Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Інформаційна система "Кабінет відповідального працівника приймальної комісії"
Назва теми

КвРІСТ. 200184.01.54.00 ПЗ
Шифр

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *126 «Інформаційні системи та технології»*
Шифр, назва

Освітня програма *«Інформаційні системи та технології»*
Назва

Виконав: студент IV курсу, група ІСТ-20-1 _____ Д. В. Погорєлов
Підпис Ініціали, прізвище

Керівник _____ Т. М. Кисіль
Підпис, дата Ініціали, прізвище

Нормоконтролер _____ С.М. Лисенко
Підпис, дата Ініціали, прізвище

До захисту допускаю:
Зав. кафедри комп'ютерної інженерії та інформаційних систем _____ Т.О.Говорущенко
Підпис Ініціали, прізвище

«20» червня 2024 р.

Хмельницький 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Комп'ютерної інженерії та інформаційних систем

Освітній рівень БАКАЛАВР

Галузь знань 12 Інформаційні технології

Спеціальність 126 Інформаційні системи та технології

Освітня програма «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ

Зав. кафедри Т.О.Говорущенко

" 16 " 01 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Погорєлову Дмитру Вікторовичу

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Інформаційна система "Кабінет відповідального працівника_ приймальної комісії"

Керівник проекту (роботи) Кисіль Т.М., к.ф.-м.н., доц.

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 01.03.2024 р. № 5

2. Строк подання студентом проекту (роботи) на кафедру 01.06.2024 р.

3. Вихідні дані до проекту (роботи) Завдання на кваліфікаційну роботу

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Аналіз відомих засобів та інформаційних систем автоматизованого заповнення документів

Проектування інформаційної системи "кабінет відповідального працівника приймальної комісії"

Реалізація інформаційної системи "кабінет відповідального працівника приймальної комісії"

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Моделювання поведінки системи

Моделювання динамічних аспектів системи

Модель класів

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Лисенко С.М., професор кафедри КІІС		
Антиплагіат	Нічепорук А.О., доцент кафедри КІІС		

7. Дата видачі завдання «10» 01 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2024	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2024	виконано
3	Робота над розділом 1 – Аналіз відомих методів та інформаційних систем автоматичного заповнення документів	01.03.2024	виконано
4	Робота над розділом 2 – Проєктування інформаційної системи "Кабінет відповідального працівника приймальної комісії"	01.04.2024	виконано
5	Робота над розділом 3 – Реалізація інформаційної системи "Кабінет відповідального працівника приймальної комісії"	29.04.2024	виконано
6	Оформлення пояснювальної записки згідно вимог	25.05.2024	виконано
7	Попередній захист ВКР	26.05.2024	виконано
8	Захист ВКР на засіданні ЕК	Червень 2024 року	

Студент _____ Д. В Погорєлов.
Підпис Ініціали, прізвище

Керівник роботи _____ Т. М Кисіль
Підпис Ініціали, прізвище

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРІСТ. 200184.01.54.00 ПЗ	Пояснювальна записка	55		
			Графічні матеріали			
2		КвРІСТ. 200184.01.54.00 Е8	Моделювання поведінки	1		
			системи			
3		КвРІСТ. 200184.01.54.00 Е8	Моделювання динамічних	1		
			Аспектів системи			
4		КвРІСТ. 200184.01.54.00 Е8	Модель класів	1		

Зм	Арк	№ докум	Підпис	Дата	КвРІСТ. 200184.01.54.00 ПЗ		
Розробив		Погорелов			Відомість проекту	Літера	Аркуш	Аркушів
Перевір.		Кисіль				У	1	1
Н. контр.		Лисенко				ХНУ, ІСТ-20-1		
Затв.		Говорущенко		20.06				

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Інформаційна система "Кабінет відповідального працівника приймальної комісії"».

Автор роботи: Погорєлов Дмитро Вікторович.

Керівник роботи: Кисіль Тетяна Миколаївна.

Пояснювальна записка: 55 с., 21 рис., 3 табл., 4 дод., 39 джерел.

Графічна частина: 3 креслення.

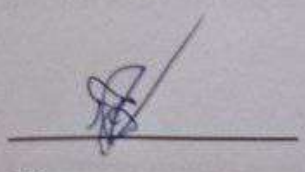
ПРИЙМАЛЬНА КОМІСІЯ, ШАБЛОНИ ДОКУМЕНТІВ, АВТОМАТИЧНЕ ЗАПОВНЕННЯ НАБОРУ ДОКУМЕНТІВ

Метою роботи є розробка програмного продукту, який зможе спростити роботу приймальної комісії, автоматизувати процес заповнення документації, як то договір про навчання та договір про надання платної освітньої послугу, тощо.

Об'єктом дослідження є процеси автоматизації заповнення документів, які абітурієнт заповнює при вступі.

Предметом дослідження є інформаційна система, що надає можливість автоматичного заповнення декількох документів одночасно після одноразового введення персональних даних абітурієнта.

В даній роботі було змодельовано та реалізовано інформаційні систему, що автоматизує внесення персональних даних абітурієнта під час особистої подачі документів на вступ.



Підпис студента

Дата

ЗМІСТ

ВСТУП 4

1 АНАЛІЗ ВІДОМИХ ЗАСОБІВ ТА ІНФОРМАЦІЙНИХ СИСТЕМ АВТОМАТИЧНОГО ЗАПОВНЕННЯ ДОКУМЕНТІВ 5

1.1 Інформаційні системи та їх структура 5

1.2 Технології автоматичного заповнення документів 8

1.3 Огляд відомих систем, що реалізують технології автоматичного заповнення документів 10

1.4 Висновки 22

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ "КАБІНЕТ ВІДПОВІДАЛЬНОГО ПРАЦІВНИКА ПРИЙМАЛЬНОЇ КОМІСІЇ" 23

2.1 Постанока задачі та формулювання цілей проекту 23

2.2 Функціональні та нефункціональні вимоги до системи 24

2.3 Формалізація системи 27

2.3.1 Діаграма використання 27

2.3.2 Діаграма діяльності 28

2.3.3 Діаграма послідовності 30

2.4 Висновки 32

3 РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ "КАБІНЕТ ВІДПОВІДАЛЬНОГО ПРАЦІВНИКА ПРИЙМАЛЬНОЇ КОМІСІЇ" 34

3.1 Проектування інтерфейсу користувача 34

3.2 Вибір середовища розробки 37

3.3 Реалізація інтерфейсу користувача в програмному середовищі 41

3.4 Програмна реалізація основних модулів системи 42

3.4.1 Модуль створення шаблону документу 43

3.4.2 Модуль створення набору документів 44

3.4.3 Модуль введення та зберігання даних 45

3.4.4 Модуль взаємодії з принтером 46
3.4.5 Модуль взаємодії з банком даних 46
3.5 Тестування інформаційної системи 47
3.5.1 Функціональне тестування 49
3.5.2 Тестування графічного інтерфейсу системи 50
3.6 Висновки розділу 54
ВИСНОВКИ 57
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ 59
ДОДАТОК А 61
ДОДАТОК Б 62
ДОДАТОК В 63
ДОДАТОК Г 64

ВСТУП

Щороку наша країна робить крок вперед у напрямку повної діджиталізації. Однак приймальна комісія досі потребує значних людських ресурсів, який реалізує процес прийому, систематизації, сортування та перевірки документів абітурієнтів. Абітурієнтам доводиться вручну заповнювати досить велику кількість документів, а працівникам комісії доводиться вручну співставляти та порівнювати дані з різних джерел, наданих абітурієнтами. Окрім того, щоденно необхідні звіти про кількість абітурієнтів, кількість поданих документів та інші дані вступників.

Через ці труднощі виникла ідея створити систему, яка може допомогти вирішити більшість проблем, з якими стикається технічні працівники приймальної комісії, і в майбутньому бути впровадженою в нашому університеті.

Метою роботи є розробка програмного продукту, який зможе спростити роботу приймальної комісії, автоматизувати процес заповнення документації, як то договір про навчання та договір про надання платної освітньої послугу, тощо.

Об’єктом дослідження є процеси автоматизації заповнення документів, які абітурієнт заповнює при вступі

Предметом дослідження є інформаційна система, що надає можливість автоматичного заповнення декількох документів одночасно після одноразового введення персональних даних абітурієнта

Крім того, для зручності користувача всі дані можна експортувати в електронні таблиці MS Excel, де за допомогою налаштування все можливих фільтрів можна формувати звіти, що стосуються динаміки подачі документів, тощо.

1 АНАЛІЗ ВІДОМИХ ЗАСОБІВ ТА ІНФОРМАЦІЙНИХ СИСТЕМ АВТОМАТИЧНОГО ЗАПОВНЕННЯ ДОКУМЕНТІВ

1.1 Інформаційні системи та їх структура

Інформаційні системи є невід'ємною частиною сучасного світу, забезпечуючи основу для зберігання, обробки та передачі інформації. Вони поєднують у собі різні технічні засоби, програмне забезпечення, бази даних, процедури та людей, створюючи цілісну структуру, яка дозволяє ефективно керувати даними та підтримувати прийняття рішень.

Уявіть собі великий організм, де кожен орган виконує свою унікальну функцію. Апаратне забезпечення тут відіграє роль фізичних частин цього організму – комп'ютери, сервери, мережеве обладнання і периферійні пристрої. Без них інформаційні системи не змогли б функціонувати. Комп'ютери та сервери обробляють і зберігають дані, мережеве обладнання забезпечує зв'язок між різними компонентами, а периферійні пристрої дозволяють вводити та виводити інформацію [1].

Програмне забезпечення можна порівняти з нервовою системою, яка координує всі дії та процеси. Воно включає системне програмне забезпечення, таке як операційні системи та системи управління базами даних, які забезпечують фундаментальні функції. Прикладне програмне забезпечення, на кшталт програм для обробки текстів, електронних таблиць або спеціалізованих програмних комплексів, дозволяє користувачам виконувати конкретні завдання, полегшуючи роботу з даними.

Дані та бази даних – це серце інформаційних систем. Вони зберігають усю важливу інформацію, яку системи збирають, обробляють і передають. Системи управління базами даних (СУБД) відіграють ключову роль у забезпеченні ефективного зберігання та доступу до даних, дозволяючи організаціям зберігати великі обсяги інформації в структурованому вигляді.

Процедури та політики – це правила гри, які визначають, як дані мають оброблятися, зберігатися та передаватися. Вони забезпечують дотримання стандартів безпеки, конфіденційності та точності, а також надають інструкції користувачам щодо взаємодії з інформаційною системою.

Нарешті, людські ресурси – це ті, хто приводить усе це в дію. Користувачі, адміністратори, аналітики, програмісти та технічна підтримка – усі вони є невід'ємною частиною інформаційних систем. Без людей, які розробляють, підтримують і використовують ці системи, жодна інформаційна система не змогла б функціонувати [2].

Інформаційні системи поділяються на різні типи, залежно від їхнього призначення. Наприклад, транзакційні системи обробляють рутинні операції, такі як замовлення чи платежі, тоді як управлінські інформаційні системи забезпечують підтримку управлінських функцій, надаючи звітність та аналітичні дані. Системи підтримки прийняття рішень допомагають аналізувати великі обсяги даних і моделювати різні сценарії, тоді як системи управління знаннями зберігають і поширюють знання всередині організації. Експертні системи використовують знання фахівців для вирішення складних завдань у вузьких сферах [3].

Розробка інформаційних систем базується на принципах системного підходу, інтеграції, модульності, масштабованості та безпеки. Системний підхід дозволяє розглядати інформаційну систему як комплекс взаємопов'язаних компонентів. Інтеграція забезпечує об'єднання різних частин системи для безперебійного обміну інформацією. Модульність дозволяє розділяти систему на окремі частини, які можна розробляти та оновлювати незалежно одна від одної. Масштабованість забезпечує можливість розширення системи без значних змін у її архітектурі. Безпека і надійність гарантують захист даних і безперебійну роботу системи.

Основні виклики, з якими стикаються інформаційні системи, включають кібербезпеку, управління великими даними та інтеграцію нових технологій, таких як штучний інтелект.

Проте перспективи розвитку цих систем величезні.

Вони включають розширення можливостей автоматизації, підвищення точності прогнозування та прийняття рішень, а також покращення взаємодії між людьми та системами за допомогою інтуїтивно зрозумілих інтерфейсів [4].

Таким чином, інформаційні системи є складними, багатокомпонентними структурами, що забезпечують ефективне управління інформацією та підтримку різноманітних процесів у сучасному світі (рис. 1.2).

Вони поєднують у собі технології, дані, процеси та людей, створюючи цілісні рішення для різних завдань.

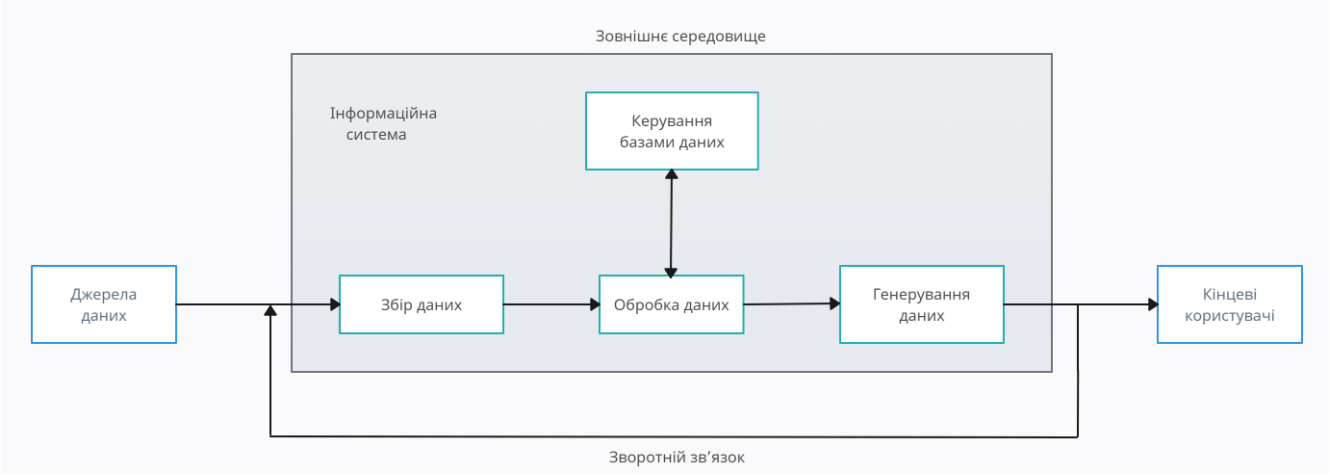


Рисунок 1.1 – Структура інформаційної системи

1.2 Технології автоматичного заповнення документів

Технології автоматичного заповнення документів докорінно змінюють спосіб, у який ми працюємо з паперовими та цифровими формами, значно спрощуючи рутинні процеси. Уявіть, що ви заповнюєте довгу анкету або складний звіт: скільки часу й зусиль це вимагає? А тепер уявіть, що більшість цієї роботи виконується автоматично, без необхідності вводити кожне слово вручну. Це саме те, що роблять сучасні технології автоматичного заповнення документів [5].

Серед основних методів, які використовуються в цій сфері, можна виділити оптичне розпізнавання символів (OCR).

Ця технологія дозволяє сканувати паперові документи й перетворювати їх у цифровий текст, який потім можна редагувати або зберігати в електронному вигляді.

Сучасні OCR-системи настільки точні, що можуть розпізнавати навіть рукописний текст, що значно розширює їхнє застосування.

