорість ціноутворення	—	+—	+—

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато програмних засобів для обліку викладацької роботи. Проте деякі програмні засоби для обліку викладацької роботи можуть бути досить складними для налаштування та адаптації системи до специфіки конкретного навчального закладу. Кожен заклад має власні вимоги до обліку видів діяльності та звітності, і далеко не кожне програмне забезпечення здатне гнучко налаштуватися під ці особливості без залучення фахівців або значних витрат часу. Ще одним викликом є потреба в навчанні персоналу, крім того, інтерфейс деяких програм для обліку викладацької роботи може бути незручним або перевантаженим, що ускладнює роботу користувачів і знижує загальну ефективність використання. З технічного боку, програмне забезпечення для обліку викладацької роботи може мати проблеми з надійністю або сумісністю. У випадку локального встановлення ПЗ для обліку викладацької роботи додатковою складністю є потреба в обслуговуванні та

резервному копіюванні, що вимагає технічного персоналу або залучення сторонніх спеціалістів. Тому актуальною є розробка програмного забезпечення для обліку викладацької роботи.

При розробці програмного забезпечення для обліку викладацької роботи необхідно забезпечити такі функціональні вимоги:

- підтримка можливості редагування інформації про викладача;
- підтримка можливості керування інформацією про викладацьку діяльність, зокрема, про дисципліни, які викладає викладач, а також навчальні групи та аудиторії, де проводяться заняття;
- можливість редагування розкладу викладача;
- можливість формування таблиці занять викладача згідно розкладу;
- підтримка можливості створення журналу обліку викладацької роботи;
- підтримка можливості відображення занять викладача згідно обраних за допомогою календаря дат.

1.3 Висновки за розділом 1

Визначено, що розробка програмного забезпечення для освітньої діяльності та обліку викладацької роботи активно змінює традиційні підходи до навчального процесу, забезпечуючи більш індивідуалізоване, інтерактивне та динамічне навчання. Впровадження новітніх технологій сприяє створенню ефективних інструментів для онлайн освіти, а також оптимізації адміністративних функцій. Сучасні програмні засоби дозволяють удосконалювати методи навчання і значно полегшують облік викладацької роботи, завдяки чому вчителі можуть більше зосереджуватись на навчальному процесі, а адміністрація – на керуванні даними.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для обліку викладацької роботи. Проте

деякі програмні засоби для обліку викладацької роботи можуть бути досить складними для налаштування та адаптації системи до специфіки конкретного навчального закладу. Кожен заклад має власні вимоги до обліку видів діяльності та звітності, і далеко не кожне програмне забезпечення здатне гнучко налаштуватися під ці особливості без залучення фахівців або значних витрат часу. Ще одним викликом є потреба в навчанні персоналу, крім того, інтерфейс деяких програм для обліку викладацької роботи може бути незручним або перевантаженим, що ускладнює роботу користувачів і знижує загальну ефективність використання. З технічного боку, програмне забезпечення для обліку викладацької роботи може мати проблеми з надійністю або сумісністю. У випадку локального встановлення ПЗ для обліку викладацької роботи додатковою складністю є потреба в обслуговуванні та резервному копіюванні, що вимагає технічного персоналу або залучення сторонніх спеціалістів. Тому актуальною є розробка програмного забезпечення для обліку викладацької роботи.

Сформульовано функціональні вимоги до програмного забезпечення для обліку викладацької роботи.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці програмного забезпечення для обліку викладацької роботи було використано мову програмування C# [15], [16].

Мова програмування C#, розроблена компанією Microsoft, від початку свого існування демонструє динамічний розвиток і високу ефективність у вирішенні широкого спектру завдань. Її створення було зумовлене потребою розробників у безпечному, об'єктно-орієнтованому підході до програмування, який би дозволяв гнучко адаптуватися до вимог сучасного програмного середовища [15], [16].

Особливістю C# є її тісна інтеграція з платформою .NET, що відкриває доступ до багатого набору інструментів, бібліотек і компонентів, які значно спрощують розробку додатків різної складності. Мова дозволяє легко реалізовувати функціональність, пов'язану з обробкою даних, побудовою вебсервісів, асинхронними операціями та забезпеченням безпеки. Її синтаксис вирізняється інтуїтивною зрозумілістю та чистотою, що позитивно впливає на зменшення кількості помилок у процесі написання коду [15], [16].

Протягом років у C# впроваджувалися численні сучасні можливості, від шаблонів проектування до покращеної роботи з посилованими типами й мовними конструкціями, що дозволяють досягти високого рівня читабельності та підтримання програмного коду. Це забезпечує її актуальність і конкурентоспроможність серед інших мов [15], [16].

Сфера застосування C# охоплює настільні застосунки для Windows, мобільні рішення на базі платформ Xamarin або MAUI, ігрові продукти за допомогою Unity, хмарні сервіси у межах інфраструктури Azure, а також вебзастосунки на базі ASP.NET Core. Такий функціональний діапазон робить мову універсальним інструментом для створення як складних корпоративних систем, так і невеликих проєктів [15], [16].

Значну роль у поширенні мови відіграє її підтримка з боку професійних інструментів розробки, передусім середовища Visual Studio, що пропонує широкі можливості для написання, тестування й налагодження програмного коду. Наявність великої кількості освітніх матеріалів, документації, бібліотек та активної спільноти розробників сприяє швидкому навчанню і ефективній роботі з мовою [15], [16].

Ураховуючи вбудовану систему керованого виконання, яка знижує ймовірність типових помилок, пов'язаних із керуванням пам'яттю, мова програмування C# забезпечує надійність і стабільність розроблюваного програмного продукту. Саме завдяки сукупності цих якостей мова користується стійкою популярністю в академічному середовищі та в індустрії розробки програмного забезпечення [15], [16].

Доцільність застосування C# для створення програмного забезпечення для обліку в навчальних закладах зумовлюється як технічною досконалістю самої мови, так і широкими можливостями, що надаються інструментальним середовищем та супутніми технологіями [15], [16].

У цій мові реалізовано принципи об'єктного програмування, що дозволяє формувати програмні рішення, орієнтовані на модульну побудову, повторне використання логіки та ефективну структурування коду. Платформа .NET, у рамках якої функціонує C#, забезпечує зручний доступ до розгалуженої інфраструктури готових бібліотек та фреймворків, завдяки яким можлива розробка систем різного рівня складності [15], [16].

Широке впровадження цієї мови спостерігається і в освітньому секторі, де розробка застосунків здійснюється з використанням таких технологій, як Windows Forms чи WPF. Для створення вебрішень застосовується середовище ASP.NET, яке дозволяє запускати програми через локальні мережі або через Інтернет, забезпечуючи зручний доступ з різних пристроїв без залежності від конкретного браузера [15], [16].

Ще однією значущою перевагою є ефективна взаємодія з реляційними базами даних, зокрема з Microsoft SQL Server, а також з іншими СУБД з відкритим вихідним кодом. Це забезпечує можливість повноцінного зберігання та обробки великих обсягів інформації, що стосуються навчального процесу, при цьому особлива увага приділяється безпеці даних і захисту доступу до них [15], [16].

Розробка проєктів на C# здійснюється в потужному інтегрованому середовищі Visual Studio, яке об'єднує в собі засоби для написання, налагодження, тестування та збирання програмних продуктів. Це значно прискорює цикл створення програм і полегшує роботу над спільними проєктами [15], [16].

Таким чином, C# виступає як технологія, що поєднує надійність, високу продуктивність, гнучкість реалізації програмних рішень і широку підтримку, завдяки чому її вибір для автоматизації викладацького обліку в закладах освіти є цілком обґрунтованим [15], [16].

Обґрунтування вибору мови програмування для розробки програмного забезпечення для обліку викладацької роботи наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки програмного забезпечення для обліку викладацької роботи

Критерій порівняння мов програмування	Мова програмування		
	C#	Python	Java
Підтримка графічного інтерфейсу	+	+—	+—
Продуктивність виконання	+	—	+
Документація та підтримка	+	+—	+
Зручність для створення ПЗ для обліку викладацької роботи	+	—	—

Наявність готових бібліотек та фреймворків	+	+-	+
--	---	----	---

Отже, для реалізації програмного забезпечення для обліку викладацької роботи обрано мову програмування C#, яка підтримує об'єктно-орієнтоване програмування, модульність, повторне використання коду, інкапсуляцію та інші принципи, що значно спрощують проєктування складних програмних систем. Завдяки платформі .NET розробник на мові C# отримує доступ до великої кількості бібліотек, інструментів і фреймворків, які дозволяють реалізовувати функціональність для роботи з базами даних, створення графічного або вебінтерфейсу, а також реалізацію логіки бізнес-процесів.

2.2 Вибір середовища розробки для створення програмного забезпечення для обліку викладацької роботи

В якості середовища розробки для створення програмного забезпечення для обліку викладацької роботи було обрано середовище Visual Studio [17], [18].

Середовище розробки Visual Studio забезпечує повний цикл програмної розробки з підтримкою широкого спектру мов програмування, зокрема мови C#. Завдяки своїй багатофункціональності платформа дозволяє реалізовувати проєкти, призначені для найрізноманітніших систем — від настільних застосунків до вебрішень і кросплатформних мобільних застосунків, охоплюючи Windows, Linux, Android, iOS та хмарні сервіси [17], [18].

Інструментарій Visual Studio забезпечує можливість адаптації під конкретні потреби проєкту шляхом глибокої інтеграції сторонніх рішень, таких як розширення для аналізу коду, поліпшення навігації або взаємодії з хмарною інфраструктурою. Використання додаткових модулів дозволяє значно

розширити функціонал без необхідності змінювати основне середовище [17], [18].

Платформа Visual Studio підтримує повноцінне впровадження процесів тестування, що включає автоматизовані, модульні, функціональні та навантажувальні тести. Завдяки інтеграції з такими фреймворками, як MSTest, NUnit та xUnit, створюється можливість оперативного виявлення помилок і підвищення стабільності продукту [17], [18].

Організація колективної розробки реалізується завдяки підтримці систем контролю версій і сервісів спільної роботи над кодом. Інтеграція з Git, GitHub, Azure DevOps сприяє безперервній інтеграції й доставці, а також дозволяє налагодити ефективну взаємодію між учасниками команди на різних етапах життєвого циклу продукту [17], [18].

Крім локальних інструментів, Visual Studio забезпечує доступ до хмарних сервісів, що особливо актуально при розгортанні інфраструктур у середовищі Azure. Така інтеграція полегшує розробку, розгортання та підтримку сучасних рішень з елементами аналітики, захистом даних та автентифікацією користувачів [17], [18].

Середовище Visual Studio пропонує декілька редакцій, кожна з яких орієнтована на свою аудиторію. Версія Community дозволяє безкоштовно реалізовувати як навчальні, так і комерційні проєкти малого масштабу, тоді як Professional і Enterprise адресовані більш вимогливим професійним середовищам. Завдяки такому підходу забезпечується доступність необхідного інструментарію для проєктів різного рівня складності [17], [18].

Під час створення програм для освітніх цілей доцільно застосовувати середовище розробки Visual Studio як універсальне середовище з розширеними можливостями та повною підтримкою технологій C# і .NET. У цьому середовищі реалізується повний цикл програмної розробки — від початкового проєктування до фінального впровадження, з урахуванням потреб сучасних користувачів [17], [18].

Інструментарій Visual Studio дозволяє реалізовувати як веборієнтовані, так і настільні рішення. При розробці інтерфейсів активно використовується як Windows Forms, так і WPF, а для вебзастосунків — шаблони ASP.NET, що входять до базової конфігурації середовища. Завдяки прямій інтеграції з системами управління базами даних на зразок Microsoft SQL Server стає можливим здійснення всіх необхідних дій із даними — від формування таблиць до обробки запитів і налагодження [17], [18].

Функціонал середовища включає інтелектуальне авто заповнення коду, вбудовану систему підказок і зручні інструменти для пошуку та усунення помилок, що суттєво підвищує якість розроблюваного програмного забезпечення. Інтеграція із системами керування версіями, зокрема Git, сприяє ефективному управлінню змінами та командній роботі [17], [18].

Однією з головних переваг середовища розробки Visual Studio є потужна підтримка створення як настільних, так і вебзастосунків. Для розробки локальних систем обліку можна використовувати Windows Forms або WPF. Інтеграція з Microsoft SQL Server дозволяє легко підключати базу даних, створювати запити, переглядати структуру таблиць і проводити відлагодження безпосередньо зі середовища розробки. Середовище розробки Visual Studio має вбудовану систему підказок, авто доповнення коду, а також потужні засоби для візуального проєктування інтерфейсу користувача. Це значно пришвидшує розробку та зменшує кількість помилок. Середовище також дозволяє ефективно працювати з системами контролю версій, що важливо у командній розробці [17], [18].

Конфігурація середовища може бути адаптована до конкретних вимог завдяки можливості додавання розширень. Плагіни дозволяють автоматизувати тестування, формування звітів і оцінювання, що відкриває додаткові перспективи у створенні інформаційних систем для обліку в освітніх установах [17], [18].

У результаті використання Visual Studio забезпечує централізований підхід до реалізації всіх завдань, що виникають під час розробки. Це спрощує процес, роблячи його більш логічним, послідовним і зрозумілим як для окремих спеціалістів, так і для цілих команд [17], [18].