Ще однією важливою технологією є використання шаблонів документів. Це стандартні форми, в які автоматично вставляються необхідні дані.

Уявіть собі, що вам потрібно заповнити десятки однакових форм, але всі необхідні дані автоматично підставляються у відповідні поля.

Це не лише економить час, але й зменшує ймовірність помилок, які можуть виникнути при ручному введенні даних [6].

Інтеграція з базами даних є ще одним важливим аспектом автоматичного заповнення документів.

У великих організаціях, де зберігаються великі обсяги даних, ця технологія дозволяє автоматично заповнювати форми на основі вже наявної інформації. Наприклад, коли потрібно створити звіт чи анкету, система автоматично витягує необхідні дані з бази й заповнює поля документів.

Розпізнавання природної мови (NLP) також грає важливу роль у цій сфері. Ця технологія дозволяє аналізувати та інтерпретувати текст, заповнюючи документи на основі аналізу наявної інформації.

Наприклад, система може витягувати ключові дані з електронних листів або інших текстових документів і автоматично вставляти їх у відповідні поля [7].

Роботизовані процеси автоматизації (RPA) також широко використовуються для автоматичного заповнення документів. Ці роботи здатні виконувати рутинні завдання на основі попередньо визначених правил і сценаріїв. Вони можуть взаємодіяти з різними системами, автоматично виконуючи завдання, які раніше вимагали ручного втручання.

Штучний інтелект (AI) і машинне навчання (ML) стають все більш важливими в цій сфері.

Ці технології можуть вивчати патерни й тенденції в даних, передбачати необхідну інформацію та автоматично заповнювати поля на основі аналізу великої кількості змінних.

Це особливо корисно для складних документів, де потрібно враховувати багато факторів [8].

Приклади використання цих технологій можна побачити в різних сферах. У фінансовому секторі автоматичне заповнення документів значно прискорює процес обробки кредитних заявок і відкриття рахунків.

В охороні здоров'я автоматичне заповнення медичних форм і звітів зменшує адміністративне навантаження на медичних працівників, дозволяючи їм зосередитися на лікуванні пацієнтів.

У юридичній сфері ці технології використовуються для підготовки контрактів і угод, підвищуючи точність юридичних документів.

В освітніх установах автоматичне заповнення документів допомагає адміністраторам швидше обробляти заявки на вступ і заповнювати навчальні плани [9].

Звичайно, такі системи мають свої переваги та недоліки. Вони підвищують продуктивність, знижують кількість помилок, економлять час і покращують якість даних.

Проте впровадження таких систем може вимагати значних початкових інвестицій і ретельного налаштування.

Інтеграція нових технологій з існуючими системами іноді викликає складнощі, а також необхідно забезпечити належний рівень безпеки та конфіденційності даних [10].

Технології автоматичного заповнення документів значно підвищують ефективність роботи в різних галузях, дозволяючи зосередитися на більш важливих завданнях і мінімізуючи рутинні процеси.

Вони є незамінними інструментами в сучасному світі, де точність і швидкість обробки інформації мають вирішальне значення [11].

1.3 Огляд відомих систем, що реалізують технології автоматичного заповнення документів.

Проаналізувавши найвідоміші технологій автоматичного заповнення документів, такі як.

Adobe Sign, розроблений компанією Adobe Systems Incorporated у США, є потужним інструментом для автоматичного заповнення та підписання документів. Він інтегрується з іншими продуктами Adobe, а також з Microsoft Office, що робить його надзвичайно зручним для користувачів, які вже працюють з цими програмами.

Adobe Sign підтримує електронні підписи і дозволяє відстежувати статус документів, що значно спрощує процес документального обігу.

Однією з головних переваг Adobe Sign є його зручний інтерфейс, який полегшує роботу користувачам, а також легка інтеграція з популярними програмами.

Крім того, система забезпечує високий рівень безпеки і відповідає юридичним вимогам, що важливо для компаній, які обробляють чутливу інформацію.

Adobe Sign також підтримує роботу з мобільними пристроями, що дозволяє користувачам підписувати та відправляти документи з будь-якого місця.

Однак, варто зазначити, що висока вартість підписки може бути значним недоліком для підприємств малого бізнесу.

Іноді користувачі також стикаються з проблемами сумісності з іншими системами, що може ускладнити процес інтеграції Adobe Sign у вже існуючі робочі процеси.

Приклад заповнення документа в системі Adobe Sign можна побачити на рисунку 1.2, де наочно демонструється процес заповнення полів і додавання електронного підпису.

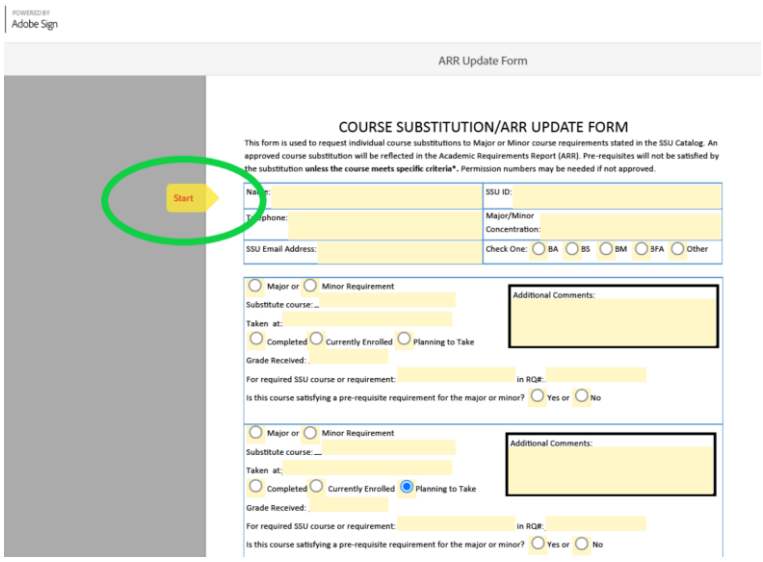


Рисунок 1.2 – Заповнення документа в Adobe Sign

DocuSign, розроблений американською компанією DocuSign Inc., є широко відомим інструментом для електронного підпису та автоматичного заповнення документів.

Ця система інтегрується з CRM-системами, такими як Salesforce, що дозволяє безперебійно вбудувати її в існуючі бізнес-процеси.

DocuSign підтримує різні формати документів і надає можливість відстеження змін, що робить роботу з документами більш прозорою і контрольованою.

Основною перевагою DocuSign є її широка інтеграція з багатьма бізнес-додатками, що дозволяє користувачам ефективно використовувати систему в рамках вже існуючих інфраструктур.

DocuSign відзначається високою надійністю і безпекою, що забезпечує захист даних на всіх етапах обробки документів.

Система також відзначається простотою використання та масштабованістю, що дозволяє компаніям різних розмірів адаптувати її під свої потреби.

Проте, деякі користувачі вважають інтерфейс DocuSign занадто складним, що може вимагати додаткового навчання для нових користувачів.

Крім того, вартість послуг DocuSign може бути високою для малих підприємств, що може стати бар'єром для його впровадження в невеликих компаніях.

Детальніше ознайомитися з функціоналом DocuSign можна за посиланням: DocuSign. На рисунку 1.3 продемонстровано ідписання документа в DocuSign.

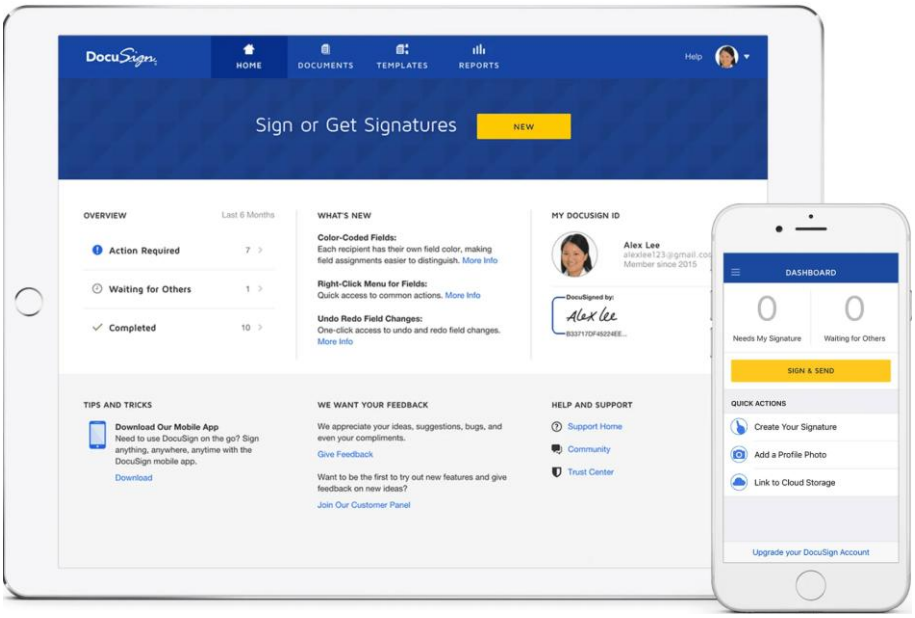


Рисунок 1.3 – Підписання документа в DocuSign

Microsoft Power Automate (раніше відомий як Microsoft Flow), розроблений корпорацією Microsoft, є потужним інструментом для автоматизації робочих процесів, зокрема для автоматичного заповнення документів.

Ця система інтегрується з широким спектром додатків, включаючи SharePoint, Dynamics 365, і Office 365, що робить її надзвичайно корисною для користувачів, які вже працюють з продуктами Microsoft.

Використання шаблонів дозволяє швидко створювати автоматизовані процеси, значно скорочуючи час, необхідний для налаштування.

Основні переваги Microsoft Power Automate включають глибоку інтеграцію з продуктами Microsoft, що забезпечує безперебійну роботу з іншими програмами та сервісами компанії.

Широкі можливості для автоматизації різних процесів дозволяють користувачам значно підвищити ефективність роботи, зменшивши кількість рутинних завдань. Гнучкість і масштабованість цієї системи роблять її придатною для використання як у малих офісах, так і у великих корпораціях.

Однак, деякі користувачі можуть зіткнутися зі складністю налаштування системи, особливо новачки, які не мають досвіду роботи з такими інструментами.

Крім того, деякі функції доступні лише в преміум-версії, що може обмежити можливості для користувачів без додаткової підписки.

Докладніше про можливості та функції Microsoft Power Automate можна дізнатися за посиланням: Microsoft Power Automate. .

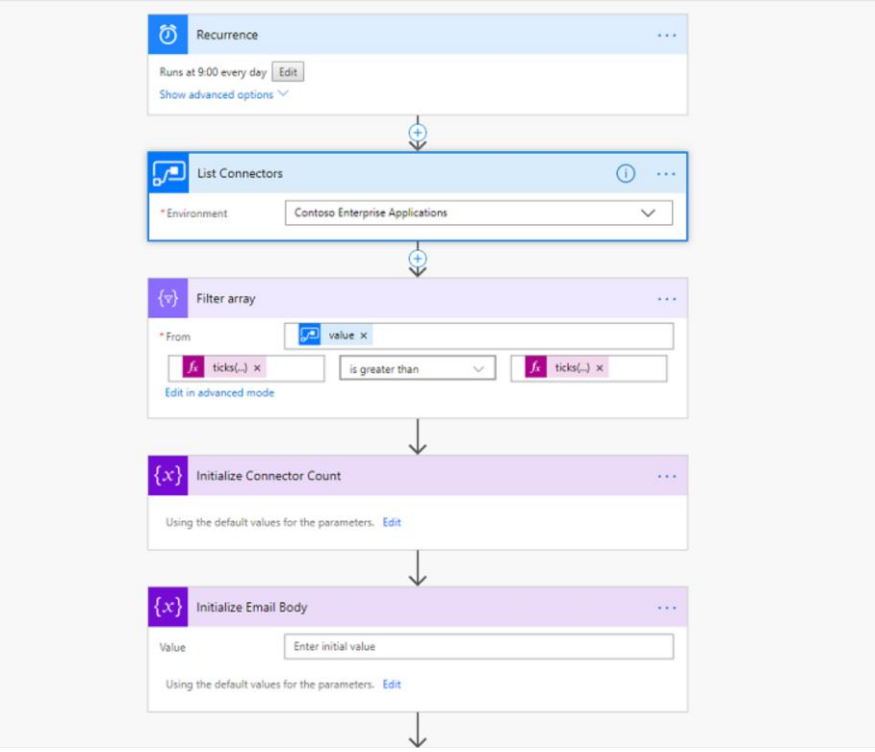


Рисунок 1.4- Автоматизація завдань за допомогою Microsoft Power Automate

Formstack, розроблений компанією Formstack LLC у США, є популярним інструментом для створення та автоматичного заповнення онлайн-форм і документів.

Ця система інтегрується з різними CRM-системами, платіжними шлюзами і хмарними сервісами, що забезпечує широкий спектр можливостей для бізнес-користувачів.

Formstack також підтримує електронні підписи і автоматизацію робочих процесів, що дозволяє значно підвищити ефективність роботи.

Основні переваги Formstack включають легкість створення та використання форм, що робить цей інструмент доступним навіть для користувачів без спеціальних технічних знань.

Багатий набір інтеграцій дозволяє користувачам підключати систему до різноманітних сервісів, що забезпечує гнучкість у роботі.

Потужні інструменти для автоматизації процесів допомагають зменшити кількість ручної праці і скоротити час на виконання рутинних завдань.

Втім, деякі функції Formstack можуть бути складними у налаштуванні, що може вимагати додаткового часу і ресурсів для освоєння.

Крім того, висока вартість підписки може стати перешкодою для невеликих компаній, які мають обмежені бюджети.

Більше інформації про можливості та функції Formstack можна знайти за посиланням: Formstack.

В Україні також є ряд аналогічних програм, проте деякі з них не дуже популярні.

Liga, розроблена українською компанією Ліга:Закон, є спеціалізованим інструментом для автоматизації юридичних документів. Ця система дозволяє автоматично заповнювати шаблони для договорів, заяв, актів та інших юридичних документів.

Вона також інтегрується з іншими юридичними базами даних, що забезпечує доступ до актуальної юридичної інформації.

Однією з ключових переваг Liga є висока точність заповнення документів, що зменшує ризик помилок і підвищує ефективність роботи юридичних фахівців. Крім того, інтеграція з різними юридичними сервісами дозволяє користувачам отримувати необхідну інформацію та підтримку в одному місці, що спрощує їх роботу.

Проте, використання Liga може бути складним для новачків у юридичній сфері, оскільки вимагає певних знань і досвіду.

Висока вартість підписки також може бути перешкодою для малих підприємств, які мають обмежені бюджети.

Докладніше про Liga та її можливості можна дізнатися за посиланням: Liga.

Рисунок 1.5 – Можливості Liga

Docflow, розроблена українською компанією IT-Enterprise, є потужною системою для автоматизації документообігу. Цей інструмент дозволяє автоматично заповнювати шаблони документів, що значно спрощує процес створення і обробки документації. Однією з важливих функцій Docflow є інтеграція з ERP-системами, що забезпечує зручне управління бізнес-процесами в єдиному середовищі.

Однією з ключових переваг Docflow є його зручна інтеграція з іншими системами IT-Enterprise, що дозволяє користувачам легко налаштовувати і використовувати систему в комплексі з іншими інструментами. Висока ефективність та надійність роблять Docflow відмінним рішенням для великих компаній, які прагнуть оптимізувати свій документообіг. Система також підтримує електронний документообіг, що сприяє зменшенню використання паперових документів і підвищенню екологічності процесів.

Проте, висока вартість впровадження Docflow може бути значною перешкодою для малих підприємств, які мають обмежені фінансові ресурси. Крім

того, налаштування системи може бути досить складним для малих бізнесів, які не мають спеціалістів з IT.

Докладніше про Docflow та її можливості можна дізнатися за посиланням: Docflow.

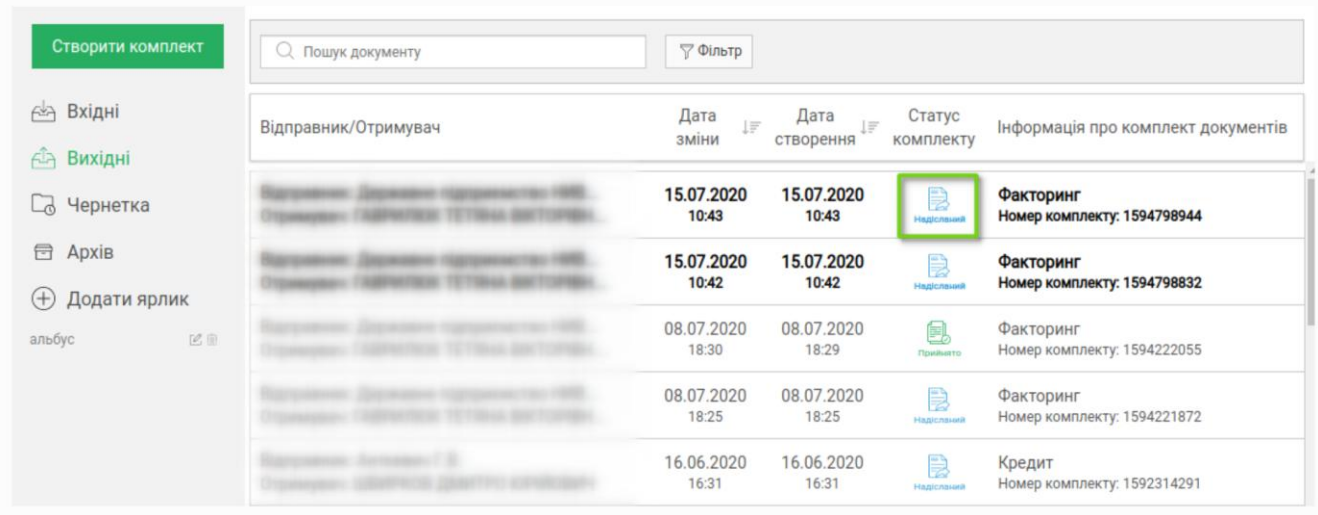


Рисунок 1.6 – Робота з комплектом документів в Docflow

M.E.Doc, розроблена українською компанією Інтелект-Сервіс, є провідною системою для автоматизації бухгалтерської та податкової звітності. Цей інструмент значно спрощує процес заповнення та подання звітів до податкових органів, забезпечуючи високу точність та відповідність актуальним вимогам законодавства. M.E.Doc також підтримує інтеграцію з банківськими системами, що дозволяє автоматизувати фінансові операції та полегшити управління бухгалтерією.

Однією з головних переваг M.E.Doc є наявність актуальних шаблонів звітності, які регулярно оновлюються відповідно до змін у податковому законодавстві. Це гарантує користувачам своєчасне подання звітів і зменшує ризик помилок. Інтеграція з податковими системами дозволяє автоматично передавати дані безпосередньо до відповідних органів, що економить час та зусилля бухгалтерів. Висока надійність і зручність використання роблять M.E.Doc незамінним інструментом для бухгалтерів і фінансових спеціалістів.

Проте система має свої недоліки. Вона потребує регулярного оновлення для забезпечення відповідності актуальним законодавчим вимогам, що може бути незручним для деяких користувачів. Крім того, висока вартість може стати суттєвою перешкодою для малого бізнесу, який має обмежені фінансові ресурси.

Детальніше про M.E.Doc та її можливості можна дізнатися за посиланням: M.E.Doc.

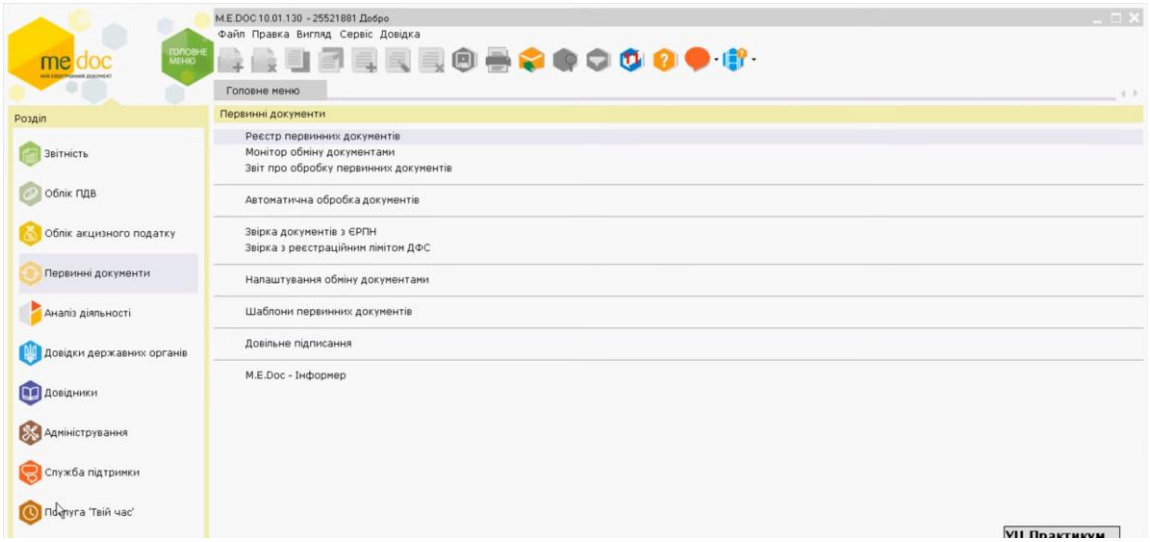


Рисунок 1.7 – Початок роботи в M.E.Doc

Наведені приклади систем є затребувані, проте вони складно адаптуються до особливостей роботи технічного працівника приймальної комісії. В сфері освіти можна виділити відомий нам «Електронний університет», «Електронний щоденник» та інші.

Українські електронні журнали і щоденники, розроблені веб-розробниками та IT-компаніями у сфері освіти, є сучасними інструментами для автоматизації ведення шкільної документації. Ці системи забезпечують автоматичне заповнення оцінок та відвідуваності, що значно полегшує роботу вчителів. Крім того, вони дозволяють генерувати звіти про успішність учнів, що робить контроль за навчальним процесом більш ефективним.

Інтеграція з електронними системами подачі завдань та оцінювання забезпечує зручність для вчителів та учнів, створюючи єдину платформу для

управління навчальним процесом. Батьки можуть отримувати доступ до інформації про успішність і відвідуваність своїх дітей, що сприяє підвищенню прозорості та залученості у навчальний процес.

Однією з головних переваг цих систем є полегшення роботи вчителів з ведення документації. Замість ручного заповнення журналів та підготовки звітів, вчителі можуть використовувати автоматизовані інструменти, що економить час і знижує ймовірність помилок. Батьки також отримують можливість оперативно отримувати інформацію про академічну успішність і відвідуваність дітей, що допомагає краще контролювати навчальний процес.

Проте електронні журнали і щоденники потребують технічного обслуговування та підтримки, що може створювати певні труднощі для шкіл, особливо у разі обмежених ресурсів. Крім того, у випадку технічних збоїв може виникати проблема з доступом до системи, що може вплинути на своєчасність отримання інформації.

Для детальнішого ознайомлення можна відвідати ресурс "Мій клас": Мій клас.

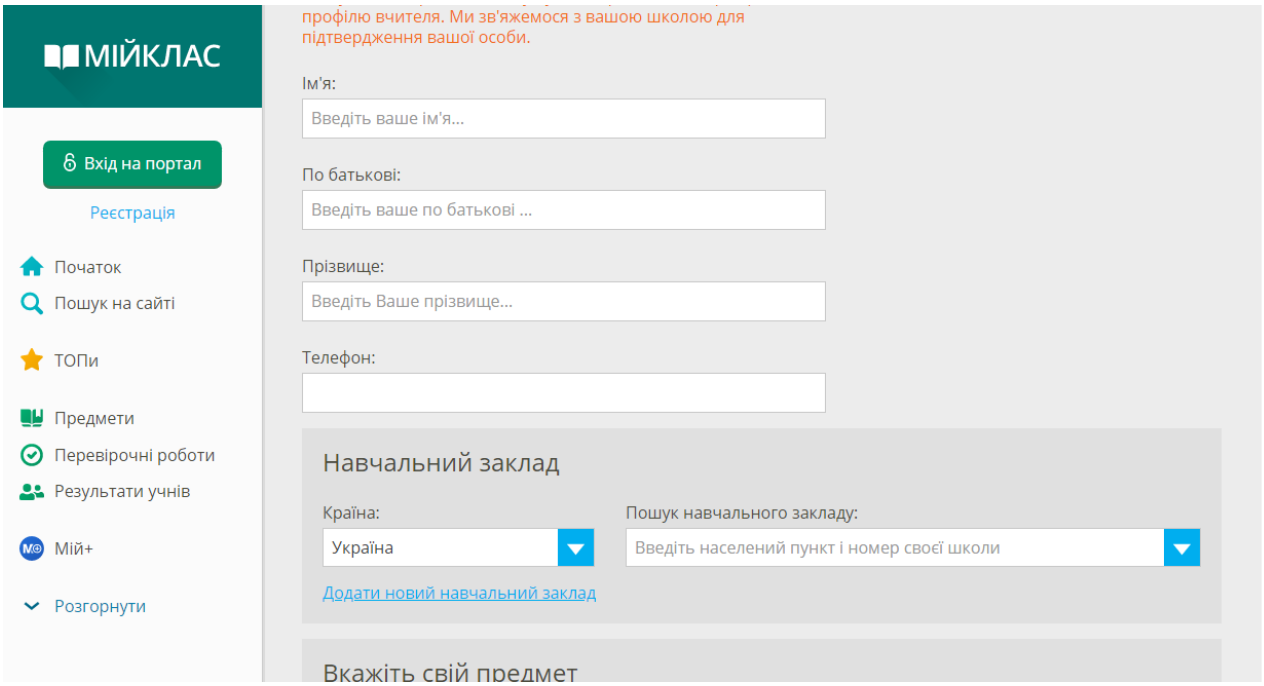


Рисунок 1.8 – Реєстраційна форма для вчителя на платформі для дистанційного навчання «МійКлас»

Система "Електронний університет", розроблена українськими веб-розробниками та IT-компаніями в сфері освіти, є сучасним інструментом для автоматизації різних адміністративних процесів в університетах. Ця система забезпечує автоматичне заповнення заяв на вступ, стипендії та інші адміністративні документи, що значно полегшує роботу адміністративного персоналу. Крім того, вона дозволяє генерувати та управляти студентськими квитками, дипломами та іншими документами, інтегруючись з базами даних студентів та викладачів.

Однією з головних переваг системи "Електронний університет" є спрощення адміністративних процесів в університеті. Автоматизація заповнення та обробки документів підвищує точність і швидкість роботи, знижуючи ризик помилок. Студенти та викладачі отримують легкий доступ до необхідної інформації, що покращує загальний рівень обслуговування в університеті.

Проте, як і будь-яка технологія, система має свої недоліки. Впровадження та підтримка такої системи можуть вимагати значних фінансових витрат, що може бути проблемою для деяких навчальних закладів. Крім того, для ефективного використання системи необхідно навчати персонал, що потребує додаткових ресурсів та часу.

Для детальнішого ознайомлення можна відвідати ресурс "Електронний університет": Електронний університет.

На жаль, не знайдено інформації про університети в Україні, які використовують системи автоматичного заповнення документів для абітурієнтів під час особистої подачі документів та укладання договорів про навчання . Ця інформація може бути специфічна для кожного університету або вищого навчального закладу і залежить від їхніх внутрішніх систем та технологій. Але точно відомо, що Хмельницький національний університет поки що такої системи не має.

1.4 Висновки

Інформаційні системи, що автоматично генерують документи, є важливим інструментом для підвищення ефективності в різних сферах діяльності, включаючи освітні установи. Вони дозволяють значно скоротити час, що витрачається на рутинні завдання, підвищують точність документів, зменшують кількість помилок та оптимізують робочі процеси. Для приймальних комісій це особливо актуально, оскільки забезпечення своєчасного та точного оформлення документів є критично важливим для процесу вступу абітурієнтів.

Автоматизація процесу створення документів у приймальних комісіях сприяє зменшенню навантаження на працівників, дозволяє зосередитись на більш важливих завданнях та покращує якість обслуговування абітурієнтів. Крім того, це сприяє зменшенню стресу у відповідальних працівників, пов’язаного з обробкою великого обсягу документації, особливо у пікові періоди вступної кампанії.

На жаль, не знайдено інформації про університети в Україні, які використовують системи автоматичного заповнення документів для абітурієнтів під час особистої подачі документів. Ця інформація може бути специфічна для кожного університету або вищого навчального закладу і залежить від їхніх внутрішніх систем та технологій. Але точно відомо, що Хмельницький національний університет поки що такої системи не має.

Тому постає задача в розробці інформаційної системи "Кабінет відповідального працівника приймальної комісії" для автоматичного створення та заповнення документів на основі введених даних [12].

Мета: Автоматизувати процес заповнення документів, що використовуються у приймальній комісії, зокрема договір про навчання, договір про надання платої освітньої послуги, заяву на гуртожиток, опис документів у справі та інші супровідні документи.

Для досягнення мети можна виділити такі завдання:

1.Створення зручного інтерфейсу.

2.Зберігання даних абітурієнтів.

3.Розробка шаблонів для різних видів документів, які використовуються у процесі прийому абітурієнтів. Це можуть бути договори про навчання, контракти, заяви на проживання у гуртожитку та інші супровідні документи..

4. Розробка алгоритмів, що забезпечать автоматичне заповнення шаблонів документів даними.

5. Забезпечення можливості швидкого та зручного друку готових документів безпосередньо з інтерфейсу системи

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ “КАБІНЕТ ВІДПОВІДАЛЬНОГО ПРАЦІВНИКА ПРИЙМАЛЬНОЇ КОМІСІЇ”

2.1 Постанока задачі та формулювання цілей проекту

Проектування системи перед її реалізацією є критично важливим етапом у процесі розробки програмного забезпечення з кількох ключових причин. Цей етап допомагає забезпечити чітке розуміння вимог, покращити якість кінцевого продукту та зменшити ризики, пов'язані з розробкою. Чітко сформульовані цілі задають напрямок розвитку проєкту, окреслюють очікувані результати та створюють базу для оцінки його ефективності й успішності. Вони допомагають зосередитися на завданнях і забезпечують координацію дій на кожному етапі виконання [15].

Під час вступної компанії абітурієнт повинен особисто укласти договір з університетом про навчання, укласти договір про надання платної освітньої послуги, написати заяву на гуртожиток тощо. Всі ці документи потребують внесення особистих даних абітурієнта, які часто дублюються в різних документах. Це призводить до рутинної роботи, яка окрім того зазвичай може призвести до помилок та неточностей. Технічний працівник приймальної комісії повинен перевірити і виявити ці неточності, і повторно переглядає заповнені абітурієнтом документи. Тому ключовою метою проєкту є створення інформаційної системи, яка дозволить технічному персоналу і абітурієнтам легко, швидко та без зайвих помилок виконувати рутинну роботу під час паперової подачі документів на вступ.