Обґрунтування вибору середовища розробки для створення програмного забезпечення для обліку викладацької роботи наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для створення програмного забезпечення для обліку викладацької роботи

Критерій порівняння середовищ розробки	Середовища розробки		
	Visual Studio	JetBrains Rider	MonoDevelop
Повна підтримка C# та .NET	+	+	+—
Вбудований дизайнер інтерфейсу	+	+—	+—
Зручність інструментів налагодження	+	+	+—
Підтримка розширень та плагінів	+	+—	+—
Засоби автоматичного тестування	+	+	+—

Таким чином, для створення програмного забезпечення для обліку викладацької роботи обрано середовище розробки Visual Studio, що надає зручні інструменти для повного циклу розробки – від проєктування до тестування, налагодження та розгортання програмного продукту. Visual Studio має вбудовану систему підказок, авто доповнення коду, інтегрований налагоджувач коду, а також потужні засоби для візуального проєктування

інтерфейсу користувача, що значно пришвидшує розробку та зменшує кількість помилок.

2.3 Висновки за розділом 2

Для реалізації програмного забезпечення для обліку викладацької роботи обрано мову програмування C#, яка підтримує об'єктно-орієнтоване програмування, модульність, повторне використання коду, інкапсуляцію та інші принципи, що значно спрощують проєктування складних програмних систем. Завдяки платформі .NET розробник на мові C# отримує доступ до великої кількості бібліотек, інструментів і фреймворків, які дозволяють реалізовувати функціональність для роботи з базами даних, створення графічного або вебінтерфейсу, а також реалізацію логіки бізнес-процесів.

Для створення програмного забезпечення для обліку викладацької роботи обрано середовище розробки Visual Studio, що надає зручні інструменти для повного циклу розробки – від проєктування до тестування, налагодження та розгортання програмного продукту. Visual Studio має вбудовану систему підказок, авто доповнення коду, інтегрований налагоджувач коду, а також потужні засоби для візуального проєктування інтерфейсу користувача, що значно пришвидшує розробку та зменшує кількість помилок.

3 ОПИС ПРОГРАМИ

3.1 Структура програмного забезпечення для обліку викладацької роботи

Структуру програмного забезпечення для обліку викладацької роботи наведено на рис. 3.1.

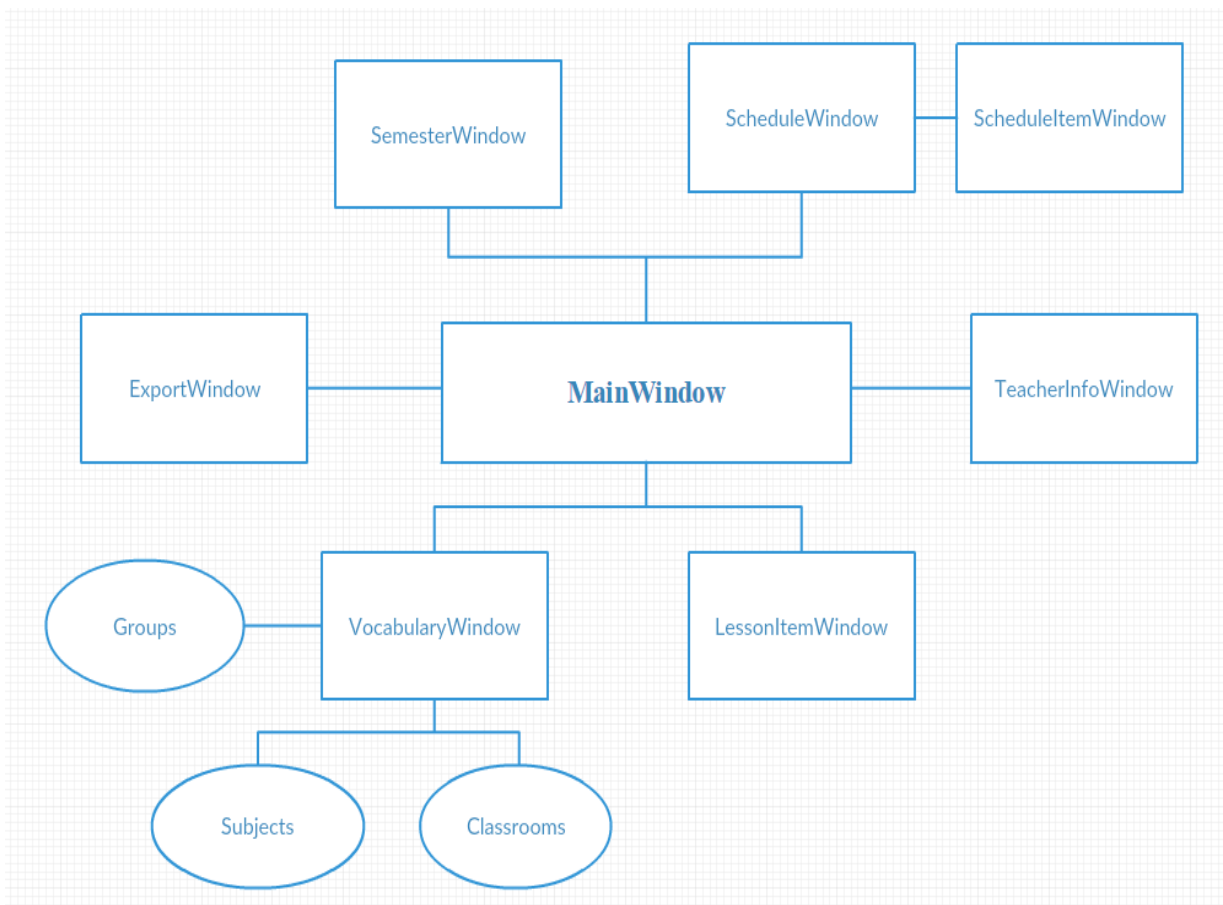


Рисунок 3.1 – Структура програмного забезпечення для обліку викладацької роботи

У програмі для обліку викладацької роботи реалізовано кілька взаємопов'язаних функціональних елементів, кожен з яких виконує окрему роль у загальній логіці роботи програмного забезпечення. Центральним елементом програмного забезпечення для обліку викладацької роботи є головне вікно, де відбувається ініціалізація ключових компонентів,

завантажуються вихідні дані й організовується доступ до всіх основних функцій — від перегляду занять до переходу в інші частини системи. Саме з цього інтерфейсу користувач розпочинає роботу.

Організація навчальних періодів здійснюється через спеціальний модуль, призначений для створення або редагування семестрів. Побудова розкладу представлена окремим інтерфейсом, де розміщено таблицю із запланованими одиницями навчального процесу. Для внесення змін або додавання нових записів передбачено контекстне вікно програмного забезпечення для обліку викладацької роботи, яке дає змогу точно налаштувати параметри кожного елементу розкладу.

Формування конкретного заняття виконується в іншому функціональному модулі, який викликається безпосередньо з головного вікна програмного забезпечення для обліку викладацької роботи. У структурі програми також передбачено обробку довідкової інформації, що охоплює предмети, аудиторії, навчальні групи. Ці дані керуються через окремий інтерфейс, який дозволяє редагувати словники програми.

Окремий структурний елемент програмного забезпечення для обліку викладацької роботи відповідає за підготовку звітів на основі даних розкладу. Його завданням є узгодження параметрів експорту та фільтрація даних за вибраними семестрами. Ще одна частина системи відповідає за зберігання та редагування інформації про викладачів, включаючи прізвище, ім'я та по батькові, навантаження та інші персональні характеристики. Процес формування звітів викладача організовано через проміжне вікно, яке виступає посередником між основним інтерфейсом і модулем експорту. Перед початком формування підсумкових даних користувачу пропонується вибрати відповідні навчальні періоди. Ще один важливий елемент програми — вікно, призначене для перегляду і коригування інформації про викладачів, що дозволяє зберігати актуальні персональні відомості.

Кожен елемент програмного забезпечення для обліку викладацької роботи побудовано таким чином, щоб забезпечити чітку функціональну ізоляцію і водночас безперервну інтеграцію з іншими компонентами, що дозволяє реалізувати злагоджену роботу всієї системи.

3.2 Функціонування програмного забезпечення для обліку викладацької роботи

Функціонування програмного забезпечення для обліку викладацької роботи подамо за допомогою діаграми прецедентів, зображеної на рис. 3.2.

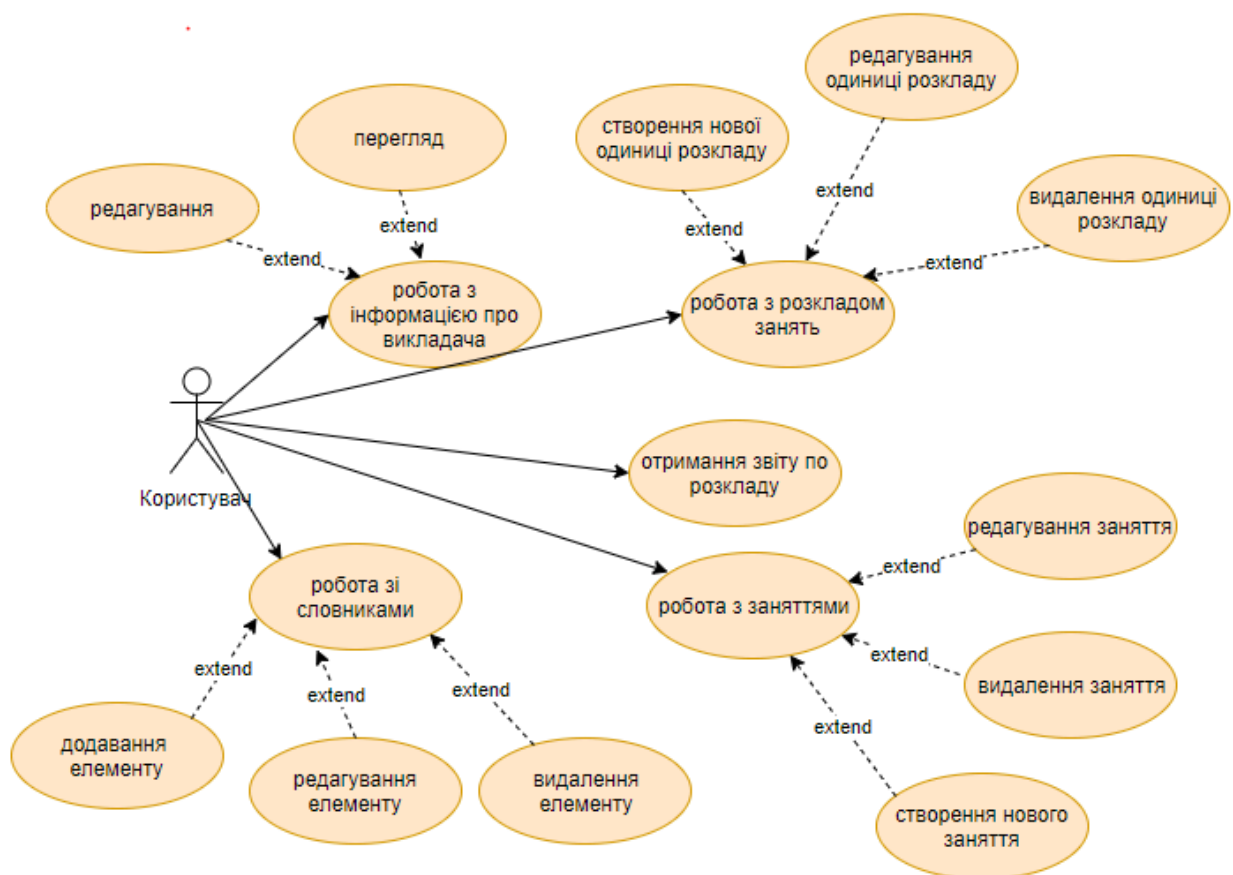


Рисунок 3.2 – Діаграма прецедентів програмного забезпечення для обліку викладацької роботи

У межах діаграми прецедентів програмного забезпечення, призначеного для обліку викладацької діяльності, визначається основна роль користувача, яка охоплює взаємодію з ключовими функціональними можливостями системи. Центральним суб'єктом виступає користувач, що ініціює всі основні дії в межах графічного інтерфейсу.

Функціональність передбачає можливість керування інформацією про викладача. У цьому випадку здійснюється доступ до персональних даних, із подальшим їх переглядом та редагуванням. Інша суттєва складова — управління розкладом. Прецедент охоплює формування, зміну та перегляд структури розкладу занять для певного семестру чи періоду.

Окремо реалізовано прецедент роботи з окремими навчальними одиницями. Тут користувач має змогу створювати нові заняття, редагувати вже наявні або вилучати непотрібні. Ці дії пов'язані з конкретними параметрами: датами, часом, аудиторіями, групами, викладачами тощо.

Ще одним напрямом діяльності в межах діаграми є формування звітів. Цей процес вимагає вибору відповідних параметрів, зокрема семестру або розкладу певного викладача, після чого система автоматично створює звітний документ.

Окремий набір функцій відповідає за роботу зі словниками. Це включає можливість наповнення, редагування та використання таких структур, як перелік дисциплін, навчальних груп і аудиторій, що є необхідним для точного формування розкладу.