Проаналізувавши можливі технології автоматичного заповнення документів, було вирішено застосувати технологію створення шаблонів [16].

2.2 Функціональні та нефункціональні вимоги до системи

Функціональні вимоги допомагають чітко визначити, що система повинна робити, і забезпечують основу для розробки, тестування та впровадження програмного забезпечення. Вони є критичним елементом процесу розробки, який забезпечує, що кінцевий продукт відповідатиме потребам користувачів і виконає всі необхідні завдання [17].

Як правило функціональні вимоги висуваються до користувацького інтерфейсу **(**описують, як користувачі взаємодіють із системою, які елементи інтерфейсу доступні та як вони повинні виглядати**)**, до управління даними **(**описують, як система повинна зберігати, обробляти та управляти даними**), до** функціоналу самої системи **(**описують основні функціональні можливості системи, які забезпечують її основну діяльність), можливості інтегрування **(**описують, як система повинна взаємодіяти з іншими системами або зовнішніми сервісами), до безпеки **(**описують заходи, які забезпечують захист даних та доступ до системи), до оброблення помилок **(**описують, як система повинна поводитися в разі виникнення помилок).

Загалом, як видно з рисунку 1.2 формулювання функціональних вимог до системи є лише одним з етапів опису специфікацій до системи. Необхідно ще враховувати користувацькі та системні вимоги [18].

Нефункціональні вимоги визначають якості, властивості та обмеження системи, які не стосуються конкретних функцій, які система повинна виконувати. Вони описують атрибути, такі як продуктивність, надійність, безпека, зручність використання та сумісність, що забезпечують якість роботи системи та її зручність для користувачів.

Іншими словами, нефункціональні вимоги описують "як" система виконує свої функції, а не "що" саме вона робить [19].

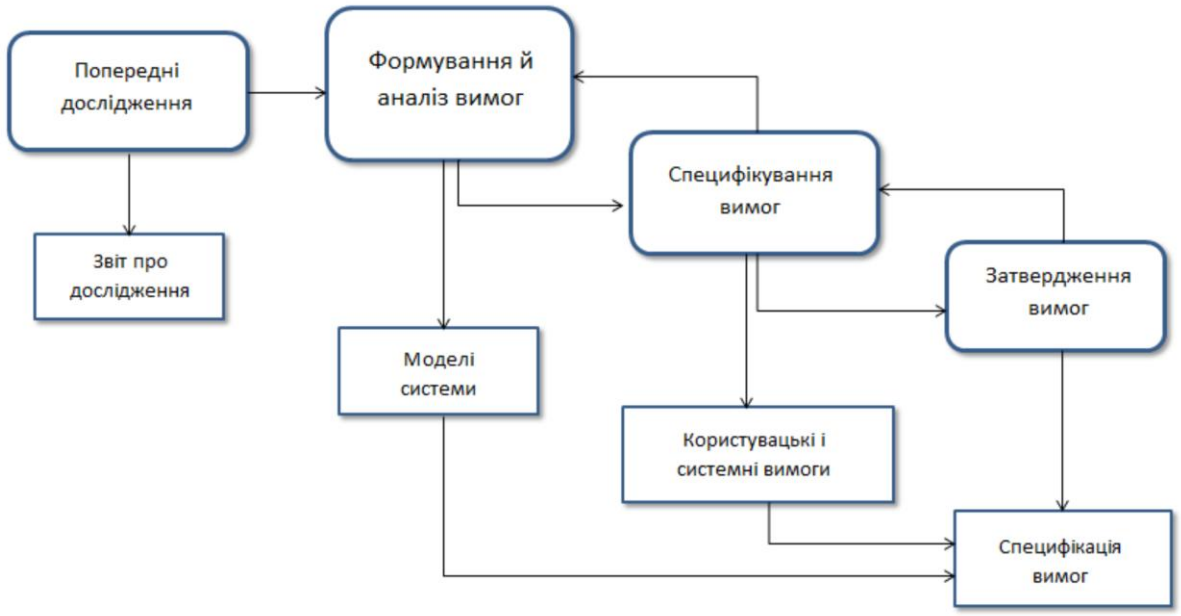


Рисунок 2.1 Розробка специфікації вимог

До таких показників можна віднести: продуктивність (система повинна забезпечувати швидке введення та обробку даних абітурієнтів), надійність (доступність системи - система повинна бути доступною для оператора протягом 99.9% часу роботи приймальної комісії, .у разі технічного збою система повинна забезпечувати можливість відновлення даних без втрати інформації), .безпека (конфіденційність даних - дані абітурієнтів повинні бути захищені від несанкціонованого доступу), зручність використання (інтерфейс повинен бути інтуїтивно зрозумілим і зручним для використання, з урахуванням потреб операторів приймальної комісії), сумісність (система повинна працювати на всіх сучасних версіях Windows та мати можливість інтеграції з іншими інформаційними системами університету через стандартні API), масштабованість(система повинна бути розроблена таким чином, щоб нові функції могли бути легко додані без значних змін у вже існуючій архітектурі), портативність (:дані повинні мати можливість легкого переносення на інші системи або сервери при необхідності зміни обладнання чи платформи).

Виконання цих нефункціональних вимог забезпечить високу якість, ефективність та надійність системи, зробивши її зручною і безпечною для використання в роботі приймальної комісії [20].

Сформулюємо функціональні та нефункціональні вимоги до системи «Кабінет відповідального працівника приймальної комісії»:

Назва системи: Кабінет відповідального працівника приймальної комісії.

Цільова аудиторія: Технічні працівники приймальної комісії навчального закладу, які відповідають за обробку та оформлення документів абітурієнтів.

Система має забезпечити можливість автоматичної генерації різних документів, таких як договори про навчання, контракти, заяви на проживання у гуртожитку та інші, на основі введених даних про абітурієнтів та їхніх умов навчання [21].

Система повинна мати інтуїтивно зрозумілий та зручний інтерфейс для введення даних, вибору необхідних опцій та взаємодії з користувачем. Інтерфейс має бути адаптований для використання на різних пристроях, включаючи комп'ютери, планшети та смартфони.

Система повинна мати можливість підключення до бази даних, де будуть зберігатися дані про абітурієнтів, їхні вступні та особисті дані. Це дозволить автоматично отримувати необхідну інформацію [22].

Система має надавати можливість надрукувати готові документи у вигляді PDF-файлів.

Система повинна забезпечувати швидке введення та обробку даних абітурієнтів.

Орієнтовно тестування та впровадження системи планується завершити до початку наступної приймальної кампанії [23].

Враховуючи вищезазначені вимоги, інформаційна система "Кабінет відповідального працівника приймальної комісії" має за мету спростити та оптимізувати процес обробки документів для абітурієнтів, забезпечуючи високу продуктивність та зручність користування для працівників приймальної комісії.

2.3 Формалізація системи

2.3.1 Діаграма використання

Діаграма прецедентів представляє функціональне призначення системи, відповідаючи на основне питання моделювання: що робить система у зовнішньому світі? Вона використовує два типи базових сутностей: варіанти використання (прецеденти) і діючі особи (актори), між якими встановлюються певні типи відношень. Це можуть бути: асоціації між актором і прецедентом, узагальнення між акторами, узагальнення між прецедентами, залежності чи асоціації різних типів між прецедентами [24].

Прецеденти визначаються за специфікацією вимог до системи. Актором може бути будь-який зовнішній суб'єкт, який взаємодіє із системою: фізична особа, зовнішня програмна система чи пристрій. Усі процеси взаємодії між діючими суб'єктами і системою розглядаються як прецеденти.

Діаграми прецедентів відображають статичні аспекти системи з позиції користувачів. Ця позиція охоплює поведінку системи, тобто видимі ззовні сервіси, які вона надає [25].

На початкових стадіях моделювання статичних аспектів системи діаграми прецедентів застосовують у двох напрямках. Для моделювання контексту системи, що дозволяє умовно відмежувати систему і виявити діючих осіб, які знаходяться за цією межею і взаємодіють із системою - діаграми прецедентів на цьому етапі необхідні для ідентифікації акторів і визначення їхніх ролей. Для моделювання вимог до системи: це допомагає визначити, що система повинна робити з позиції зовнішнього користувача, незалежно від того, як саме вона це буде реалізовувати. Діаграми прецедентів на цьому етапі служать для ідентифікації бажаної поведінки системи, представляючи її як "чорну скриньку".

Визначивши акторів системи та процесів взаємодії між ними і системою, можна визначити вимоги системи.

Абітурієнт – подає документи та підписує друковані примірники документів.

Технічний працівник приймальної комісії – вносить і превіряє персональні дані абітурієнтів, роздруковує необхідні документи.

Пристрій для друку (принтер) – забезпечує можливість друку заповнених примірників документів [26].

Банк даних – отримує від системи деякі дані абітурієнтів для зберігання.

Побудована діаграма використання зображена на рисунку 2.2.

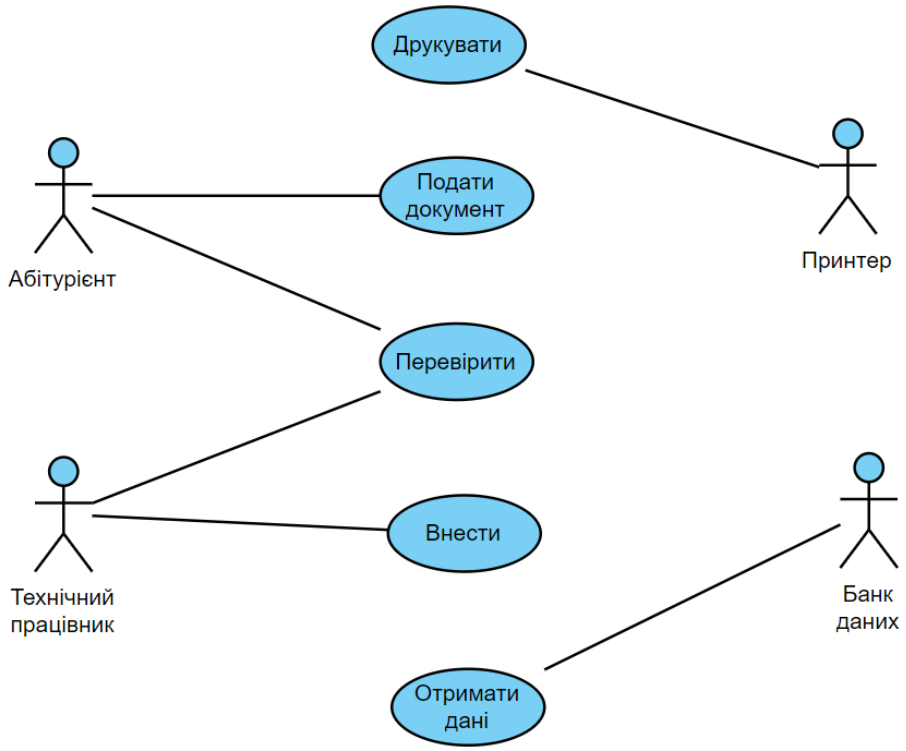


Рисунок 2.2 – Діаграма прецедентів системи

2.3.2 Діаграма діяльності

Діаграми дій (activity diagrams) відображають динамічні аспекти проекту, представляючи схеми потоків управління в системі від однієї дії до іншої, а також паралельні дії та альтернативні потоки. На різних етапах життєвого циклу

проекту, ці діаграми можуть демонструвати потоки між функціями або всередині окремої функції, відображаючи послідовність виконання операцій [27].

Діаграми дій ілюструють дії, переходи між ними, елементи вибору та лінії синхронізації. У мовах моделювання, таких як UML (Unified Modeling Language), дії зображуються у вигляді прямокутників із закругленими кутами. Переходи між діями позначаються стрілками, що вказують на потік управління. Елементи вибору представлені ромбами, які вказують на розгалуження потоку, а лінії синхронізації, що символізують паралельні процеси або їх злиття, зображуються товстими горизонтальними або вертикальними лініями [28].

Ці діаграми є важливим інструментом для розробників і аналітиків, оскільки вони допомагають візуалізувати і зрозуміти логіку виконання процесів, виявити можливі проблеми та оптимізувати роботу системи.

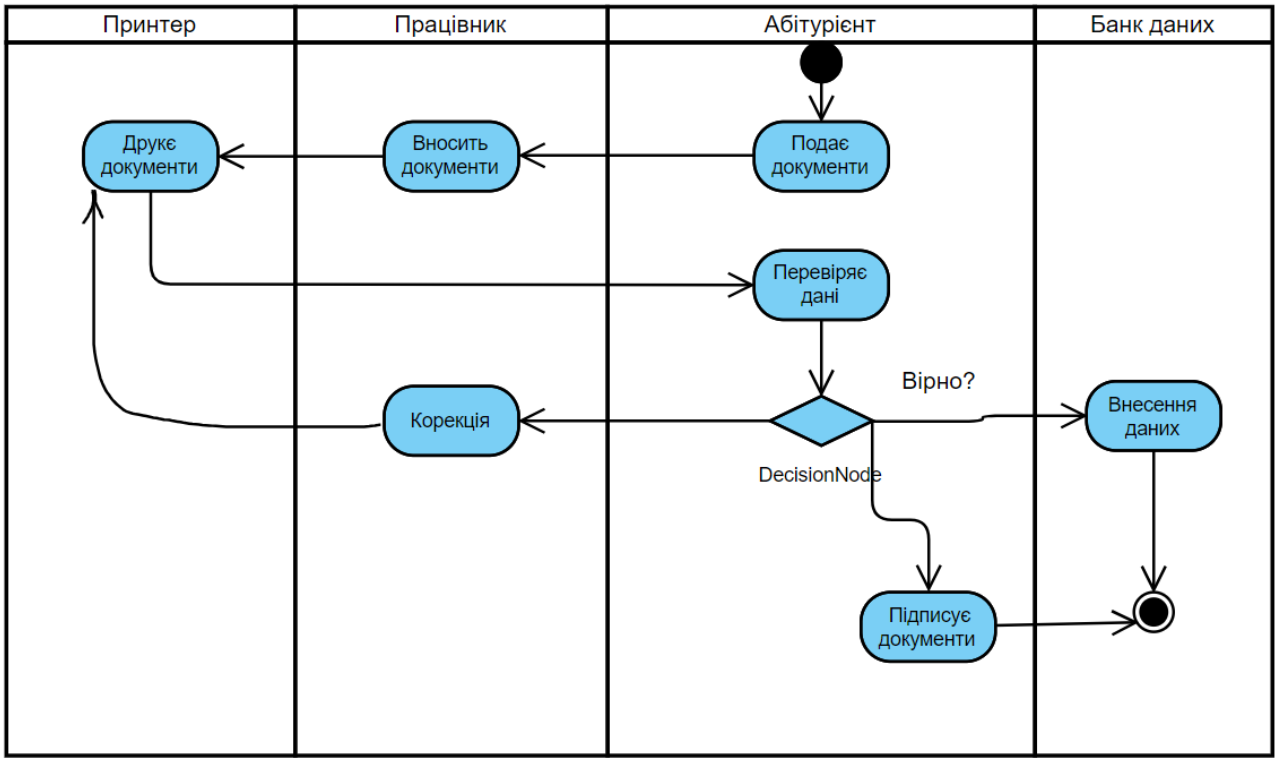


Рисунок 2.3 – Діаграма діяльності

2.3.3 Діаграма послідовності

Діаграма послідовності (Sequence Diagram) використовується для моделювання динамічних аспектів системи, зокрема для опису того, як об'єкти системи взаємодіють між собою через передачу повідомлень у певній послідовності. Ці діаграми є частиною мови моделювання UML і широко застосовуються для аналізу і проєктування систем [29].

Діаграми послідовності можуть бути використані для:

1.документування взаємодії між об'єктами або компонентами в системі, показуючи, як вони співпрацюють для виконання певної функції або бізнес-процесу.

2.уточнення і перевірки вимог до системи, дозволяючи виявити необхідні об'єкти і їхню взаємодію на ранніх стадіях проєктування.

3.проєктування логіки роботи системи, зокрема для розробки алгоритмів, сценаріїв використання, а також для визначення відповідальності різних об'єктів за виконання певних дій [30].

4.комунікації між членами команди розробників, полегшуючи розуміння складних процесів і забезпечуючи спільне бачення логіки роботи системи.

5.верифікації правильності проєкту, а також для розробки сценаріїв тестування, забезпечуючи, що всі взаємодії об'єктів відповідають очікуванням [31].

Основні елементи діаграми послідовності представлені внизу.

Актори - представляють зовнішніх користувачів або інші системи, що взаємодіють із системою.

Об’єкти - представляють компоненти системи, які беруть участь у виконанні сценарію [32].

Життєві лінії - вертикальні лінії, що відображають існування актора або об'єкта протягом часу. Життєва лінія показує, коли об'єкт активний і здатний приймати або відправляти повідомлення.

Повідомлення - горизонтальні стрілки, що представляють комунікацію між об'єктами. Повідомлення можуть бути синхронними (виконання блокується до отримання відповіді) або асинхронними (виконання продовжується без очікування відповіді) [33].

Активація - тонкі прямокутники на життєвій лінії, які показують період, протягом якого об'єкт виконує операцію або чекає на завершення операції.

У системі діаграма послідовності може показати взаємодію між технічним працівником приймальної комісії, системою автоматичного заповнення документів, банком даних і модулем друку документів [34].

Технічний працівник вводить дані абітурієнта в систему.

Система автоматичного заповнення документів генерує необхідні документи.

Система автоматичного заповнення документів зберігає ці дані в банку даних.

Модуль друку отримує згенеровані документи і друкує їх.

Цей сценарій допомагає зрозуміти послідовність дій, необхідних для обробки заявки, і виявити потенційні місця для оптимізації процесу [35].

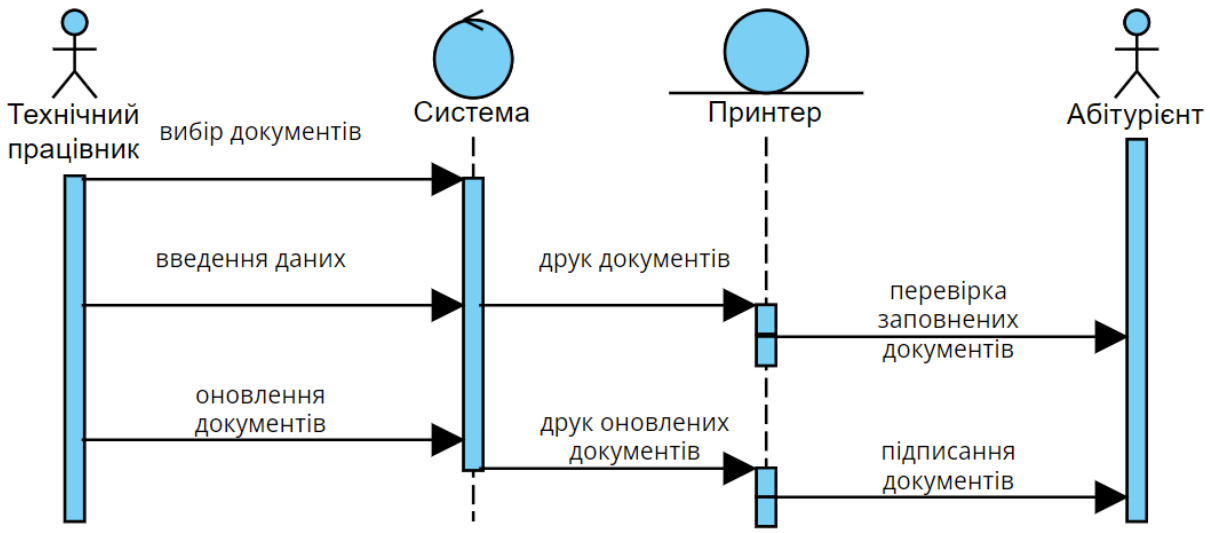


Рисунок 2.4 – Діаграма послідовності

2.4 Висновки

У другому розділі було детально спроектовано інформаційну систему "Кабінет відповідального працівника приймальної комісії". Це включало створення кількох видів діаграм, що дозволило всебічно проаналізувати та візуалізувати основні аспекти функціонування системи [36].

Одним із ключових етапів проектування було побудування діаграми варіантів використання. Ця діаграма дала можливість визначити основні функціональні можливості системи та взаємодію між різними акторами і системою. Зокрема, було виділено основні сценарії роботи системи, такі як введення даних, автоматичне заповнення документів, друк документів. Діаграма варіантів використання дозволила окреслити межі системи та визначити ролі основних користувачів, таких як працівник приймальної комісії та абітурієнт [37].

Наступним кроком було побудування діаграми дій, яка відображає динаміку проекту та схеми потоків управління в системі від дії до дії. Діаграма дій дозволила детально розглянути послідовність операцій, які виконує система під час обробки даних. Це включало початковий етап введення даних абітурієнтів, їх верифікацію, автоматичне рознесення по різним документам та кінцевий етап – друк готових документів. Крім того, діаграма дій візуалізувала паралельні процеси та альтернативні потоки, що можуть відбуватися під час роботи системи, наприклад, повернення до попередніх етапів для коригування введених даних.

Діаграма послідовностей була використана для деталізації взаємодії між різними компонентами системи та її користувачами в контексті виконання конкретних функцій. Ця діаграма допомогла ілюструвати порядок обміну повідомленнями між об'єктами системи під час виконання певних сценаріїв. Наприклад, для сценарію введення даних абітурієнта було показано, як працівник приймальної комісії вводить інформацію, система автоматично розміщує їх у відповідні документи, а потім надсилає повідомлення про завершення операції.

Діаграма послідовностей також дала можливість побачити, як окремі частини системи взаємодіють одна з одною, що є важливим для забезпечення узгодженості та ефективності роботи системи [38].

У результаті проведеного моделювання було досягнуто кілька важливих результатів. По-перше, було чітко визначено функціональні вимоги до системи "Кабінет вдповідального працівника приймальної комісії", що є основою для подальшого етапу розробки. Моделювання дозволило виявити потенційні проблеми та неузгодженості на ранньому етапі, що значно знижує ризики виникнення серйозних помилок під час реалізації системи.

По-друге, створення діаграм забезпечило візуальне представлення структури та динаміки системи.

По-третє, моделі, створені в рамках цього розділу, служать важливими документами, які можна використовувати як основи для написання технічної документації, розробки тестових сценаріїв. Це забезпечує безперервність проекту та полегшує подальші етапи розробки і впровадження системи [39].

3 РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ "КАБІНЕТ ВІДПОВІДАЛЬНОГО ПРАЦІВНИКА ПРИЙМАЛЬНОЇ КОМІСІЇ"

3.1 Проектування інтерфейсу користувача

Інтерфейс користувача (User Interface, UI) — це сукупність засобів та методів, які забезпечують взаємодію користувача з програмним або апаратним забезпеченням. Інтерфейс користувача охоплює все, з чим користувач може взаємодіяти для керування та використання системи, включаючи елементи керування, дисплеї, індикатори та інші пристрої введення-виведення.

Інтерфейс користувача відіграє критичну роль у забезпеченні зручності та ефективності роботи з системою, а також впливає на загальний користувацький досвід (UX). Хороший UI повинен бути інтуїтивно зрозумілим, доступним та функціональним, забезпечуючи легкість і ефективність виконання завдань.

Графічний інтерфейс користувача використовує візуальні елементи, такі як вікна, іконки, меню, кнопки та інші графічні компоненти, для взаємодії з користувачем. GUI є найбільш поширеним типом інтерфейсу в сучасних операційних системах та програмному забезпеченні.

Використовуючи командний інтерфейс користувач вводить текстові команди, які система виконує. CLI зазвичай використовується адміністраторами систем та розробниками через його ефективність для виконання складних завдань.

Інтерфейс на основі жестів використовує сенсорні екрани або пристрої розпізнавання жестів для взаємодії з системою. Прикладами є смартфони та планшети.

Голосовий інтерфейс дозволяє користувачам взаємодіяти з системою через голосові команди. Прикладом є голосові помічники, такі як Siri, Alexa, Google Assistant.

Інтерфейс доповненої та віртуальної реальності забезпечує взаємодію через окуляри або шоломи доповненої чи віртуальної реальності ,створюючи інтерактивне тривимірне середовище.

Інтерфейс користувача відіграє критичну роль у забезпеченні зручності та ефективності роботи з системою, а також впливає на загальний користувацький досвід (UX). Хороший UI повинен бути інтуїтивно зрозумілим, доступним та функціональним, забезпечуючи легкість і ефективність виконання завдань.

Інтерфейс користувача може складатися з різних компонентів, які забезпечують взаємодію користувача з системою. Основні складові інтерфейсу користувача включають:

Вхідні пристрої: клавіатура (використовується для введення тексту та команд); миша (дозволяє користувачеві вибирати та маніпулювати об'єктами на екрані); сенсорний екран (дозволяє взаємодію за допомогою дотику); голосові пристрої (мікрофони для введення голосових команд).

Вихідні пристрої: монітор (основний дисплей для візуалізації інформації); принтер (для виведення інформації на папір); динаміки (для виведення звуку).

Елементи графічного інтерфейсу (GUI): вікна: основні області, де відображається інформація та з якими взаємодіє користувач; іконки (графічні символи, які представляють програми, файли або функції); меню (списки опцій або команд, які користувач може вибрати); кнопки (елементи, які можна натискати для виконання певних дій); панелі інструментів (набори кнопок або іконок, які забезпечують швидкий доступ до часто використовуваних функцій); форми (інтерфейси для введення та перегляду даних); смуги прокручування (дозволяють переміщуватися по вмісту, який не вміщується на екрані)[26].

Інтерактивні елементи: поля для введення тексту; перемикачі (радіо-кнопки); прапорці (чекбокси) –дозволяють вибрати одну або декілька опцій; випадаючі списки –дозволяють вибрати одну опцію з декількох у списку.

Навігаційні елементи: гіперпосилання(дозволяють переходити між різними сторінками або частинами програми); навігаційні панелі (допомагають користувачеві орієнтуватися в програмі або веб-сайті).

Зворотний зв'язок та повідомлення: повідомлення про помилки (інформують користувача про виникнення проблем); спливаючі підказки (тултіпи)

(надають додаткову інформацію про елементи інтерфейсу); інформаційні повідомлення (надають корисну інформацію або підтверджують виконання дій).

Дизайн та макет: теми та стилі (визначають загальний вигляд та відчуття інтерфейсу, включаючи кольори, шрифти та макети); анімація та переходи (забезпечують плавність переходів між різними станами інтерфейсу).

Додаткові елементи: панелі стану (відображають поточний стан програми або пристрою); прогрес-бари (показують прогрес виконання завдань).

Врахувавши висунуті вимоги, було вирішено зупинитись на макеті інтерфейсу, зображеному на рисунку 3.1

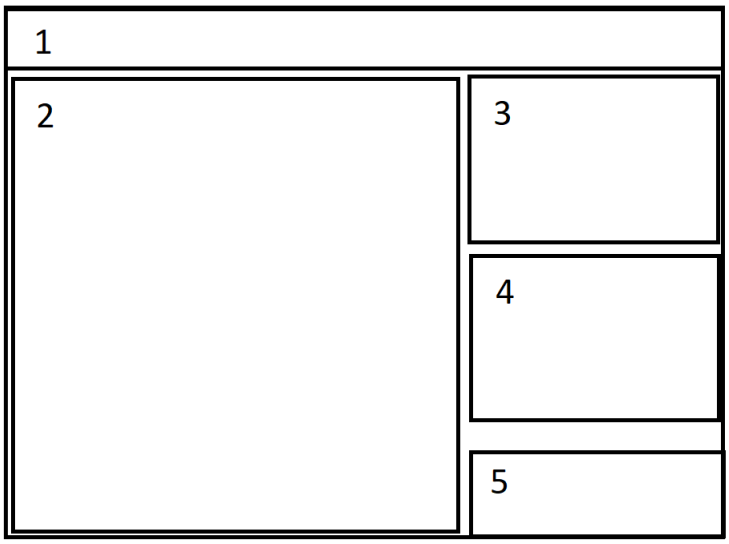


Рисунок 3.1 – Макет Головного вікна системи

Опис компонентів головного вікна згідно їх нумерації на рисунку 3.1 наведено нижче.

1. Назва програми та елементи керування вікном програми.
2. Відображає документи або набори документів, які потребують заповнення
3. Група елементів керування наборами документів.
4. Група елементів керування шаблонами документів.
5. Додаткові елементи в програмі.

3.2 Вибір середовища розробки

У даному розділі я розгляну вибір і налаштування середовища розробки для створення інформаційної системи "Кабінет відповідального працівника приймальної комісії".

Розробка цієї системи проводилася на мові програмування C# з використанням Windows Forms у середовищі Visual Studio. Це середовище було вибрано завдяки його потужним інструментам для створення графічних інтерфейсів, інтеграції з різними базами даних та зручності для розробників.

Мова програмування C# є однією з найбільш популярних мов для розробки додатків під операційну систему Windows. Вона має високу продуктивність, підтримує об'єктно-орієнтоване програмування та має широкий спектр бібліотек для вирішення різноманітних завдань.

Windows Forms – це технологія, яка дозволяє швидко створювати додатки з графічним інтерфейсом. Вона підтримує розробку складних форм та інтерфейсів користувача, забезпечуючи інтеграцію з іншими компонентами .NET Framework.

Visual Studio є одним із найбільш потужних та популярних середовищ розробки, особливо для мови C#. Воно пропонує великий набір інструментів для розробки, тестування та налагодження програмного забезпечення. Серед основних переваг Visual Studio можна виділити: інтуїтивно зрозумілий інтерфейс – надає зручні інструменти для розробки, які значно спрощують процес створення додатків; інтеграція з системами контролю версій – Visual Studio легко інтегрується з такими системами, як Git, що спрощує керування версіями коду; підтримка розширень – існує велика кількість розширень, які додають нові функції та покращують роботу розробників; потужні інструменти для налагодження – дозволяють швидко знаходити та виправляти помилки в коді.

Після детального проектування інформаційної системи було вирішено, що інформаційна система буде локальною однокористувацькою системою. У такій системі всі дані та програми зберігаються і виконуються на одному комп'ютері, і лише один користувач має до неї доступ.

Для розуміння структури та функціональності системи важливо ознайомитися з окремими компонентами, які складають її інтерфейс та забезпечують користувачам системи можливість взаємодії з системою.

У таблиці 3.1 описано основні класи системи, їх призначення та особливості.