Таким чином, діаграма прецедентів демонструє цілісну картину взаємодії користувача з усіма компонентами системи, відображаючи логіку та сценарії використання програмного забезпечення в процесі обліку викладацького навантаження.

Під час роботи програмного забезпечення для обліку викладацької діяльності весь функціонал спрямовується на забезпечення зручного створення, редагування та обліку даних про навчальний процес. Усе

починається з головного вікна, де після завантаження системи користувач отримує доступ до основних модулів. Після вибору необхідного семестру відкривається можливість формувати розклад, змінювати або доповнювати інформацію про заняття та керувати відповідними словниками — такими як дисципліни, аудиторії, навчальні групи.

У розділі створення розкладу здійснюється перегляд і редагування структури занять, збереження нових елементів та їх візуалізація у зручному вигляді. Для внесення точних параметрів кожного заняття передбачено окремий інтерфейс, що дозволяє обрати дату, час, вид заняття, а також прив'язати його до викладача й навчальної групи.

Після заповнення даних користувач може звернутися до модуля експорту, в якому формується звіт за обраними параметрами. Цей етап передбачає відбір релевантних семестрів і автоматизовану генерацію документа, що відображає викладацьке навантаження. Окрім цього, передбачено керування персональними даними викладачів, які можна переглядати й редагувати в спеціальному вікні — це охоплює ідентифікаційну інформацію, посадове навантаження та інші характеристики.

Взаємодія між модулями реалізується на основі чітко визначених сценаріїв переходів, що забезпечує безперервність процесу й інтуїтивне керування. Завдяки інтеграції з базою даних здійснюється постійне збереження змін, перевірка введених значень та доступ до історії змін. Таким чином, функціонування програмного комплексу зводиться до ефективної координації всіх дій, пов'язаних із плануванням, обліком і контролем викладацької роботи.

3.3 Розробка бази даних програмного забезпечення для обліку викладацької роботи

База даних програмного забезпечення для обліку викладацької роботи призначена для зберігання, впорядкування та зв'язування інформації, що

стосується семестрів, навчальних занять, розкладу, предметів, аудиторій, груп та інших супровідних об'єктів. Вона забезпечує централізоване управління даними, зменшує ризики помилок під час обробки інформації та дає змогу ефективно формувати звіти про навантаження викладачів. Архітектура бази даних побудована за принципами реляційної моделі, де інформація структурована у вигляді взаємопов'язаних таблиць.

Схему бази даних програмного забезпечення для обліку викладацької роботи наведено на рис. 3.3.

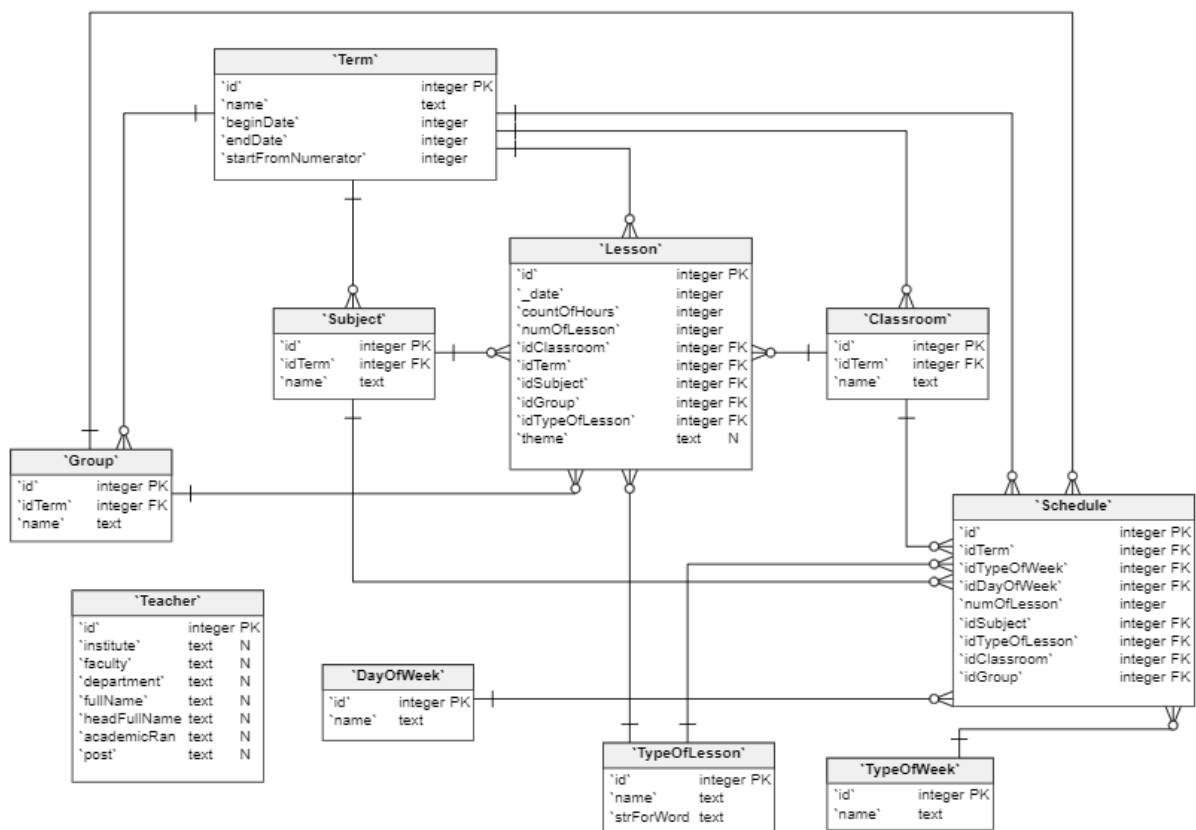


Рисунок 3.3 – Схема бази даних програмного забезпечення для обліку викладацької роботи

У процесі збереження та обробки даних у системі для обліку викладацької роботи застосовується реляційна база даних, основу моделі даних становлять таблиці, що пов'язані між собою через зовнішні ключі, що дозволяє підтримувати узгодженість і забезпечує логічну цілісність інформації.

Кожен семестр представлено як окрема одиниця в таблиці, яка містить не лише текстову назву й часові межі, а й інформацію про те, з якого типу тижня починається облік. Формування навчального розкладу реалізовано через окрему таблицю, де кожен запис відповідає одному елементу тижневого плану занять і включає детальну прив'язку до типу тижня, дня, порядкового номера пари, дисципліни, групи, аудиторії та типу заняття.

Ще одна важлива частина структури — таблиця, що акумулює інформацію про вже проведені заняття. У ній фіксуються дата, тривалість, тема, а також зв'язок із семестром, навчальним предметом і відповідною групою. Це дозволяє відслідковувати фактичне викладацьке навантаження.

Словникові таблиці, серед яких — предмети, навчальні групи та аудиторії, дають змогу формувати і використовувати стандартні елементи розкладу. Дані цієї категорії зберігаються в універсальному форматі з ідентифікатором, назвою та вказівкою на відповідний семестр, до якого вони належать.

Інші таблиці, що містять перелік днів тижня, типів тижнів та типів занять, формуються автоматично при створенні бази даних і не підлягають редагуванню з боку користувача. Вони використовуються системою як довідники для подальшого наповнення основних таблиць. Типи занять, зокрема, містять не лише текстову назву, а й спеціальне текстове представлення для експорту у форматі документів.

Уся структура бази даних програмного забезпечення для обліку викладацької роботи проєктувалася з урахуванням зручності розширення, збереження внутрішньої логіки та забезпечення ефективного виконання запитів до даних під час основних сценаріїв використання системи.

3.4 Проєктування інтерфейсу програмного забезпечення для обліку викладацької роботи

Проектування інтерфейсу взаємодії користувача з програмним забезпеченням для обліку викладацької роботи виконано з урахуванням вимог технічного завдання. При розробленні програмного забезпечення для обліку викладацької роботи враховані функціональні вимоги до програми:

- підтримка можливості редагування інформації про викладача;
- підтримка можливості керування інформацією про викладацьку діяльність, зокрема, про дисципліни, які викладає викладач, а також навчальні групи та аудиторії, де проводяться заняття;
- можливість редагування розкладу викладача;
- можливість формування таблиці занять викладача згідно розкладу;
- підтримка можливості створення журналу обліку викладацької роботи;
- підтримка можливості відображення занять викладача згідно обраних за допомогою календаря дат.

Інтерфейс головної форми програмного забезпечення для обліку викладацької роботи наведено на рис. 3.4.

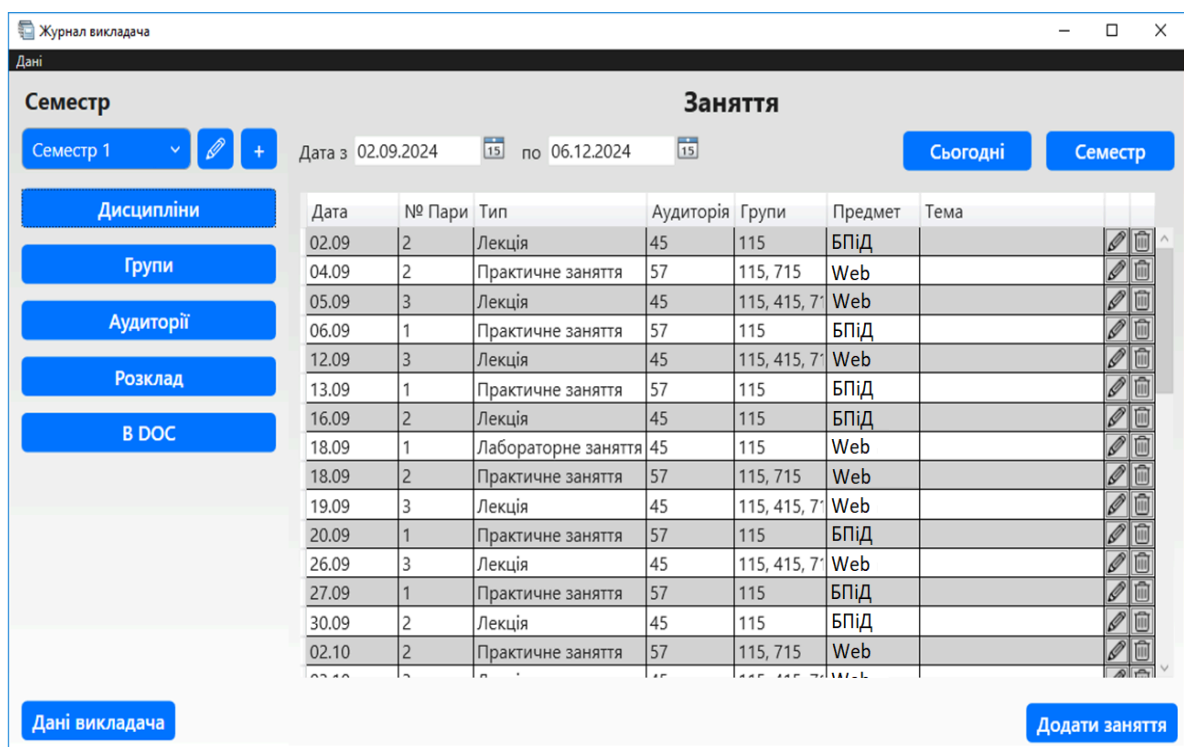
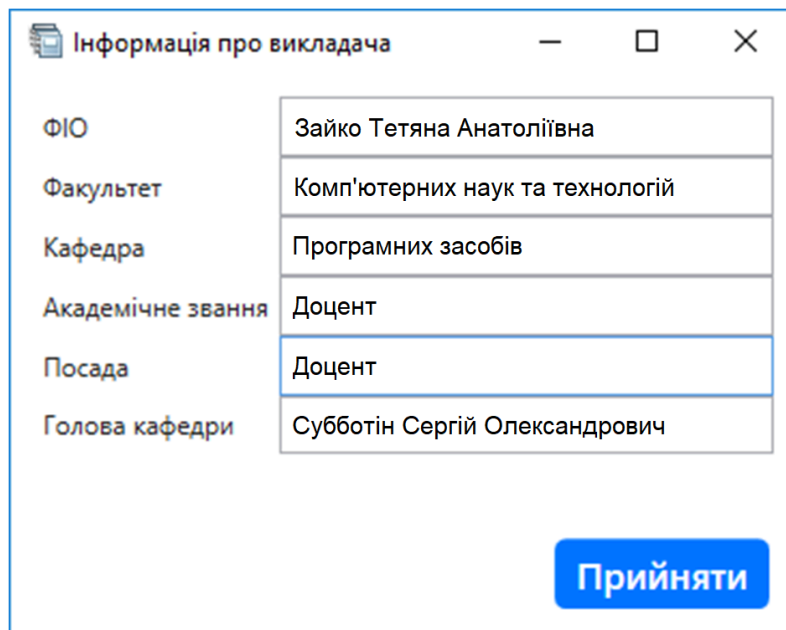


Рисунок 3.4 – Інтерфейс головної форми програмного забезпечення для обліку викладацької роботи

Форму з інформацією про викладача наведено на рис. 3.5.