Таблиця 3.1 – Основні класи системи

Клас	Опис
Templates	Реалізує логіку створення шаблонів документів
DocumentSet	Реалізує логіку формування комплекту документу для кожного абітурієнта, надає можливість формування pdf-документу для друку, надає можливість передавання даних в банк даних
DataEnter	Призначений для внесення і запам’ятовування даних. Реалізує автоматичне внесення даних в комплект документів

Діаграми класів (рисунок 3.2-3.5) описують ключові компоненти, їхні основні функції та призначення в контексті функціоналу системи.

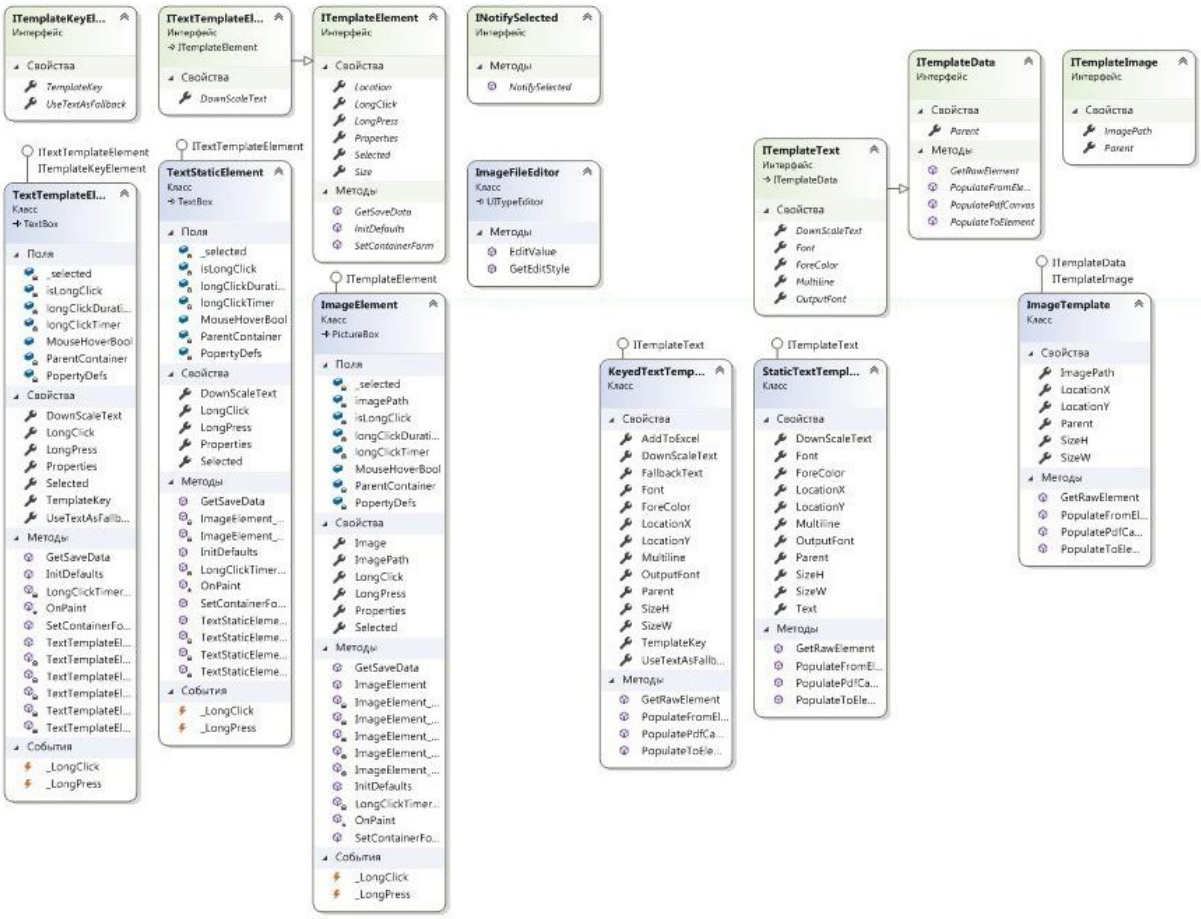


Рисунок 3.2 – Діаграма класів

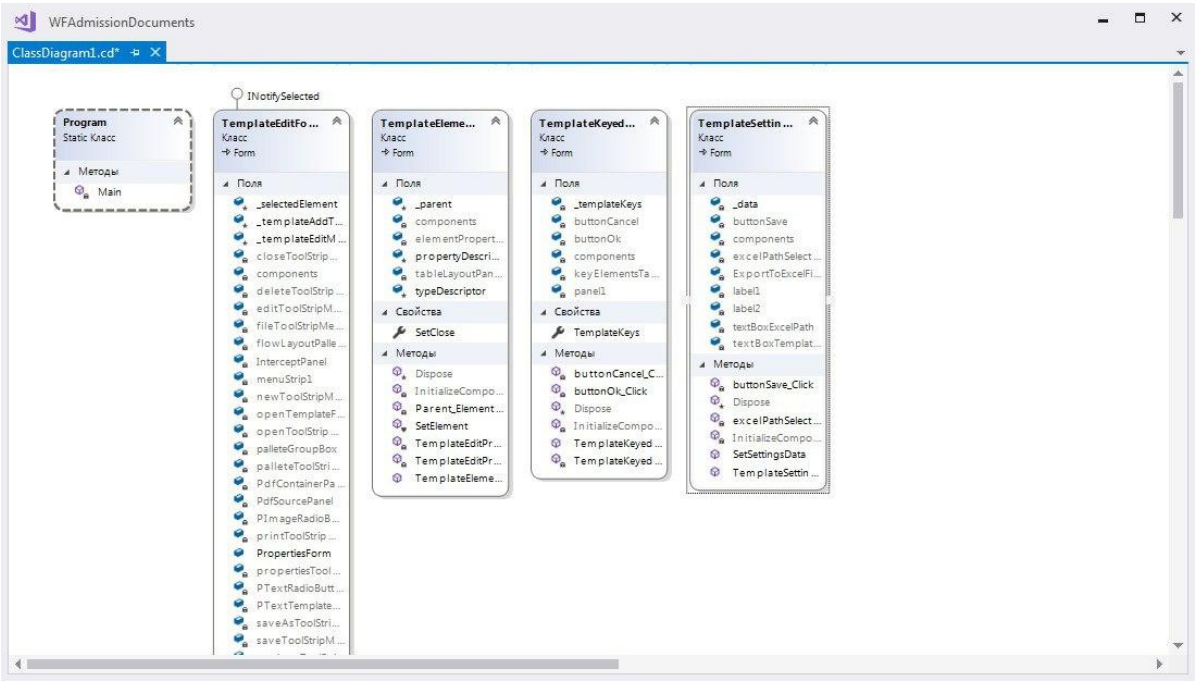


Рисунок 3.2 – Модель класів для редактора шаблонів сторінок (фрагмент)

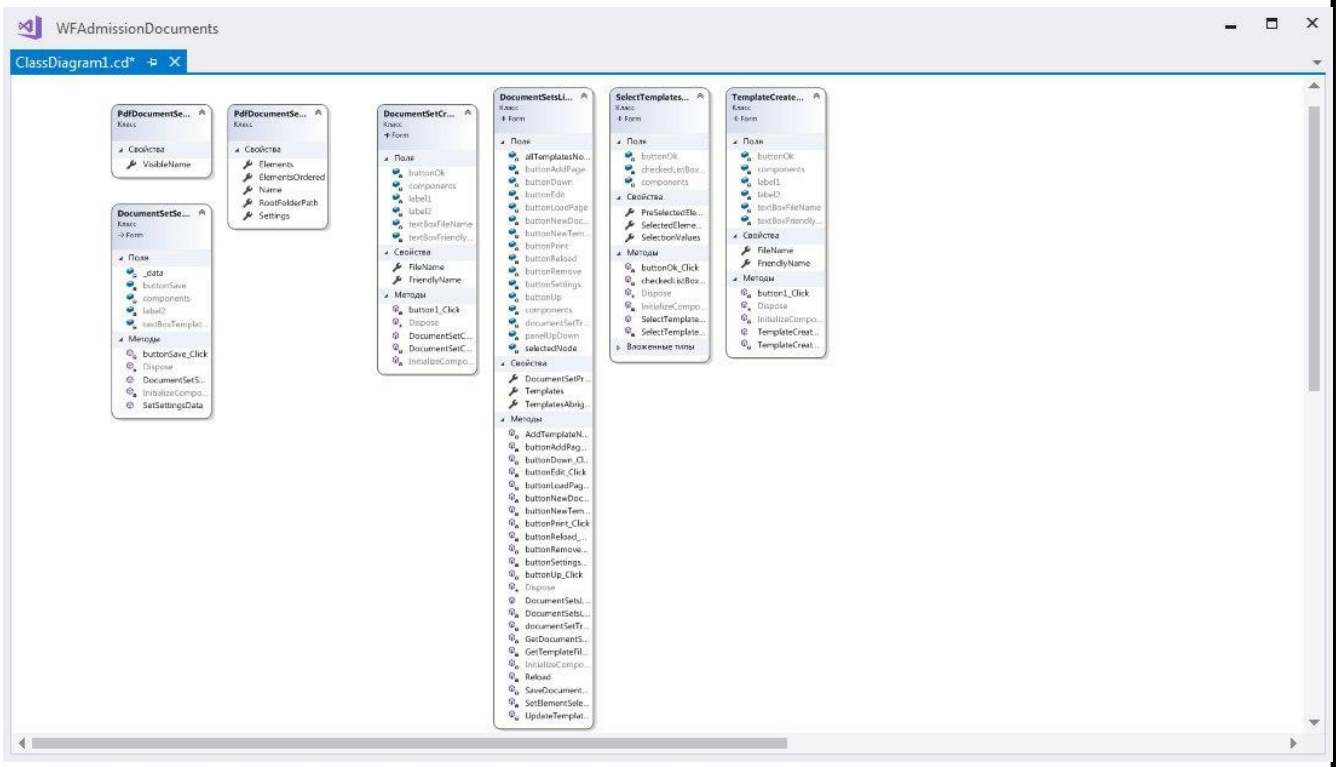


Рисунок 3.3 – Модель класів для редагування наборів документів

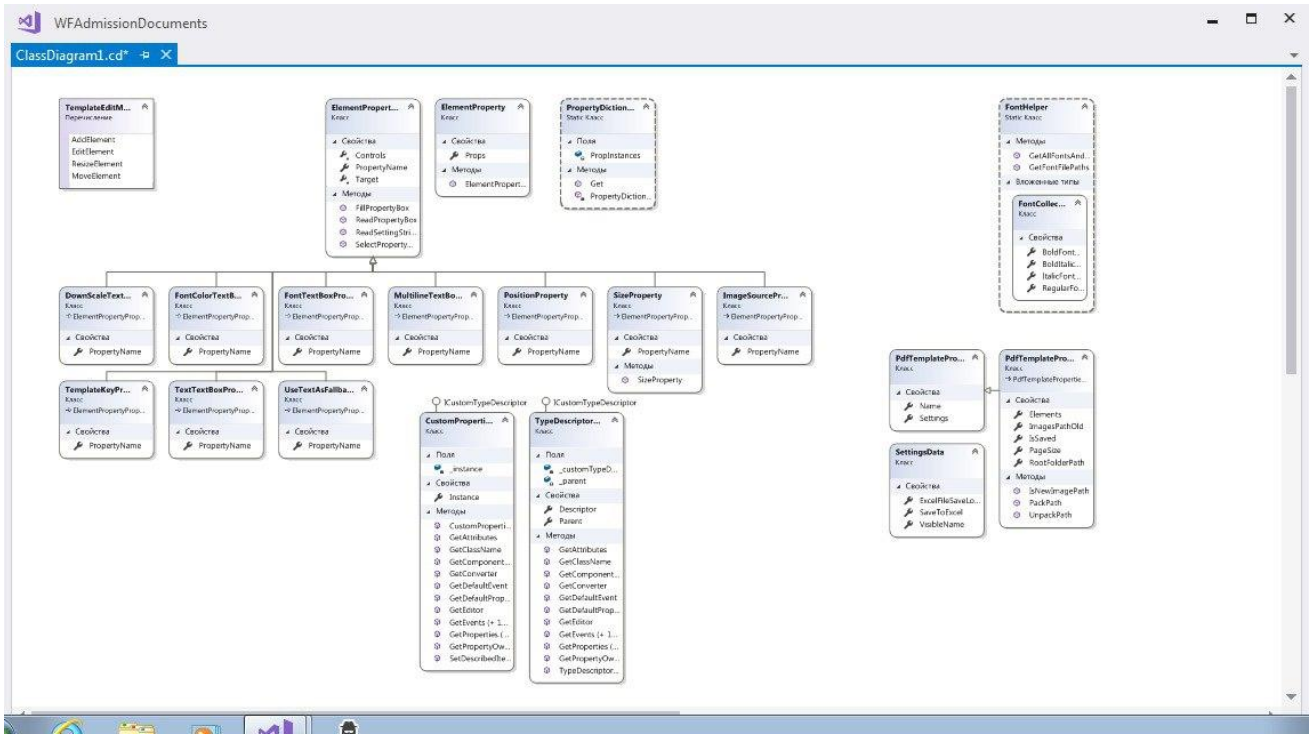


Рисунок 3.4 – Шаблони класів для елементів редактора

Рисунок 3.5 – Шаблони класів для властивостей редагованих елементів

3.3 Реалізація інтерфейсу користувача в програмному середовищі

Створення інтерфейсу користувача (UI) у програмному середовищі з використанням Windows Forms є захоплюючим і багатоетапним процесом. Він вимагає ретельного планування і творчого підходу, щоб забезпечити зручну та інтуїтивно зрозумілу взаємодію користувачів із програмою. Розробка починається з вибору та налаштування середовища розробки, і найчастіше для цього використовується Visual Studio. Ця потужна платформа надає всі необхідні інструменти для розробки, відлагодження та тестування додатків, забезпечуючи зручний та ефективний робочий процес.

Використовуючи візуальний дизайнер, можна просто перетягувати елементи управління, такі як кнопки, текстові поля, мітки та списки, на форму. Кожен з цих елементів відіграє важливу роль у взаємодії з користувачем, надаючи інтуїтивно зрозумілі засоби для введення та відображення інформації.

Основне вікно програми, відоме як форма, є контейнером для всіх інших елементів управління. Контролі можуть бути налаштовані відповідно до потреб проекту, змінюючи їхні властивості, такі як ім'я, текст, розмір та розташування. Це дозволяє створити унікальний та ефективний інтерфейс, який відповідає специфічним вимогам користувача.

Один із ключових аспектів розробки інтерфейсу – це обробка подій. Кожен елемент управління може генерувати різноманітні події, такі як натискання

кнопки або зміна тексту в текстовому полі. Обробка цих подій дозволяє програмі реагувати на дії користувача, виконуючи необхідні функції, такі як обробка введених даних, взаємодія з базою даних або генерування звітів.

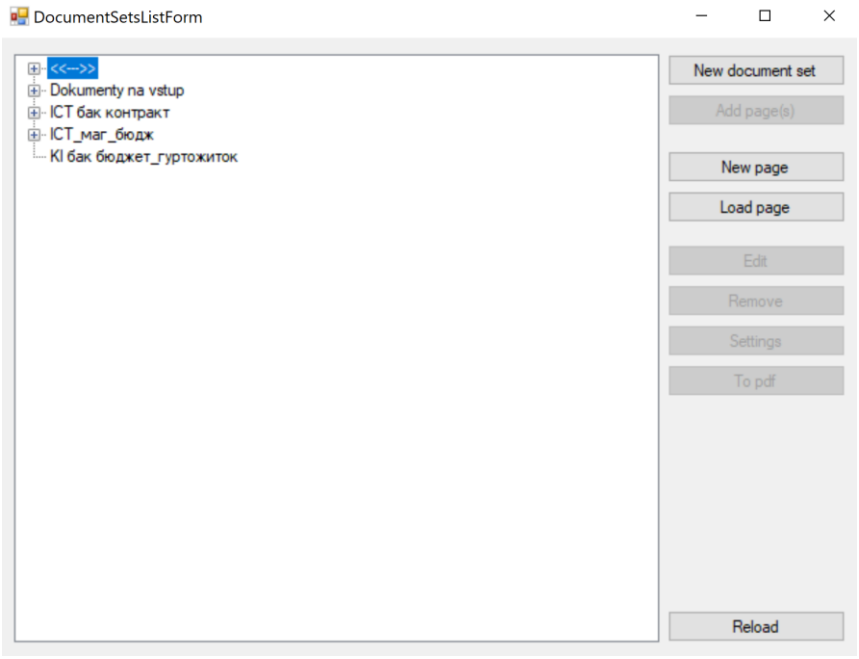


Рисунок 3.6 – Реалізований інтерфейс головного вікна системи

3.4 Програмна реалізація основних модулів системи

Програмна реалізація системи "Кабінет відповідального працівника приймальної комісії" здійснюється з використанням мови програмування C# і платформи .NET з Windows Forms. У цьому розділі розглянемо основні модулі системи, їхню функціональність та реалізацію.

Можемо виділити такі модулі системи:

1. Модуль створення шаблону документу.
2. Модуль створення набору документів.
3. Модуль введення та зберігання даних.
4. Модуль взаємодії з принтером.
5. Модуль взаємодії з банком даних.

3.4.1 Модуль створення шаблону документу

Завдання модуля створення шаблону документу полягає в створенні шаблону для заповнення друкованого бланку. Логіка реалізації наступна: модуль отримує на вхід зображення документу (якщо бланк документу є двостороннім, то тоді буде шаблон для кожної сторінки), кваліфікований користувач в потрібні для заповнення місця проставляє потрібні елементи форми (такі як, поля для введення тексту, чек-бокси, випадаючі списки).

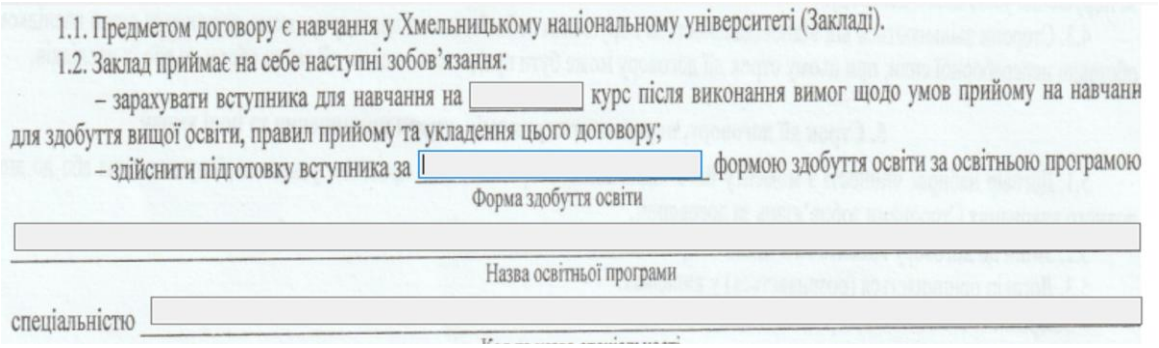


Рисунок 3.7 – Фрагмент шаблону документу із нанесеними полями для введення даних

У випадку співпадіння зачень поля з полем іншого документів у властивостях даного поля має прописуватись ключ, який є ідентичним до вже існуючого поля (це в майбутньому дозволить одночасне заповнення всіх полів при одноразовому введенні даних). Шаблони зберігатимуться в папці програми, до якої інші модулі матимуть доступ.

Нижче наведено фрагменти реалізації деяких елементів модуля створення шаблону документу. Реалізацію всього модуля створення шаблону документу для системи «Кабінет відповідального працівника приймальної комісії» наведено в додатку Г.

```
{
    public partial class TemplateEditForm : Form, INotifySelected
    {
        public bool FileEdit { get; set; } = false;
        public string FileToOpen { get; set; }

        #region Edit Part
```

```
        public event EventHandler ElementSelected;
        public TemplateElementPropertiesForm PropertiesForm;
        protected Type _templateAddType;
        protected TemplateEditMode _templateEditMode =
TemplateEditMode.EditElement;
        protected TemplateEditMode TemplateEditMode {
            get => _templateEditMode;
            set
            {
                if (value != _templateEditMode)
                {
                    _templateEditMode = value;
                    UpdateEditMode();
                }
            }
        }
```

3.4.2 Модуль створення набору документів

Під час подачі документів абітурієнту необхідно власноруч заповнити декілька бланків, дані в яких зазвичай дублюються, іноді потрібно дублювати і самі бланки. Деякі бланки потребують заповнення всіма абітурієнтами, а деякі потрібно заповнювати за власною потребою, як то Заява на гуртожиток тощо. Зручно буде формувати комплект необхідних документів для кожного абітурієнта.

Функціональні задачі модуля створення набору документів полягають в об’єднанні потрібних документів в групу, щоб автоматизувати заповнення всього набору документів.

Логіка функціонування модуля наступна: оператор системи створює набір шаблонів документів, які відповідатимуть одній із типових ситуацій (наприклад, абітурієнт вступає на спеціальність «Інформаційні системи і технооології» на бюджетну форму навчання, гуртожиток потрібен), відповідно до набору документів будуть входити необхідні документи (в продовження прикладу, до набору документів увійдуть: договір про навчання, опис документів у справі, заява на гуртожиток).

Після формування набору документу модуль дозволить вносити зміни у всі документи набору. Редагувати набір документів – у нашому випадку означає внести персональні дані в даний комплект документів. Набір документів може бути використано також і для іншого абітурієнта.

Також він може бути доповнений іншими документами.

Нижче наведено фрагменти реалізації деяких елементів форми модуля створення набору документів. Реалізацію всього модуля створення набору документів для системи «Кабінет відповідального працівника приймальної комісії» наведено в додатку Г.

3.4.3 Модуль введення та зберігання даних

Даний модуль призначено для реалізації можливості введення даних в шаблон документу. Нижче наведено програмний код класу, що відповідає за введення даних в шаблон документу

```
partial class DataEnterTemplateForm
    {
        private System.ComponentModel.IContainer components = null;
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }
        #region Windows Form Designer generated code
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.AutoScaleMode =
System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(800, 450);
            this.Text = "DataEnterTemplateForm";
        }
    }
```

Можливість заповнення набору документів реалізована за допомогою формування нового вікна, яке містить унікальні поля всього набору документів (наприклад, поле з імя’м абітурієнта зустрічається у всіх документах, але ми його

заповнюємо лише раз в цій новоствореній формі). Програмний код наведено в додатку Г.

3.4.4 Модуль взаємодії з принтером

Даний модуль дозволяє друк документів. Реалізовано його із використанням стандартних засобів Windows для друку. Після формування .pdf-файлу із внесеними даними про абітурієнта, цей файл може бути роздруковано звичайним чином у за стосунку для перегляду .pdf-файлів[34].

```
//
            // buttonPrint
            //
            this.buttonPrint.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorSty
les.Top | System.Windows.Forms.AnchorStyles.Right)));
            this.buttonPrint.Location = new
System.Drawing.Point(486, 237);
            this.buttonPrint.Name = "buttonPrint";
            this.buttonPrint.Size = new System.Drawing.Size(129,
23);
            this.buttonPrint.TabIndex = 9;
            this.buttonPrint.Text = "To pdf";
            this.buttonPrint.UseVisualStyleBackColor = true;
            this.buttonPrint.Click += new
System.EventHandler(this.buttonPrint_Click);
```

3.4.5 Модуль взаємодії з банком даних

Під час внесення даних в набір документів, технічний працівник використовує персональні дані абітурієнтів, які можуть бути також перенесені в банк даних. Банк даних реалізовано у вигляді таблиці MS Excel. В електронних таблицях MS Excel передбачено можливість фільтрування даних, що надасть змогу формувати деякі звіти про хід вступної компанії, але не обтяжить інтерфейс системи додатковими елементами.

3.5 Тестування інформаційної системи

Після завершення дизайну інтерфейсу і написання основної логіки програми настає етап тестування та відлагодження. Visual Studio надає потужні інструменти для виявлення та виправлення помилок у коді, що дозволяє забезпечити високу якість та надійність готового продукту. Тестування допомагає переконатися, що всі елементи інтерфейсу працюють належним чином і що програма виконує всі необхідні функції відповідно до вимог користувачів.

Тестування інформаційної системи є критичним етапом у процесі її розробки та впровадження. Воно включає в себе кілька важливих аспектів, спрямованих на забезпечення якості, надійності та відповідності системи вимогам користувачів та бізнесу. Суть тестування інформаційної системи можна розглянути з кількох ключових перспектив.

Однією з основних цілей тестування є виявлення помилок або багів у системі до її впровадження в реальне середовище. Тестувальники проводять різні види тестування, такі як функціональне, регресійне, інтеграційне та стресове тестування, щоб знайти можливі дефекти в коді. Виявлені помилки документуються, передаються розробникам для виправлення, а потім перевіряються повторно.

Тестування також дозволяє переконатися, що інформаційна система відповідає визначеним вимогам і специфікаціям. Це означає, що всі функціональні та нефункціональні вимоги, встановлені на початкових етапах розробки, виконані належним чином. Тестувальники створюють тестові сценарії та випадки на основі цих вимог, щоб перевірити їх виконання.

Система повинна не тільки функціонувати коректно, але й бути здатною обробляти задані навантаження без втрати продуктивності. Тестування продуктивності включає оцінку швидкості, масштабованості та стійкості системи під різними умовами експлуатації. Це допомагає визначити, чи зможе система справлятися з очікуваними обсягами даних та користувачів.

Існує багато різних видів тестування, кожен з яких має свої цілі, методи та підходи. Основні види тестування включають: функціональне тестування (це тип тестування, який перевіряє, чи відповідає функціонал системи встановленим вимогам.

Основна мета — перевірка кожної функції програми, щоб переконатися, що вона працює належним чином); нефункціональне тестування (цей вид тестування спрямований на перевірку нефункціональних аспектів системи, таких як продуктивність, масштабованість, надійність, безпека та зручність використання); тестування продуктивності (це тип нефункціонального тестування, який визначає, наскільки швидко система працює під певним навантаженням.

Включає в себе тестування навантаження, стрес-тестування та об'ємне тестування); тестування безпеки (мета цього тестування — виявити вразливості в системі, захистити дані від несанкціонованого доступу та забезпечити відповідність вимогам безпеки); тестування зручності використання (Usability Testing) (це вид тестування, що оцінює, наскільки зручно та інтуїтивно зрозуміло користувачам працювати з системою.

Включає оцінку інтерфейсу, навігації та загального користувацького досвіду) та інші.

3.5.1 Функціональне тестування

Функціональне тестування — це процес перевірки програмного забезпечення, що забезпечує відповідність його функціональності встановленим вимогам і специфікаціям. Основна мета функціонального тестування — гарантувати, що кожна функція програмного продукту працює правильно і дає очікувані результати.

Таблиця 3.2 – Аналіз тестових сценаріїв та тестових випадків

Тестовий випадок	Очікуваний сценарій	Реальний результат
Запуск програми	Після активації команди запуску, завантажується головна сторінка системи.	Завантажилась головна сторінка
Завершення роботи	Після натискання керяючого елементу«Вихід» головне вікно системи закрилось.	Головне вікно закрилось після відповіді на питання про збереження змін.
Створення нового шаблону документу	Після натискання «New page» відкриється сторінка створення нового шаблону документа	Перехід в інше вікно відбувся
Створення набору документів для одночасного заповнення	Після натискання «New document set» відкриється вікно для назви цього набору документів	Перехід в інше вікно відбувся. При спробі перейти далі, не ввівши ім’я нового набору документів, система сповістила про помилку
Додавання нових документів до набору (новоствореного або існуючого)	При виділенні потрібного набору документів і натискані «Add page(s)» відкриється вікно зі вже створеними шаблонами, в якому за допомогою чек-боксів можна обрати шаблони необхідних документів	Перехід в інше вікно відбувся, зявилась можливість обирати потрібні документи.

Кінець таблиці 3.2

Автоматичне заповнення декількох документів	Після обрання будь-якого документу з набору документів і натискання «Edit», відкриється шаблон документу. Пройшовши в меню вікна шлях File->To pdf відкриється вікно, що містить необхідні для заповнення поля всього набору документів	Перехід в інше вікно відбувся, зявилась можливість заповнення полів всіх документів.

Після виконання наведених в таблиці 3.2 тестів отримані результати порівнюються з очікуваними результатами. Відхилень, неочікуваної поведінки не виявлено. Фкціональне тестування вважаємо успішним.

Слід зазначити, що в процесі розробки функціонал системи також перевірявся, і помилки, що виникали, були успішно виправлені.

3.5.2 Тестування графічного інтерфейсу системи

Тестування графічного інтерфейсу є невід'ємною частиною процесу забезпечення якості програмного забезпечення. Воно дозволяє гарантувати, що всі елементи інтерфейсу працюють належним чином і відповідають вимогам користувачів. Використання як ручного, так і автоматизованого тестування забезпечує високу точність і ефективність перевірки, дозволяючи виявити та виправити помилки ще до початку активної експлуатації продукту кінцевими користувачами. Частина тестів, які висувались до інтерфейсу системи наведено в таблиці 3.3

Таблиця 3.3 – Аналіз тестування графічного інтерфейсу

Опис тесту	Кроки виконання	Очікуваний результат	Фактичний результат	Статус
Перевірка відкриття головного вікна	1. Запустити програму 2. Перевірити відкриття головного вікна	Головне вікно програми відкривається без помилок	Головне вікно програми відкривається без помилок	Пройдено
Перевірка кнопки «New page»	1.Натиснути кнопку 2. Перевірити відкриття вікна для створення нового шаблону	Перехід в інше вікно відбувся	Перехід в інше вікно відбувся	Пройдено
Перевірка кнопки «New document set»	1.Натиснути кнопку 2. Перевірити відкриття вікна для створення назви нового набору документів New Document Set. 3. Перевірити як у вікні New Document Set працює кнопка Ok	Після натискання «New document set» відкриється вікно для назви цього набору документів Після натискання Ok у вікні New Document Set у випадку порожнього поля система має сповістити, що File name cannot be empty.	Перехід в інше вікно відбувся. При спробі перейти далі, не ввівши ім’я нового набору документів, система сповістила про помилку	Пройдено

Продовження таблиці 3.3

Перевірка кнопки «Add page(s)»	1. Обрати набір документів в робочій області головного вікна 2.Натиснути кнопку «Add page(s)» 3. Перевірити відкриття вікна зі вже створеними шаблонами, в якому за допомогою чек-боксів можна обрати шаблони необхідних документів 3. Перевірити роботу чек-боксів 4. Натиснути кнопку Ok 5. Перевірити чи відбулось додавання шаблонів у обраний набір	При виділенні потрібного набору документів і натискані «Add page(s)» відкриється вікно зі вже створеними шаблонами, в якому за допомогою чек-боксів можна обрати шаблони необхідних документів	Перехід в інше вікно відбувся, зявилась можливість обирати потрібні документи за допомогою чек-боксів, після натискання Ok обрані документи відобразились в структурі набору документів.	Пройдено

Кінець таблиці 3.3

Перевірка кнопки «Edit»	1. Обрати набір документів 2. Обрати довільний документ набору 3. Натиснути кнопку 4. Перевірити чи відобразився даний документ із графічними об’єктами, такі як: поле введення, випадаючий список тощо. 5. Обрати в меню вікна шлях File->To pdf Перевірити, чи зявилась можливість зберегти файл 6. Перевірити чи відкрилось після збереження файлу вікно для заповнення полів всього набору документів	Після обрання будь-якого документу з набору документів і натискання «Edit», відкриється шаблон документу. Пройшовши в меню вікна шлях File->To pdf відкриється вікно, що містить необхідні для заповнення поля всього набору документів. Після заповнення полів і натискання Ok, в катлозі файлів відобразився файл із всіма заповненими документами (всі документи в одному файлі)	Перехід в інше вікно відбувся, зявилась можливість заповнення полів всіх документів. Збережений .pdf файл містить всі документи, які були аавтоматично заповнені системою	Пройдено

Слід зазначити, що в процесі розробки системи навігація вікон і робочій стан всіх графічних елементів системи перевірявся, і невідповідності, що виникали, були усунуті.