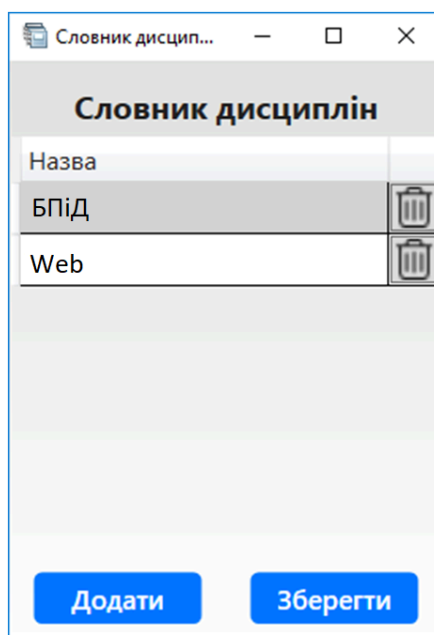


ФІО	Зайко Тетяна Анатоліївна
Факультет	Комп'ютерних наук та технологій
Кафедра	Програмних засобів
Академічне звання	Доцент
Посада	Доцент
Голова кафедри	Субботін Сергій Олександрович

Прийняти

Рисунок 3.5 – Форма з інформацією про викладача

Форму з інформацією про дисципліни наведено на рис. 3.6.

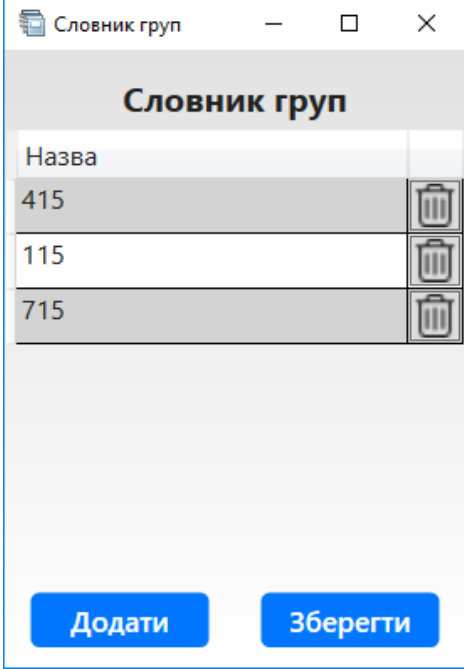


Словник дисциплін		
Назва		
БПід		
Web		

Додати Зберегти

Рисунок 3.6 – Форма з інформацією про дисципліни

Форму з інформацією про академічні групи наведено на рис. 3.7.



Назва	
415	
115	
715	

Додати Зберегти

Рисунок 3.7 – Форма з інформацією про групи

Форму з інформацією про аудиторії наведено на рис. 3.8.

Словник аудиторій

Назва	
45	
57	

Додати **Зберегти**

Рисунок 3.8 – Форма з інформацією про аудиторії

Форму формування розкладу викладача наведено на рис. 3.9.

Семестр

Розклад **Додати**

Тип неділі	День неділі	Номер заняття	Предмет	Тип заняття	Аудиторія	Групи	
Чисельник	Понеділок	2	БПід	Лекція	45	115	
Чисельник	Середа	2	Web	Практичне заняття	45	115, 715	

Заповнити

Рисунок 3.9 – Форма формування розкладу викладача

3.5 Висновки за розділом 3

Розроблено програмне забезпечення для обліку викладацької роботи.

Запропоновано структуру програмного забезпечення для обліку викладацької роботи. У програмі для обліку викладацької роботи реалізовано кілька взаємопов'язаних функціональних елементів, кожен з яких виконує окрему роль у загальній логіці роботи програмного забезпечення. Кожен елемент програмного забезпечення для обліку викладацької роботи побудовано таким чином, щоб забезпечити чітку функціональну ізоляцію і водночас безперервну інтеграцію з іншими компонентами, що дозволяє реалізувати злагоджену роботу всієї системи.

За допомогою діаграми прецедентів описано функціонування програмного забезпечення для обліку викладацької роботи. У межах діаграми прецедентів програмного забезпечення, призначеного для обліку викладацької діяльності, визначається основна роль користувача, яка охоплює взаємодію з ключовими функціональними можливостями системи. Діаграма прецедентів подає цілісну картину взаємодії користувача з усіма компонентами програмної системи, відображаючи логіку та сценарії використання програмного забезпечення в процесі обліку викладацького навантаження.

Розроблено базу даних програмного забезпечення для обліку викладацької роботи, яка призначена для зберігання, впорядкування та зв'язування інформації, що стосується семестрів, навчальних занять, розкладу, предметів, аудиторій, груп та інших супровідних об'єктів. Вона забезпечує централізоване управління даними, зменшує ризики помилок під час обробки інформації та дає змогу ефективно формувати звіти про навантаження викладачів. Архітектура бази даних побудована за принципами реляційної моделі, де інформація структурована у вигляді взаємопов'язаних таблиць.

Описано особливості реалізації програмного забезпечення для обліку викладацької роботи. Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для обліку викладацької роботи.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для обліку викладацької роботи. Програма написана на мові C#, яка підтримує об'єктно-орієнтоване програмування, модульність, повторне використання коду, інкапсуляцію та інші принципи, що значно спрощують проєктування складних програмних систем. В якості середовища розробки для створення програмного забезпечення для обліку викладацької роботи обрано Visual Studio, що надає зручні інструменти для повного циклу розробки – від проєктування до тестування, налагодження та розгортання програмного продукту.

Програмне забезпечення для обліку викладацької роботи забезпечує виконання таких функцій:

- підтримка можливості редагування інформації про викладача;
- підтримка можливості керування інформацією про викладацьку діяльність, зокрема, про дисципліни, які викладає викладач, а також навчальні групи та аудиторії, де проводяться заняття;
- можливість редагування розкладу викладача;
- можливість формування таблиці занять викладача згідно розкладу;
- підтримка можливості створення журналу обліку викладацької роботи;
- підтримка можливості відображення занять викладача згідно обраних за допомогою календаря дат.

Для роботи програмного забезпечення для обліку викладацької роботи потрібні такі технічні та програмні ресурси, які забезпечують стабільну і безперебійну обробку інформації у процесі взаємодії з базою даних і графічним інтерфейсом користувача. Мінімально допустимий рівень обчислювальної потужності передбачає наявність центрального процесора з тактовою частотою не менше 2.0 ГГц, що дає змогу оперативно виконувати

запити та обробляти події в реальному часі. Обсяг оперативної пам'яті повинен бути не менше 4 ГБ для забезпечення комфортної роботи з великими обсягами інформації та паралельного виконання кількох модулів. Обсяг вільного місця на жорсткому диску має забезпечувати збереження бази даних і допоміжних файлів, тому рекомендоване значення — від 500 МБ. Графічний адаптер повинен забезпечувати роботу з інтерфейсами, підтримка роздільної здатності екрана від 1024×768 також є бажаною для коректного відображення вікон і таблиць. Необхідно забезпечити стабільне мережеве підключення.

Функціональні характеристики розробленого програмного забезпечення для обліку викладацької роботи наведено у технічному завданні (додаток А). Фрагмент тексту програмного забезпечення для обліку викладацької роботи наведено у додатку Б.

4.2 Характеристики програми для обліку викладацької роботи

Програмне забезпечення для обліку викладацької роботи дозволяє виконувати ведення даних про семестри, розклад занять, конкретні навчальні події та інформацію про викладачів. Передбачено інтегровану підтримку словників, що дозволяє формувати списки навчальних дисциплін, груп, аудиторій та інших довідкових елементів, необхідних для автоматизованого створення розкладу.

Користувачеві надається інтуїтивно зрозумілий інтерфейс із можливістю створення, редагування та перегляду записів у структурованому вигляді. Реалізовано модуль формування звітності, який дає змогу отримувати документально оформлену інформацію про проведені заняття згідно з заданими параметрами. У межах архітектури застосовано локальне зберігання даних, що дозволяє працювати автономно без підключення до зовнішніх серверів. Програмна реалізація забезпечує збереження цілісності даних,

стабільну обробку користувацьких запитів та підтримує ключові операції для щоденного ведення обліку педагогічного навантаження.

База даних програмного забезпечення для обліку викладацької роботи дозволяє виконувати зберігання, впорядкування та зв'язування інформації, що стосується семестрів, навчальних занять, розкладу, предметів, аудиторій, груп та інших супровідних об'єктів. Вона забезпечує централізоване управління даними, зменшує ризики помилок під час обробки інформації та дає змогу ефективно формувати звіти про навантаження викладачів. Архітектура бази даних побудована за принципами реляційної моделі, де інформація структурована у вигляді взаємопов'язаних таблиць.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для запуску програмного забезпечення для обліку викладацької роботи необхідно запустити основний файл.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмного забезпечення для обліку викладацької роботи є інформація про викладачів, здобувачів, дисципліни, аудиторії та ін.

Вихідними даними програмного забезпечення для обліку викладацької роботи є створені документи, зокрема, розклад викладача та журнал його роботи.

4.3.3 Повідомлення

Користувач програмного забезпечення для обліку викладацької роботи може отримати такі повідомлення: повідомлення про некоректно введені дані, повідомлення про формування журналу роботи викладача, повідомлення про те, що введено не всі дані.

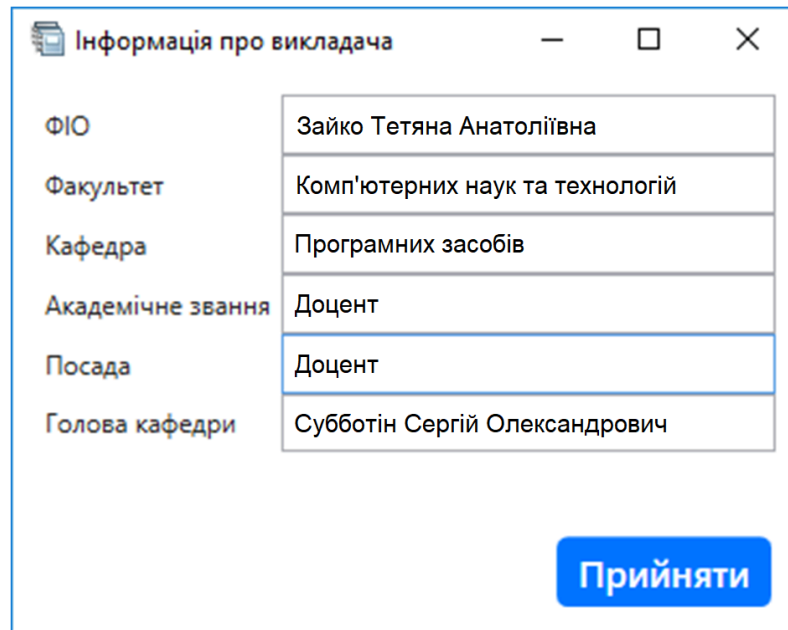
4.4 Виконання програмного забезпечення для обліку викладацької роботи

Після запуску програмного забезпечення для обліку викладацької роботи користувачу відображається головна форма (рис. 4.1), де надається можливість переходу до редагування інформації стосовно навчальних дисципліни, груп, аудиторій, а також формування розкладу роботи викладача.

Рисунок 4.1 – Інтерфейс головної форми програмного забезпечення для обліку викладацької роботи

При натисканні на кнопку Дані викладача відбувається перехід до відповідної форми (рис. 4.2), де можна ввести та відредагувати інформацію про викладача.

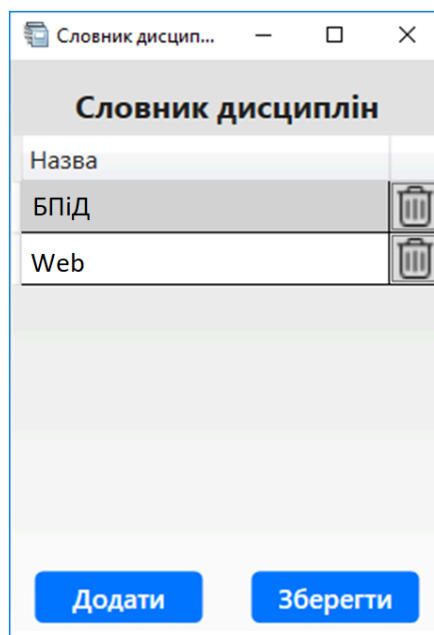
При натисканні на кнопку Предмети відбувається перехід до форми зі словником дисциплін (рис. 4.3), де подається перелік дисциплін викладача. Для додавання нової дисципліни до переліку необхідно натиснути кнопку Додати. Видалення дисципліни з переліку відбувається шляхом натискання кнопки кошику праворуч від дисципліни.



ФІО	Зайко Тетяна Анатоліївна
Факультет	Комп'ютерних наук та технологій
Кафедра	Програмних засобів
Академічне звання	Доцент
Посада	Доцент
Голова кафедри	Субботін Сергій Олександрович

Прийняти

Рисунок 4.2 – Форма з інформацією про викладача

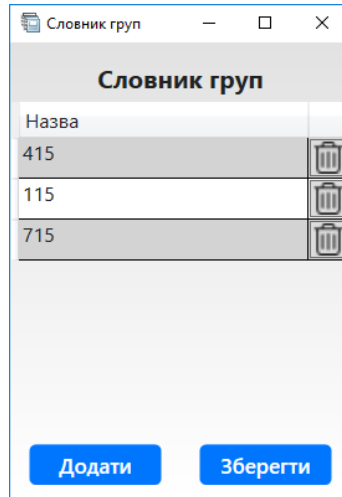


Словник дисциплін		
Назва		
БПід		
Web		

Додати Зберегти

Рисунок 4.3 – Форма з інформацією про дисципліни

При натисканні на кнопку Групи відбувається перехід до форми зі словником навчальних груп (рис. 4.4), де подається перелік груп, з якими працює викладач. Для додавання нової групи до переліку необхідно натиснути кнопку Додати. Видалення навчальної групи з переліку відбувається шляхом натискання кнопки кошику праворуч від групи.

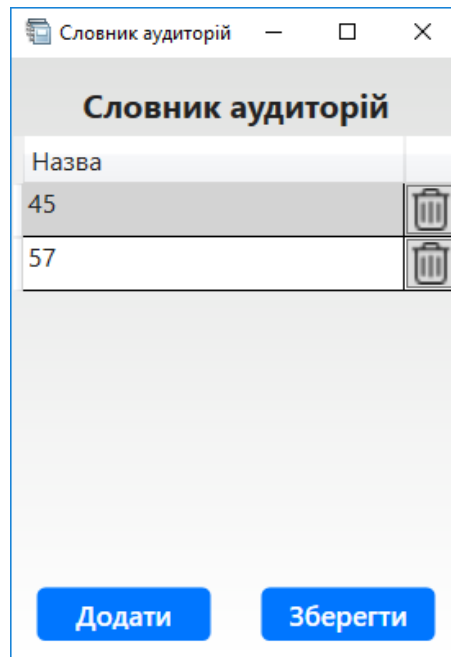


Назва	
415	
115	
715	

Додати Зберегти

Рисунок 4.4 – Форма з інформацією про групи

При натисканні на кнопку Аудиторії відбувається перехід до форми зі словником аудиторій (рис. 4.5), де подається перелік аудиторій, в яких проводить заняття викладач. Для додавання нової аудиторії до переліку необхідно натиснути кнопку Додати. Видалення аудиторії з переліку відбувається шляхом натискання кнопки кошику праворуч від аудиторії.

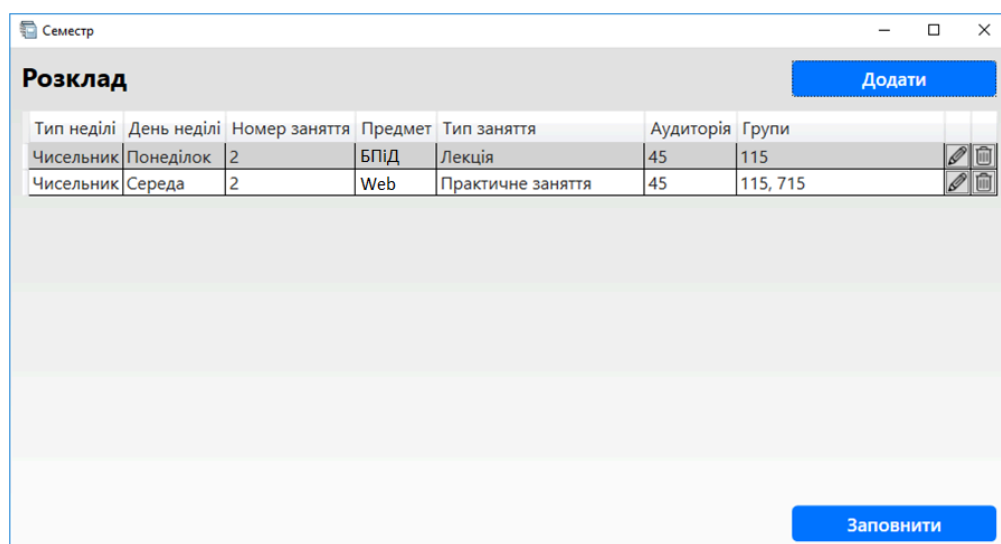


Назва	
45	
57	

Додати Зберегти

Рисунок 4.5 – Форма з інформацією про аудиторії

При натисканні на кнопку Розклад відбувається перехід до форми з розкладом викладача (рис. 4.6). Формування розкладу відбувається шляхом додавання кожного заняття, для чого необхідно натиснути кнопку Додати та ввести день проведення відповідного заняття, аудиторію та навчальну групу.



Тип неділі	День неділі	Номер заняття	Предмет	Тип заняття	Аудиторія	Групи	
Чисельник	Понеділок	2	БПід	Лекція	45	115	
Чисельник	Середа	2	Web	Практичне заняття	45	115, 715	

Додати

Заповнити

Рисунок 4.6 – Форма формування розкладу викладача

При натисканні на кнопку Заповнити відбувається автоматичне формування таблиці всіх занять викладача (рис. 4.7) на весь семестр згідно розкладу.

Журнал викладача

Дані

Семестр Заняття

Семестр 1 Дата з 02.09.2024 по 06.12.2024 **Сьогодні** **Семестр**

Дисципліни

Групи

Аудиторії

Розклад

в ДОС

Дата	№ Пари	Тип	Аудиторія	Групи	Предмет	Тема	
02.09	2	Лекція	45	115	БПід		
04.09	2	Практичне заняття	57	115, 715	Web		
05.09	3	Лекція	45	115, 415, 7	Web		
06.09	1	Практичне заняття	57	115	БПід		
12.09	3	Лекція	45	115, 415, 7	Web		
13.09	1	Практичне заняття	57	115	БПід		
16.09	2	Лекція	45	115	БПід		
18.09	1	Лабораторне заняття	45	115	Web		
18.09	2	Практичне заняття	57	115, 715	Web		
19.09	3	Лекція	45	115, 415, 7	Web		
20.09	1	Практичне заняття	57	115	БПід		
26.09	3	Лекція	45	115, 415, 7	Web		
27.09	1	Практичне заняття	57	115	БПід		
30.09	2	Лекція	45	115	БПід		
02.10	2	Практичне заняття	57	115, 715	Web		

Дані викладача **Додати заняття**

Рисунок 4.7 – Головна форма програмного забезпечення для обліку викладацької роботи після формування розкладу викладача

Після формування розкладу можна створити журнал обліку викладацької роботи шляхом натискання кнопки «в дос», в наслідок чого буде сформовано файл у форматі Microsoft Word, що буде являти собою журнал обліку роботи викладача (рис. 4.8, рис. 4.9), який створено на основі його розкладу.

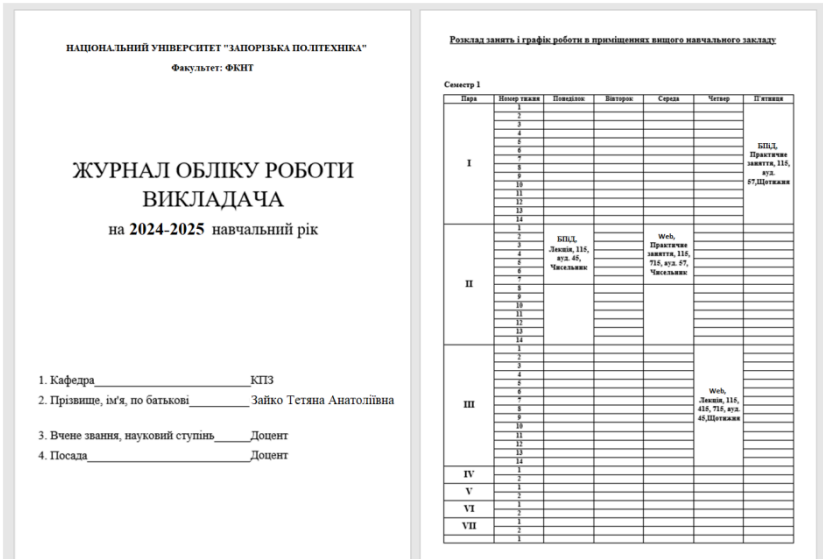


Рисунок 4.8 – Перші сторінки сформованого журналу обліку викладацької роботи

1. Облік виконання навчальної роботи викладача					
Дата	Шифр розкладу (загальної групи)	Назва навчальної дисципліни	Тема заняття	Вид заняття	К-ть годин
02.09	115	БТД		Лекція	2
04.09	115, 715	Web		Практичне заняття	2
05.09	115, 415, 715	Web		Лекція	2
06.09	115	БТД		Практичне заняття	2
12.09	115, 415, 715	Web		Лекція	2
13.09	115	БТД		Практичне заняття	2
14.09	115	БТД		Лекція	2
18.09	115, 715	Web		Практичне заняття	2
18.09	115	Web		Лабораторна заняття	2
19.09	115, 415, 715	Web		Лекція	2
20.09	115	БТД		Практичне заняття	2
26.09	115, 415, 715	Web		Лекція	2
27.09	115	БТД		Практичне заняття	2
29.09	115	БТД		Лекція	2
02.10	115, 715	Web		Практичне заняття	2
03.10	115, 415, 715	Web		Лекція	2
04.10	115	БТД		Практичне заняття	2
10.10	115, 415, 715	Web		Лекція	2
11.10	115	БТД		Практичне заняття	2
14.10	115	БТД		Лекція	2
16.10	115, 715	Web		Практичне заняття	2
17.10	115, 415, 715	Web		Лекція	2
18.10	115	БТД		Практичне заняття	2
24.10	115, 415, 715	Web		Лекція	2
25.10	115	БТД		Практичне заняття	2
28.10	115	БТД		Лекція	2
30.10	115, 715	Web		Практичне заняття	2
31.10	115, 415, 715	Web		Лекція	2
01.11	115	БТД		Практичне заняття	2
07.11	115, 415, 715	Web		Лекція	2
08.11	115	БТД		Практичне заняття	2
11.11	115	БТД		Лекція	2
13.11	115, 715	Web		Практичне заняття	2

14.11	115, 415, 715	Web		Лекція	2
15.11	115	БТД		Практичне заняття	2
21.11	115, 415, 715	Web		Лекція	2
22.11	115	БТД		Практичне заняття	2
23.11	115	БТД		Лекція	2
27.11	115, 715	Web		Практичне заняття	2
28.11	115, 415, 715	Web		Лекція	2
29.11	115	БТД		Практичне заняття	2
05.12	115, 415, 715	Web		Лекція	2

Завідувач кафедри _____ Сергій СУББОТІН
Викладач _____ Тетяна ЗАЙКО

Рисунок 4.9 – Розділ обліку викладацької роботи у сформованому програмою документах

4.5 Тестування програмного забезпечення для обліку викладацької роботи

Виконано тестування програмного забезпечення для обліку викладацької роботи.

Виявлено, що програма для обліку викладацької роботи функціонує правильно та злагоджено. Вона реалізує всі функціональні вимоги та успішно виконує свою основну задачу підтримки процесу обліку роботи викладача.

Розроблене програмне забезпечення дозволяє забезпечувати автоматизацію процесів, пов'язаних з підтримкою процесу обліку викладацької роботи.

4.6 Висновки за розділом 4

Описано програмне забезпечення для обліку викладацької роботи. Виконано тестування розробленого програмного забезпечення для обліку викладацької роботи. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою процесу обліку викладацької роботи.

Висновки

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для обліку викладацької роботи.

Визначено, що розробка програмного забезпечення для освітньої діяльності та обліку викладацької роботи активно змінює традиційні підходи до навчального процесу, забезпечуючи більш індивідуалізоване, інтерактивне та динамічне навчання. Впровадження новітніх технологій сприяє створенню ефективних інструментів для онлайн освіти, а також оптимізації адміністративних функцій. Сучасні програмні засоби дозволяють удосконалювати методи навчання і значно полегшують облік викладацької роботи, завдяки чому вчителі можуть більше зосереджуватись на навчальному процесі, а адміністрація – на керуванні даними.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для обліку викладацької роботи. Проте деякі програмні засоби для обліку викладацької роботи можуть бути досить складними для налаштування та адаптації системи до специфіки конкретного навчального закладу. Кожен заклад має власні вимоги до обліку видів діяльності та звітності, і далеко не кожне програмне забезпечення здатне гнучко налаштуватися під ці особливості без залучення фахівців або значних витрат часу. Ще одним викликом є потреба в навчанні персоналу, крім того, інтерфейс деяких програм для обліку викладацької роботи може бути незручним або перевантаженим, що ускладнює роботу користувачів і знижує загальну ефективність використання. З технічного боку, програмне забезпечення для обліку викладацької роботи може мати проблеми з надійністю або сумісністю. У випадку локального встановлення ПЗ для обліку викладацької роботи додатковою складністю є потреба в обслуговуванні та резервному копіюванні, що вимагає технічного персоналу або залучення

сторонніх спеціалістів. Тому актуальною є розробка програмного забезпечення для обліку викладацької роботи.

Сформульовано функціональні вимоги до програмного забезпечення для обліку викладацької роботи.

Для реалізації програмного забезпечення для обліку викладацької роботи обрано мову програмування C#, яка підтримує об'єктно-орієнтоване програмування, модульність, повторне використання коду, інкапсуляцію та інші принципи, що значно спрощують проєктування складних програмних систем. Завдяки платформі .NET розробник на мові C# отримує доступ до великої кількості бібліотек, інструментів і фреймворків, які дозволяють реалізовувати функціональність для роботи з базами даних, створення графічного або вебінтерфейсу, а також реалізацію логіки бізнес-процесів.

Для створення програмного забезпечення для обліку викладацької роботи обрано середовище розробки Visual Studio, що надає зручні інструменти для повного циклу розробки – від проєктування до тестування, налагодження та розгортання програмного продукту. Visual Studio має вбудовану систему підказок, авто доповнення коду, інтегрований налагоджувач коду, а також потужні засоби для візуального проєктування інтерфейсу користувача, що значно пришвидшує розробку та зменшує кількість помилок.

Розроблено програмне забезпечення для обліку викладацької роботи.

Запропоновано структуру програмного забезпечення для обліку викладацької роботи. У програмі для обліку викладацької роботи реалізовано кілька взаємопов'язаних функціональних елементів, кожен з яких виконує окрему роль у загальній логіці роботи програмного забезпечення. Кожен елемент програмного забезпечення для обліку викладацької роботи побудовано таким чином, щоб забезпечити чітку функціональну ізоляцію і водночас безперервну інтеграцію з іншими компонентами, що дозволяє реалізувати злагоджену роботу всієї системи.

За допомогою діаграми прецедентів описано функціонування програмного забезпечення для обліку викладацької роботи. У межах діаграми прецедентів програмного забезпечення, призначеного для обліку викладацької діяльності, визначається основна роль користувача, яка охоплює взаємодію з ключовими функціональними можливостями системи. Діаграма прецедентів подає цілісну картину взаємодії користувача з усіма компонентами програмної системи, відображаючи логіку та сценарії використання програмного забезпечення в процесі обліку викладацького навантаження.

Розроблено базу даних програмного забезпечення для обліку викладацької роботи, яка призначена для зберігання, впорядкування та зв'язування інформації, що стосується семестрів, навчальних занять, розкладу, предметів, аудиторій, груп та інших супровідних об'єктів. Вона забезпечує централізоване управління даними, зменшує ризики помилок під час обробки інформації та дає змогу ефективно формувати звіти про навантаження викладачів. Архітектура бази даних побудована за принципами реляційної моделі, де інформація структурована у вигляді взаємопов'язаних таблиць.

Описано особливості реалізації програмного забезпечення для обліку викладацької роботи. Виконано проектування інтерфейсу взаємодії користувача з програмним забезпеченням для обліку викладацької роботи.

Виконано тестування розробленого програмного забезпечення для обліку викладацької роботи. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою процесу обліку викладацької роботи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Complete Guide to Education Software Development [Electronic resource]. – Access mode: <https://www.sphinx-solution.com/blog/education-software-development/>.
2. List of Different Types of Educational Software [Electronic resource]. – Access mode: <https://www.tatvasoft.com/outsourcing/2021/12/types-of-educational-software.html>.
3. A Comprehensive Guide to Learning Management System Development [Electronic resource]. – Access mode: <https://infostride.com/guide-to-learning-management-system-lms-development/>.
4. Learning Management System development: 8 steps [Electronic resource]. – Access mode: <https://syndicode.com/blog/how-to-build-a-learning-management-system/>.
5. A Complete Guide on eLearning Software Development [Electronic resource]. – Access mode: <https://www.matellio.com/blog/elearning-software-development-guide/>.
6. Best Software for Teachers [Electronic resource]. – Access mode: <https://www.shoeboxed.com/blog/best-software-for-teachers>.
7. Best software tools for teachers [Electronic resource]. – Access mode: <https://paperform.co/blog/software-for-teachers/>.
8. List of top data software for teachers [Electronic resource]. – Access mode: <https://www.skoolbeep.com/blog/best-data-tracking-apps-for-teachers/>.
9. Comparison of accounting software for schools, academies and MATs [Electronic resource]. – Access mode: <https://www.theaccessgroup.com/en-gb/education/software/school-finance/comparison-of-accounting-software-for-schools-academies-and-mats/>.
10. Best School Accounting Software [Electronic resource]. – Access mode: <https://thecfoclub.com/tools/best-school-accounting-software/>.

11. Best Classroom Management Software for Teachers [Electronic resource]. – Access mode: <https://geekflare.com/software/classroom-management-software/>.
12. Schoology [Electronic resource]. – Access mode: <https://www.powerschool.com/personalized-learning-cloud/schoology-learning/>.
13. Learnidy [Electronic resource]. – Access mode: <https://alpha.learnidy.com/>.
14. NetSupport School [Electronic resource]. – Access mode: <https://www.netsupportschool.com/>.
15. A tour of the C# language [Electronic resource]. – Access mode: <https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/overview>.
16. C# Type Casting [Electronic resource]. – Access mode: https://www.w3schools.com/cs/cs_type_casting.php.
17. Visual Studio Code [Electronic resource]. – Access mode: <https://vscode.dev/?vscode=&vscode>.
18. .NET SDKs for Visual Studio [Electronic resource]. – Access mode: <https://dotnet.microsoft.com/en-us/download/visual-studio-sdks>.

Додаток А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для обліку викладацької роботи.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмне забезпечення для обліку викладацької роботи», затверджене наказом Національного університету «Запорізька політехніка» № 209 від 28 квітня 2025 р.

A.2 Призначення розробки

Програмний продукт призначений для обліку роботи викладача.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення для обліку викладацької роботи забезпечує виконання таких функцій:

- підтримка можливості редагування інформації про викладача;
- підтримка можливості керування інформацією про викладацьку діяльність, зокрема, про дисципліни, які викладає викладач, а також навчальні групи та аудиторії, де проводяться заняття;
- можливість редагування розкладу викладача;
- можливість формування таблиці занять викладача згідно розкладу;
- підтримка можливості створення журналу обліку викладацької роботи;

– підтримка можливості відображення занять викладача згідно обраних за допомогою календаря дат.

A.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення для обліку викладацької роботи повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі.

A.3.3 Вимоги до надійності

Програмне забезпечення для обліку викладацької роботи повинно забезпечити надійне функціонування.

A.3.4 Умови експлуатації

Для експлуатації програмного забезпечення для обліку викладацької роботи необхідна наявність персонального комп'ютера.

A.3.5 Вимоги до складу та параметрів технічний засобів

Для роботи програмного забезпечення для обліку викладацької роботи потрібні такі технічні та програмні ресурси, які забезпечують стабільну і безперебійну обробку інформації у процесі взаємодії з базою даних і графічним інтерфейсом користувача. Мінімально допустимий рівень обчислювальної потужності передбачає наявність центрального процесора з тактовою частотою не менше 2.0 ГГц, що дає змогу оперативно виконувати запити та обробляти події в реальному часі. Обсяг оперативної пам'яті

повинен бути не менше 4 ГБ для забезпечення комфортної роботи з великими обсягами інформації та паралельного виконання кількох модулів. Обсяг вільного місця на жорсткому диску має забезпечувати збереження бази даних і допоміжних файлів, тому рекомендоване значення — від 500 МБ. Графічний адаптер повинен забезпечувати роботу з інтерфейсами, підтримка роздільної здатності екрана від 1024×768 також є бажаною для коректного відображення вікон і таблиць. Необхідно забезпечити стабільне мережеве підключення.

А.3.6 Вимоги до маркування і пакування

Програма для обліку викладацької роботи може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення для обліку викладацької роботи».

А.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення для обліку викладацької роботи є інформація про викладачів, здобувачів, дисципліни, аудиторії та ін.

Вихідними даними програмного забезпечення для обліку викладацької роботи є створені документи, зокрема, розклад викладача та журнал його роботи.

Додаток Б
Фрагмент тексту програми

```

private void OpenSlvnk(object sender, RoutedEventArgs e)
{
    var btame = ((Button)sender).Name;
    int vcblrId = 0;
    if (crntTrm != null)
    {
        switch (btame)
        {
            case "btSbjct": vcblrId = SlvnkWindow.SBJCT; break;
            case "btKlsrm": vcblrId = SlvnkWindow.KLSRM; break;
            case "btGrp": vcblrId = SlvnkWindow.GRP; break;
        }
        SlvnkWindow slvnk = new SlvnkWindow(vcblrId, crntTrm);
        try
        {slvnk.ShowDialog();}
        }
    }
    else {
        OpenAddSemestrWindow();
    }
}

private void OpenAddSemestrWindow()
{
    SemestrWindow semstr = new SemestrWindow();
    try
    {
        if (semstr.ShowDialog() == true)
        {
            Semestr newSemestr = DBhlpr.getLastSemestr();
            semestrList.Add(newSemestr);
            cbSemestrList.SlctdItm = newSemestr;
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine("{0} Помилка ", ex);
    }
}

private void btAddNewSemestr_Click(object sender, RoutedEventArgs e)
{
    OpenAddSemestrWindow();
}

private void btRozkld_Click(object sender, RoutedEventArgs e)
{
    if (crntTrm != null)
    {
        RozkldWindow Rozkld = new RozkldWindow(crntTrm);
        try

```

```

{
if (Rozkld.ShowDialog() == true)
{
UpdateZanyatList();
}}catch (Exception ex)
{
Console.WriteLine("{0} Помилка ", ex);
}}
else
{
OpenAddSemestrWindow();
}}
private void cbSemstrList_SelectionChanged(object sender,
SelectionChangedEventArgs e)
{
Semestr semestr = (Semestr)cbSemstrList.SlctdItm;
if(semestr != null)
{
crntTrm = semestr;
if ((strtDtVl == crntTrm.bgnDt) && (endtDtVl == crntTrm.enDt))
{
UpdateZanyatList();
}
else
{
dpStrtDt.SlctdDt = crntTrm.bgnDt;
dpEnDt.SlctdDt = crntTrm.enDt;
}}}
private void ShowAllSemestr(object sender, RoutedEventArgs e)
{
if (crntTrm != null)
{
dpStrtDt.SlctdDt = crntTrm.bgnDt;
dpEnDt.SlctdDt = crntTrm.enDt;
}}
private void ShowToday(object sender, RoutedEventArgs e)
{
if (crntTrm != null)
{
dpStrtDt.SlctdDt = DateTime.Today;
dpEnDt.SlctdDt = DateTime.Today;
}}
public void UpdateZanyatList()
{
if (crntTrm != null)
{
strtDtVl = dpStrtDt.SlctdDt.Value.Date;

```

```

endtDtV1 = dpEnDt.SlctdDt.Value.Date;
List<Zanyat> tempZanyats = DBhlpr.selectZanyats(crntTrm,
strtDtV1, endtDtV1);
if (tempZanyats.Count > 0)
{
    zanyatList = new
ObservableCollection<Zanyat>(tempZanyats.OrderBy(l =>
l.date).ThenBy(l => l.numOfZanyat).ToList()));
    zanyatsGrid.ItmsSrc = zanyatList;
}
else {
    zanyatList = new ObservableCollection<Zanyat>(tempZanyats);
    zanyatsGrid.ItmsSrc = null;
}}}
private void DeleteRow_Click(object sender, RoutedEventArgs e)
{
    if (MessageBox.Show("Видаляємо запис?", "Approve",
MessageBoxButton.YesNo) == MssgBxRslt.Yes)
    {
        try
        {
            Zanyat obj = ((FrmwrkElmnt)sender).Dtkntx as Zanyat;
            zanyatList.Remove(objekt);
            DBhlpr.deleteZanyat(objekt);
        }
        catch (Exception ex)
        {
            Console.WriteLine("{0} Помилка", ex);
        }
    }
}
private void EditRow_Click(object sender, RoutedEventArgs e)
{
    Zanyat objekt = ((FrmwrkElmnt)sender).Dtkntx as Zanyat;
    OpenZanyatItemWindow(objekt);
}
private void Row_DoubleClick(object sender, MouseButtonEventArgs
e)
{
    if (e.ChangedButton == MouseButton.Left)
    {
        DataGridRow row = sender as DataGridRow;
        Zanyat objekt = ((FrmwrkElmnt)sender).Dtkntx as Zanyat;
        OpenZanyatItemWindow(objekt);
    }
    e.Handled = true;
}
private void OpenZanyatItemWindow(Zanyat ls)
{

```

```

ZanyatItemWindow window = new ZanyatItemWindow(ls, crntTrm,
this);
try
{
window.ShowDialog();
UpdateZanyatList();
}
catch (Exception ex)
{
Console.WriteLine("{0} Помилка ", ex);
}}
private void dp_SlctdDtChanged(object sender,
SelectionChangedEventArgs e)
{
DtPckr dtPckr = (DtPckr)sender;
String sndrNm = dtPckr.Name;
DateTime dtVl = dtPckr.SlctdDt.Value.Date;
if(sndrNm == "dpStrtDt")
{
if (strtDtVl != dtVl)
{
strtDtVl = dtVl;
if (endDtVl >= strtDtVl)
{
UpdateZanyatList();
}}}}
else if (sndrNm == "dpEnDt")
{
if (endDtVl != dtVl)
{
endDtVl = dtVl;
if (endDtVl >= strtDtVl)
{
UpdateZanyatList();
}
}}}}
private void dpPrvwTxtInpt(object sender,
TextCompositionEventArgs e)
{
Validators.DtPckrValidation(sender, e);
}
private void ClearSemestrs_Click(object sender, RoutedEventArgs
e)
{
if (MessageBox.Show("Потрібно очистити файл бд?", "Approve",
MessageBoxButton.YesNo) == MssgBxRslt.Yes){
try

```

```

{DBhlpr.ClearSemestrs();
semestrList.Clear();
zanyatList.Clear();
}
catch (Exception ex){
Console.WriteLine("{0} Помилка ", ex);
}}
private void OpenVyklchInfo_Click(object sender, RoutedEventArgs e)
{
VyklchInfoWindow window = new VyklchInfoWindow();
window.ShowDialog();
}
private void btEditSemstr_Click(object sender, RoutedEventArgs e)
{
if (crntTrm != null)
{
SemstrWindow window = new SemstrWindow(crntTrm);
if (window.ShowDialog() == true){
cbSemstrList.SlctdItm = null;
cbSemstrList.SlctdItm = crntTrm;
dpStrtDt.SlctdDt = crntTrm.bgnDt;
dpEnDt.SlctdDt = crntTrm.enDt;
}}
}
private void btToDoc_Click(object sender, RoutedEventArgs e)
{
ExprtWndw ExprtWndw = new ExprtWndw();
ExprtWndw.ShowDialog();
}
public void btAddZanyat(object sender, RoutedEventArgs e)
{
if (crntTrm != null){
OpenZanyatItemWindow(null);
}
else{
OpenAddSemestrWindow();
}}
}

private void btAccept_Click(object sender, RoutedEventArgs e){
if ((dpSemstrStartDate.SlctdDt != null) && (dpSemstrEnDt.SlctdDt
!= null) && (tbxSemstrName.Text != "")){
String semestrName = tbxSemstrName.Text;
DateTime semestrStartDate = dpSemstrStartDate.SlctdDt.Value.Date;
DateTime semestrEnDt = dpSemstrEnDt.SlctdDt.Value.Date;
int strtWthNmtr = chbStrtWthNmtr.IsChecked.Value ? 1 : 0;
if (crntTrm == null)

```

```

{
if (MessageBox.Show("Виконуємо створення нового семестру?",
"Approve", MessageBoxButton.YesNo) == MssgBxRslt.Yes){
crntTrm = new Semestr(Semestr.ID_FOR_WRITING, semestrName,
semestrStartDate, semestrEnDt, strtWthNmtrtr);
try{
DBhlpr.addSemestr(crntTrm);
}
catch (Exception ex){
Console.WriteLine("{0} Помилка ", ex);
}
this.DlgRslt = true;
}}
else
{
MssgBxRslt result = MessageBox.Show("Оновити дані?", "Approve",
MessageBoxButton.YesNo);
if (result == MssgBxRslt.Yes){
NeedToUpdateZanyats = true;
}
else if (result == MssgBxRslt.No){
NeedToUpdateZanyats = false;
}
else return;
crntTrm.name = semestrName;
crntTrm.bgnDt = semestrStartDate;
crntTrm.enDt = semestrEnDt;
crntTrm.startFromNumerator = strtWthNmtrtr;
try
{
DBhlpr.updateSemestr(crntTrm);
if (NeedToUpdateZanyats)
{
List<Rozkld> rozkldList = DBhlpr.selectRozklds(crntTrm);
DBhlpr.ClearZanyat(crntTrm);
DBhlpr.addZanyats(Zanyat.generateZanyats(rozkldList, crntTrm));
}
}
catch (Exception ex){
Console.WriteLine("{0} Помилка ", ex);
}
this.DlgRslt = true;
}}
else{
MessageBox.Show("Не все заповнено!", "Warning");
}}
private void dpPrvwTxtInpt(object sender,
TextCompositionEventArgs e){

```

```

Validators.DtPckrValidation(sender, e);
}
private void Window_Loaded(object sender, RoutedEventArgs e){
if (crntTrm != null){
tbxSemstrName.Text = crntTrm.name;
dpSemstrStartDate.SlctdDt = crntTrm.bgnDt;
dpSemstrEndDt.SlctdDt = crntTrm.enDt;
chbStrtWthNmtrtr.IsChecked = crntTrm.startFromNumerator == 1 ?
true : false;
}}}}

private void btFillRozkld_Click(object sender, RoutedEventArgs e)
{
if (MessageBox.Show("Прийняти розклад?", "Approve",
MessageBoxButton.YesNo) == MssgBxRslt.Yes){
if (Rozkld.isRozkldsCorrect(rozkldList.ToList())){
BackgroundRobter bg = new BackgroundRobter();
bg.DoRobt += new DoRobtEventHandler(bg_DoRobt);
bg.RunRobterCompleted += new
RunRobterCompletedEventHandler(bg_RunRobterCompleted);
bg.RunRobterAsync();
ldngFrm = new LdngFrm("Заповнення журналу занять");
ldngFrm.ShowDialog();}
else{
MessageBox.Show("Проблеми з заповненням журналу", "Помилка");
}}}}
private void bg_DoRobt(object sender, DoRobtEventArgs e)
{
try
{
DBhlpr.ClearRozkld(crntTrm);
DBhlpr.ClearZanyat(crntTrm);
DBhlpr.addRozklds(rozkldList.ToList());
DBhlpr.addZanyats(Zanyat.generateZanyats(rozkldList.ToList(),
crntTrm));
}
catch (Exception ex){
Console.WriteLine("{0} Помилка ", ex);
}}
private void bg_RunRobterCompleted(object sender,
RunRobterCompletedEventArgs e)
{
ldngFrm.RobtEnded = true;
ldngFrm.Close();
this.DlgRslt = true;
}

```

```

private void btAddNewRozkldRow_Click(object sender,
RoutedEventArgs e)
{
RozkldItemWindow addRozkldItem = new RozkldItemWindow(crntTrm,
this);
try
{
addRozkldItem.ShowDialog();
updateRozkldList();
}
catch (Exception ex)
{
Console.WriteLine("{0} Помилка ", ex);
}}
private void DeleteRow_Click(object sender, RoutedEventArgs e)
{try{
Rozkld obekt = ((FrmwrkElmnt)sender).Dtkntx as Rozkld;
rozkldList.Remove(obekt);}
catch (Exception ex){
Console.WriteLine("{0} Помилка", ex);
}}

```

```

public RozkldItemWindow(Semestr semestr, RozkldWindow window,
Rozkld rozkld = null)
{
InitializeComponent();
this.Title = ADD_TITLE;
this.crntTrm = semestr;
this.window = window;
this.poochnRozkld = rozkld;
try
{
typeTygn = new List<TypeTygn>(DBhlpr.selectTypesTygn());
dayTygn = new List<model.DayTygn>(DBhlpr.selectDaysTygn());
typeOfZanyat = new
List<TypeOfZanyat>(DBhlpr.selectTypesOfZanyat());
Sbjct = new List<Sbjct>(DBhlpr.selectSbjct(crntTrm));
Sbjct.Sort((p, q) => p.name.CompareTo(q.name));
klrm = new List<Klrm>(DBhlpr.selectKlrm(crntTrm));
klrm.Sort((p, q) => p.name.CompareTo(q.name));
grps = new List<Grp>(DBhlpr.selectGrps(crntTrm));
grps.Sort((p, q) => p.name.CompareTo(q.name));
}
catch (Exception ex)
{
Console.WriteLine("{0} Помилка", ex);
}

```

```

}}
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    cbDayTygn.ItmsSrc = dayTygn;
    cbTypeTygn.ItmsSrc = typeTygn;
    cbTypeOfZanyat.ItmsSrc = typeOfZanyat;
    cbSbjct.ItmsSrc = Sbjct;
    cbKlsrm.ItmsSrc = klsrm;
    cbNumberOfZanyat.ItmsSrc = numsOfZanyats;
    if (poochnRozkld != null)
    {
        this.Title = EDIT_TITLE;
        foreach (model.DayTygn item in cbDayTygn.Items)
        {
            if (item.id == poochnRozkld.dayTygn.id)
            {
                cbDayTygn.SlctdItm = item;
                break;
            }
        }
        foreach (TypeTygn item in cbTypeTygn.Items)
        {
            if (item.id == poochnRozkld.typeTygn.id)
            {
                cbTypeTygn.SlctdItm = item;
                break;
            }
        }
        foreach (TypeOfZanyat item in cbTypeOfZanyat.Items)
        {
            if (item.id == poochnRozkld.typeOfZanyat.id)
            {
                cbTypeOfZanyat.SlctdItm = item;
                break;
            }
        }
        foreach (Sbjct item in cbSbjct.Items)
        {
            if (item.id == poochnRozkld.Sbjct.id)
            {
                cbSbjct.SlctdItm = item;
                break;
            }
        }
        foreach (Klsrm item in cbKlsrm.Items)
        {
            if (item.id == poochnRozkld.klsrm.id)
            {
                cbKlsrm.SlctdItm = item;
                break;
            }
        }
    }
}

```

```

foreach (Grp grp in poochnRozkld.grps)
{
AddNewGrp(grp);
}
cbNumberOfZanyat.SlctdItm =
cbNumberOfZanyat.Items[poochnRozkld.numOfZanyat - 1];
}
else
{
AddNewGrp();
private void AcceptAndSave(object sender, RoutedEventArgs e)
{
if ((cbTypeTygn.SlctdItm != null) && (cbTypeOfZanyat.SlctdItm !=
null) && (cbDayTygn.SlctdItm != null)
&& (cbSbjct.SlctdItm != null) && (cbKlsrm.SlctdItm != null) &&
AreGrpsFilled() && (cbNumberOfZanyat.SlctdItm != null))
{
Rozkld rozkld = new Rozkld();
rozkld.id = this.window.tempId--;
rozkld.typeTygn = cbTypeTygn.SlctdItm as TypeTygn;
rozkld.typeOfZanyat = cbTypeOfZanyat.SlctdItm as TypeOfZanyat;
rozkld.dayTygn = cbDayTygn.SlctdItm as model.DayTygn;
rozkld.numOfZanyat = (Int32) cbNumberOfZanyat.SelectedValue;
rozkld.Sbjct = cbSbjct.SlctdItm as Sbjct;
rozkld.klsrm = cbKlsrm.SlctdItm as Klsrm;
rozkld.idSemestr = crntTrm.id;
rozkld.fieldForSort =
Rozkld.calculateFieldForSort(rozkld.dayTygn, rozkld.numOfZanyat);
rozkld.grps = new List<Grp>();
foreach (StckPnl child in GrpVerticalPanel.Children){
foreach (object _child in child.Children){
if (_child.GetType().Name == "ComboBox"){
Grp grp = ((ComboBox)_child).SlctdItm as Grp;
rozkld.grps.Add(grp);
break;
}}}var list = this.window.rozkldList;
if (poochnRozkld == null){
list.Add(rozkld);
}
else{
for (int i = 0; i < list.Count; i++){
if (list.ElementAt(i).id == poochnRozkld.id){
list[i] = rozkld;
}}}
this.Close();
}
else{

```

```

MessageBox.Show("He все заповнено!", "Warning");
}}
private bool AreGrpsFilled()
{
    if (GrpVerticalPanel.Children.Count != 0){
        foreach (StckPnl child in GrpVerticalPanel.Children){
            foreach (object _child in child.Children)
            {
                if (_child.GetType().Name == "ComboBox"){
                    ComboBox cb = (ComboBox)_child;
                    if (cb.SlctdItm == null)return false;
                    break;
                }}return true;
            }
        }
        return false;
    }
    private void btAddNewGrp_Click(object sender, RoutedEventArgs e)
    {
        AddNewGrp();
    }
    private void btDeleteGrp_Click(object sender, RoutedEventArgs e){
        StckPnl StckPnl = (StckPnl)((Button)sender).Parent;
        GrpVerticalPanel.Children.Remove(StckPnl);}
    private void AddNewGrp(Grp grp = null)
    {
        StckPnl StckPnl = new StckPnl();
        StckPnl.Orientation = Orientation.Horizontal;
        ComboBox cbGrp = new ComboBox();
        cbGrp.Width = 130;
        cbGrp.ItmsSrc = grps;
        cbGrp.DisplayMemberPath = "name";
        cbGrp.SelectedValuePath = "id";
        Button bt = new Button();
        bt.Content = (Image)FindResource("TrashBoxImage");
        bt.Margin = new Thickness(5, 0, 0, 0);
        bt.Click += new RoutedEventHandler(btDeleteGrp_Click);
        if (grp != null){
            foreach (Grp item in cbGrp.Items){
                if (item.id == grp.id){
                    cbGrp.SlctdItm = item;
                    break;
                }}
        }
        else
        {
            if (cbGrp.Items.Count > 0)
                cbGrp.SlctdItm = cbGrp.Items[0];
        }
    }

```

```
StckPnl.Children.Add(cbGrp);  
StckPnl.Children.Add(bt);  
GrpVerticalPanel.Children.Add(StckPnl);  
}
```

Додаток В
Слайди презентації

**Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів**

Програмне забезпечення для обліку викладацької роботи

Виконав: Олександр НУЖНИЙ

ст. групи КНТ-112сп

Керівник: Тетяна ЗАЙКО

к.т.н., доцент

Рисунок В.1 – Слайд 1

Об'єкт, предмет та мета роботи

- Об'єкт дослідження – процес розробки програмного забезпечення для обліку викладацької роботи.
- Предмет дослідження – програмні засоби для обліку викладацької роботи.
- Метою роботи є розробка програмного забезпечення для обліку викладацької роботи.

2

Рисунок В.2 – Слайд 2

Завдання роботи

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для обліку викладацької роботи;
- здійснити проєктування програмного забезпечення для обліку викладацької роботи;
- створити програмне забезпечення для обліку викладацької роботи;
- виконати тестування розробленого програмного забезпечення для обліку викладацької роботи.

3

Рисунок В.3 – Слайд 3

Порівняння існуючих аналогів

Критерій порівняння	Schoology	Learnidy	NetSupport School
Зручність інтерфейсу	+–	+	+–
Підтримка онлайн комунікації	+	+–	+
Наявність шаблонів для занять	+–	+	–
Функції моніторингу студентів	+–	–	+
Інтеграція з іншими сервісами	+	+–	+–
Прозорість ціноутворення	–	+–	+–

4

Рисунок В.4 – Слайд 4

Постановка завдання

При розробці програмного забезпечення для обліку викладацької роботи необхідно забезпечити такі функціональні вимоги:

- підтримка можливості редагування інформації про викладача;
- підтримка можливості керування інформацією про викладацьку діяльність, зокрема, про дисципліни, які викладає викладач, а також навчальні групи та аудиторії, де проводяться заняття;
- можливість редагування розкладу викладача;
- можливість формування таблиці занять викладача згідно розкладу;
- підтримка можливості створення журналу обліку викладацької роботи;
- підтримка можливості відображення занять викладача згідно обраних за допомогою календаря дат.

5

Рисунок В.5 – Слайд 5

Порівняння мов програмування

Критерій порівняння мов програмування	Мова програмування		
	C#	Python	Java
Підтримка графічного інтерфейсу	+	+–	+–
Продуктивність виконання	+	–	+
Документація та підтримка	+	+–	+
Зручність для створення ПЗ для обліку викладацької роботи	+	–	–
Наявність готових бібліотек та фреймворків	+	+–	+

6

Рисунок В.6 – Слайд 6

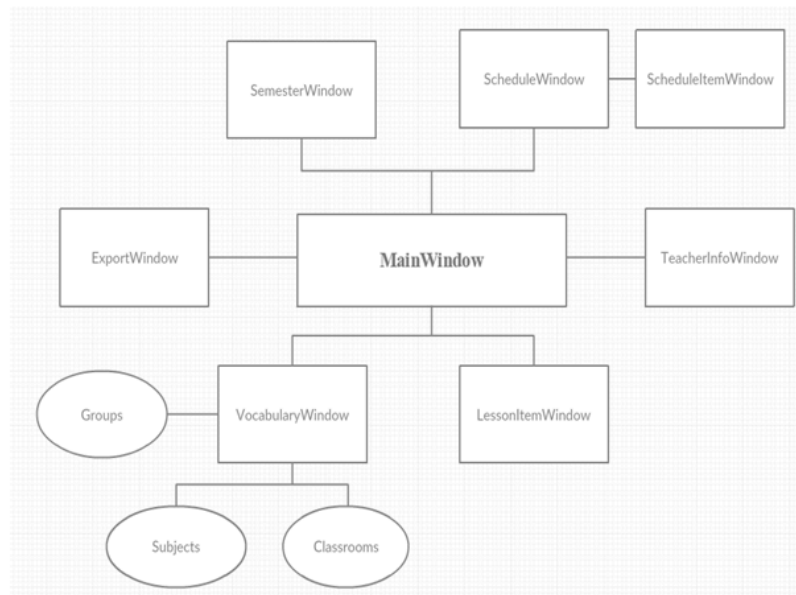
Порівняння середовищ розробки

Критерій порівняння середовищ розробки	Середовища розробки		
	Visual Studio	JetBrains Rider	MonoDevelop
Повна підтримка C# та .NET	+	+	+—
Вбудований дизайнер інтерфейсу	+	+—	+—
Зручність інструментів налагодження	+	+	+—
Підтримка розширень та плагінів	+	+—	+—
Засоби автоматичного тестування	+	+	+—

7

Рисунок В.7 – Слайд 7

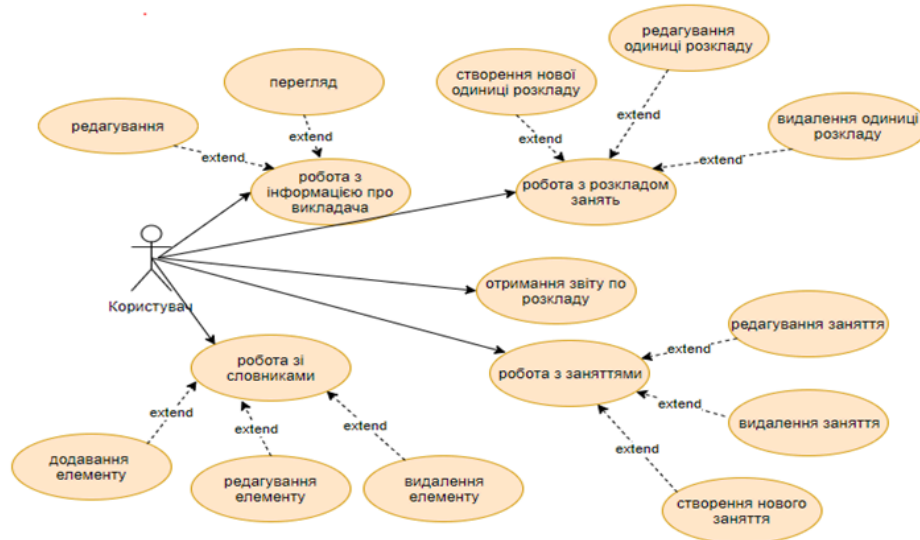
Структура програми



8

Рисунок В.8 – Слайд 8

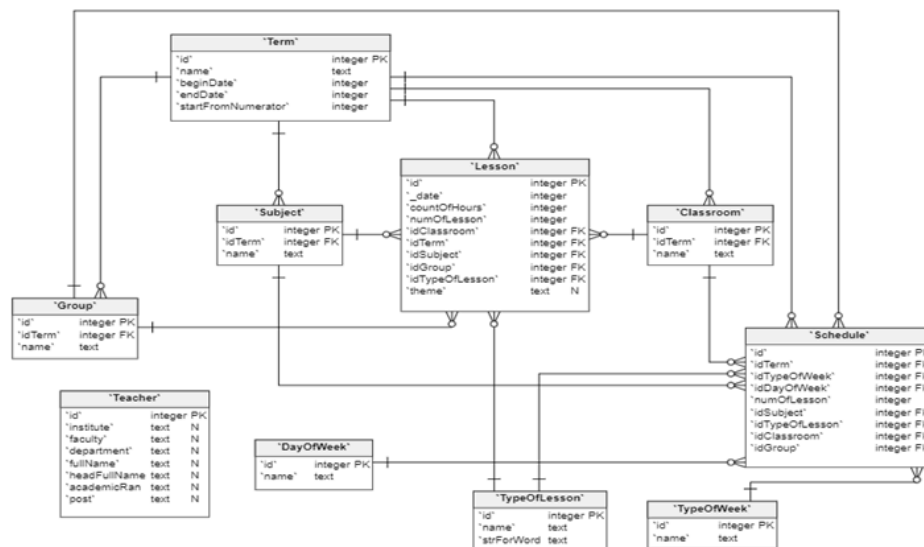
Діаграма прецедентів програмного забезпечення для обліку викладацької роботи



9

Рисунок В.9 – Слайд 9

Схема бази даних програмного забезпечення для обліку викладацької роботи



10

Рисунок В.10 – Слайд 10

Інтерфейс програми. Головна форма. Робота зі словниками даних про викладачів та дисципліни

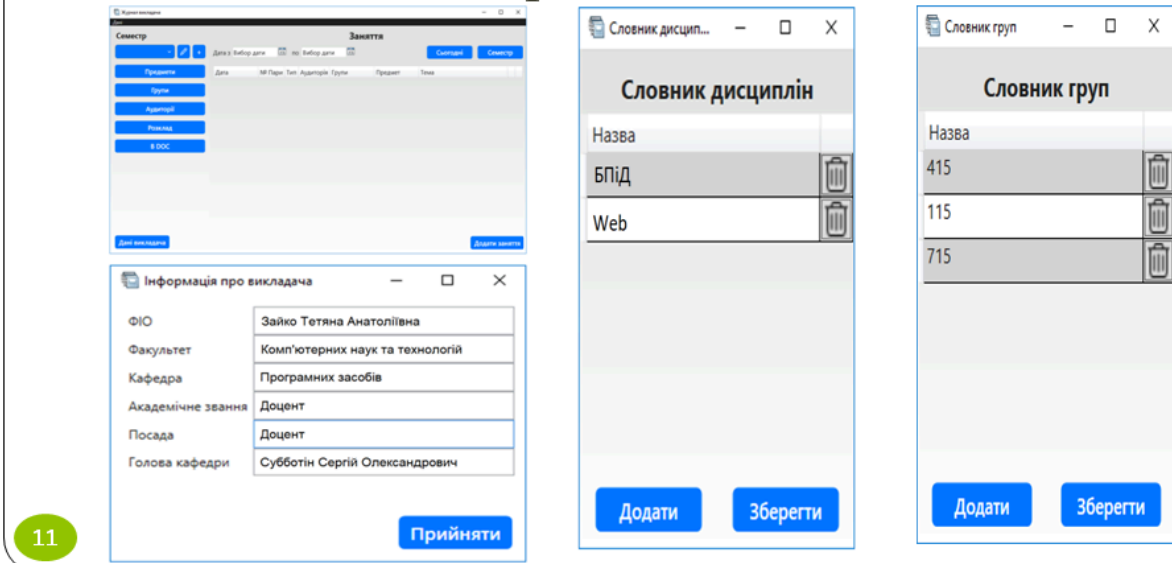


Рисунок В.11 – Слайд 11

Інтерфейс програми. Формування розкладу та звітності викладача

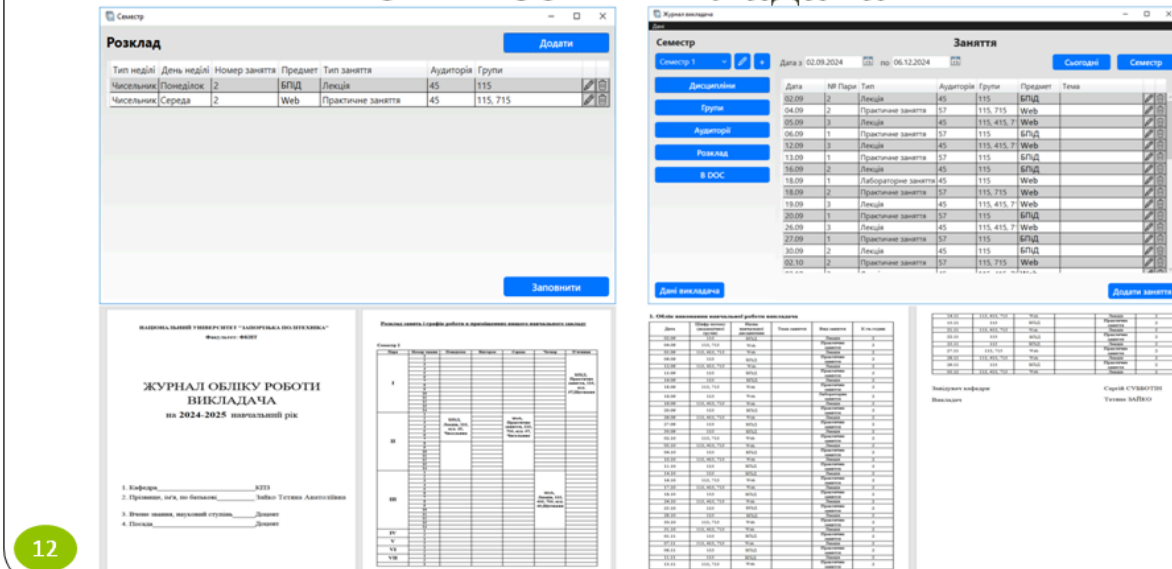


Рисунок В.12 – Слайд 12

Висновки

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для обліку викладацької роботи.

Сформульовано функціональні вимоги до програмного забезпечення для обліку викладацької роботи. Для реалізації програмного забезпечення для обліку викладацької роботи обрано мову програмування C# та середовище розробки Visual Studio.

Розроблено програмне забезпечення для обліку викладацької роботи. Запропоновано структуру програмного забезпечення для обліку викладацької роботи. Розроблено базу даних програмного забезпечення для обліку викладацької роботи, яка призначена для зберігання, впорядкування та зв'язування інформації, що стосується семестрів, навчальних занять, розкладу, предметів, аудиторій, груп та інших супровідних об'єктів.

В результаті виконання завдання було отримано систему, що відповідає всім вимогам визначеним на початку розробки та закріпленим в технічному завданні. Розроблена система є зручною для використання та виконує всі необхідні функції.

Усі завдання дипломної кваліфікаційної роботи бакалавра повністю виконано.

13

Рисунок В.13 – Слайд 13