Після реалізації наведених в таблиці 3.3 тестових сценаріїв отримані результати виявились очікуваними.

Відхилень, неочікуваної поведінки під час цього тестування також не виявлено. Тестування інтерфейсу вважаємо успішним.

3.6 Висновки розділу

У цьому розділі було детально описано процес реалізації інформаційної системи "Кабінет відповідального працівника приймальної комісії" та проведено її тестування. Розробка системи здійснювалась з використанням мови програмування C# та середовища розробки Visual Studio, а інтерфейс користувача реалізовано на базі Windows Forms.

Під час реалізації системи було приділено особливу увагу створенню інтуїтивно зрозумілого та функціонального інтерфейсу, який забезпечує ефективну взаємодію оператора з програмою.

Процес реалізації системи включав кілька важливих етапів, серед яких проектування інтерфейсу користувача, програмування основних функціональних модулів та їх інтеграція в єдину систему.

Особливий акцент було зроблено на автоматизації процесу заповнення документів, що значно знижує час та зусилля, необхідні для виконання рутинних операцій.

Система дозволяє оператору вводити дані абітурієнтів, які автоматично розносяться по кількох документам, готовим до друку.

Після завершення етапу реалізації було проведено тестування системи, яке включало як функціональне, так і нефункціональне тестування. Функціональне

тестування охоплювало перевірку всіх основних функцій системи, зокрема введення даних, автоматичне заповнення документів, друк.

Результати тестування підтвердили, що система працює стабільно та коректно виконує всі поставлені завдання.

Під час тестування графічного інтерфейсу недоліків також не виявлено.

Оцінка ефективності системи "Кабінет відповідального працівника приймальної комісії" показала, що її впровадження дозволить значно оптимізувати роботу приймальної комісії, скоротити час на обробку документів, зменшити кількість помилок, пов'язаних з ручним введенням даних, та підвищити загальну продуктивність праці.

Автоматизація рутинних процесів звільняє час працівників, дозволяючи їм зосередитися на більш важливих завданнях, таких як консультування абітурієнтів та прийняття рішень щодо вступу.

ВИСНОВКИ

У дипломній роботі проведено всебічне дослідження предметної області, змодельовано та реалізовано інформаційну систему "Кабінет відповідального працівника приймальної комісії".

Метою даної роботи було створення ефективного інструменту для автоматизації процесів приймальної комісії університету, що дозволяє оптимізувати роботу з документами абітурієнтів, зменшити кількість помилок та підвищити загальну продуктивність праці.

На початковому етапі роботи було детально вивчено предметну область, зокрема аналіз існуючих процесів прийому документів, обробки даних абітурієнтів та підготовки звітності.

Було визначено основні проблеми та недоліки, пов'язані з ручною обробкою документів, які включають значні витрати часу, високу ймовірність помилок та складність в управлінні великими обсягами даних.

Для вирішення цих проблем було спроектовано інформаційну систему, яка базується на сучасних технологіях та методах розробки програмного забезпечення.

У ході моделювання системи було створено декілька видів діаграм, включаючи діаграму варіантів використання, діаграму дій та діаграму послідовностей.

Ці діаграми допомогли чітко визначити функціональні вимоги до системи, зрозуміти взаємодію між різними компонентами та користувачами системи, а також спланувати реалізацію основних функцій.

Реалізація системи здійснювалась з використанням мови програмування C# та середовища розробки Visual Studio. Інтерфейс користувача було реалізовано на базі Windows Forms, що забезпечує зручність та інтуїтивність у роботі.

Особлива увага приділялась створенню інтерфейсу, який дозволяє працівникам приймальної комісії легко вводити дані абітурієнтів, автоматично розносити їх по необхідних документах та генерувати звіти. Інтерфейс був спроектований таким

чином, щоб мінімізувати час, необхідний на виконання рутинних завдань, та знизити ризик помилок.

Після реалізації системи було проведено її тестування, яке підтвердило, що система коректно виконує всі поставлені завдання, забезпечуючи автоматичне заповнення документів.

Таким чином, результати дипломної роботи демонструють, що розроблена інформаційна система є ефективним інструментом для автоматизації процесів приймальної комісії, що відповідає сучасним вимогам та забезпечує високу продуктивність та надійність у роботі.

У подальшому розвиток системи може бути спрямований на додавання нових функцій, інтеграцію з іншими інформаційними системами університету та вдосконалення інтерфейсу користувача.

Також перспективним напрямком є розширення функціоналу системи для підтримки онлайн-платформ та клієнт-серверної архітектури, інтеграція з електронною системою подання документів абітурієнтами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Литвиненко С. В. Автоматизація процесів прийому студентів: нові підходи та технології. *Освіта та управління*. 2022. № 4. С. 25-30.

2. Коваленко О. М., Мельник В. П. Використання інформаційних технологій у роботі приймальних комісій. *Інформаційні технології* в освіті. 2021. № 2. С. 40-45.

3. Гончаренко Т. В. Автоматизовані системи управління приймальними комісіями у вищих навчальних закладах України. *Вісник Київського національного університету ім. Т. Шевченка.* 2023. № 1. С. 58-63.

4. Шевченко Ю. А. Цифрові інструменти для оптимізації прийому студентів. *Інформаційні системи та технології в освіті*. 2022. № 3. С. 70-75.

5. Петренко В. О. Підвищення ефективності роботи приймальних комісій за допомогою інформаційних систем. *Науковий вісник Національного університету біоресурсів і природокористування України*. 2023. № 5. С. 35-41.

6. Іванова Н. В., Кравченко П. М. Інформаційні системи в управлінні приймальними комісіями: аналіз і перспективи. *Вісник Харківського національного* університету *ім. В. Н. Каразіна*. 2022. № 2. С. 82-88.

7. Романюк А. І., Заболотний О. В. Використання автоматизованих систем у процесі прийому абітурієнтів. *Науковий вісник Чернівецького національного університету*. 2023. № 3. С. 50-56.

8. Miller, F. Implementing AI for Document Automation in SMEs. *Small Business Technology Journal*, 2021. Vol. 21. P. 89-106.

9. Davis, P. Machine Learning Approaches to Document Automation." *AI & Society*, 2020. Vol.19 №4. P.2-19.

10. Anderson, O. Document Automation Tools: A Comparative Study. *Software Engineering Today*, 2021. Vol. 6. №4. P. 46-48.

11. Gupta, S. Cloud-based Solutions for Document Automation. *Journal of Cloud Computing*, 2019. P. 145-160.

12. Lee, J. Automating Legal Documents with Natural Language Processing. *Legal AI Journal*, 2020. Vol. 112. № 2. P.164-198.

13. Martin, R. Document Automation in Government Agencies. *Public Administration Review*, 2018. Vol. 23. №5. P. 564- 574.

14. Taylor, G. Improving Efficiency with Document Automation. *Operations Management Journal*, 2019. Vol. 28. №3. P. 4-70.

15. Green, M. Transforming Document Management with AI. *Journal of AI and Automation*, 2018.

16. Adams, B. Workflow Optimization Through Document Automation. *Business Process Management Journal*, 2020. P. 2-16.

17. Patel, N. Automated Document Generation in Education. Journal of *Educational Technology*, 2019. Vol. 15. №3 P. 2-13.

18. Turner, S. AI and Document Processing in the Legal Field. *Journal of Legal Technology*, 2021. P. 595-613.

19. Robinson, A. Efficiency Gains from Document Automation. *Journal of Business Efficiency*, 2020. Vol. 8. №11. P. 3-19.

20. Carter, J. Document Automation for HR Processes. *Human Resources Management Journal*, 2019. Vol. 21. №1. P. 1-12.

21. Wilson, M. Exploring Automated Documentation Tools. *Journal of Computer Science*, 2020. Vol. 58. №1. P. 2-24.

22. Harris, E. Document Automation: Challenges and Opportunities. *Journal of Automation Science*, 2021. P. 413- 423.

23. White, K. Automated Documentation in the Insurance Industry. *Insurance Technology Review*, 2019. Vol. 1. №1. P. 2–21.

24. Thomas, D. Automated Document Solutions in Banking. *Banking Technology Journal*, 2021. Vol. 16. №3. P. 2-20.

25. Evans, S. Advances in Document Automation Technology. *Journal of Advanced Technologies*, 2018. Vol. 30. №1. P. 61-74.

26. Walker, J. AI-Driven Document Management. *Artificial Intelligence Journal*, 2019. Vol. 15. №8. P. 2-6.

27. Mitchell, P. Scalability of Automated Document Systems. *Journal of Information Systems*, 2020. P. 5-25.

28. Clarke, R. Automating Legal Documentation Processes. *Journal of Legal Studies*, 2019. Vol. 35. №4. P. 1-7.

29. Young, H. Document Automation for Startups. *Startup Technology Review*, 2021. Vol. 23. №2. P. 3-19.

30. Nelson, E. Cloud-based Document Automation Solutions. *Cloud Computing Journal*, 2020. Vol. 28. №5. P. 704-722.

31. Cooper, A. Impact of AI on Document Creation. *AI Impact Review*, 2019. P. 3-10.

32. Wilson, C. Future Trends in Document Automation. *Tech Trends Journal*, 2020. Vol. 9. №1. P. 91-102.

33. М. Сидоров, Ефективні процедури прийому, *University Admissions Review*, 2020. Вип. 23. С. 3–7.

34. N. Ivanov, Development of Admission Office Systems, *Information Systems Journal*, 2018. Vol. 24. №2. P. 17-54.

35. Kozlov, Information Systems for Admission Offices, *Higher Education IT Journal*, 2019. Vol. 7. №10. P. 222-229.

36. P. Ivanov, Designing an Effective Admission Office System, *Information Management Journal*, 2020. Vol. 78. №4. P. 266-274.

37. Sokolov, Admission Office Digital Transformation, *Digital Transformation in Education*, 2021. P. 2-14.

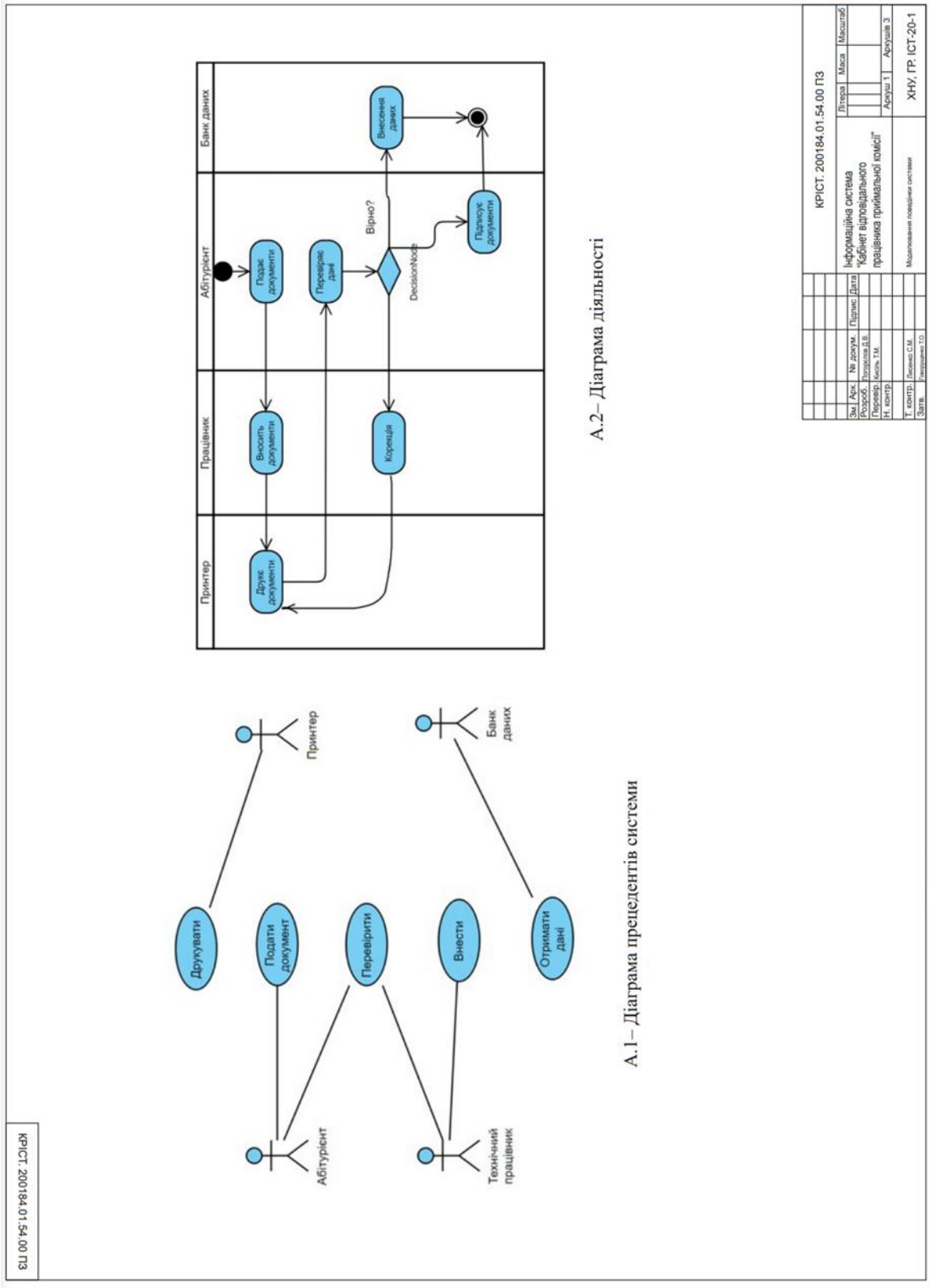
38. T. Orlov, Software Solutions for University Admissions, *Software Development Journal*, 2019. Vol. 58. №1167. P. 64-71.

39. M. Pavlov, Information Systems in Higher Education Admissions, *Journal of Education Systems*, 2020. P. 111-128.

ДОДАТОК А

(обов’язковий)

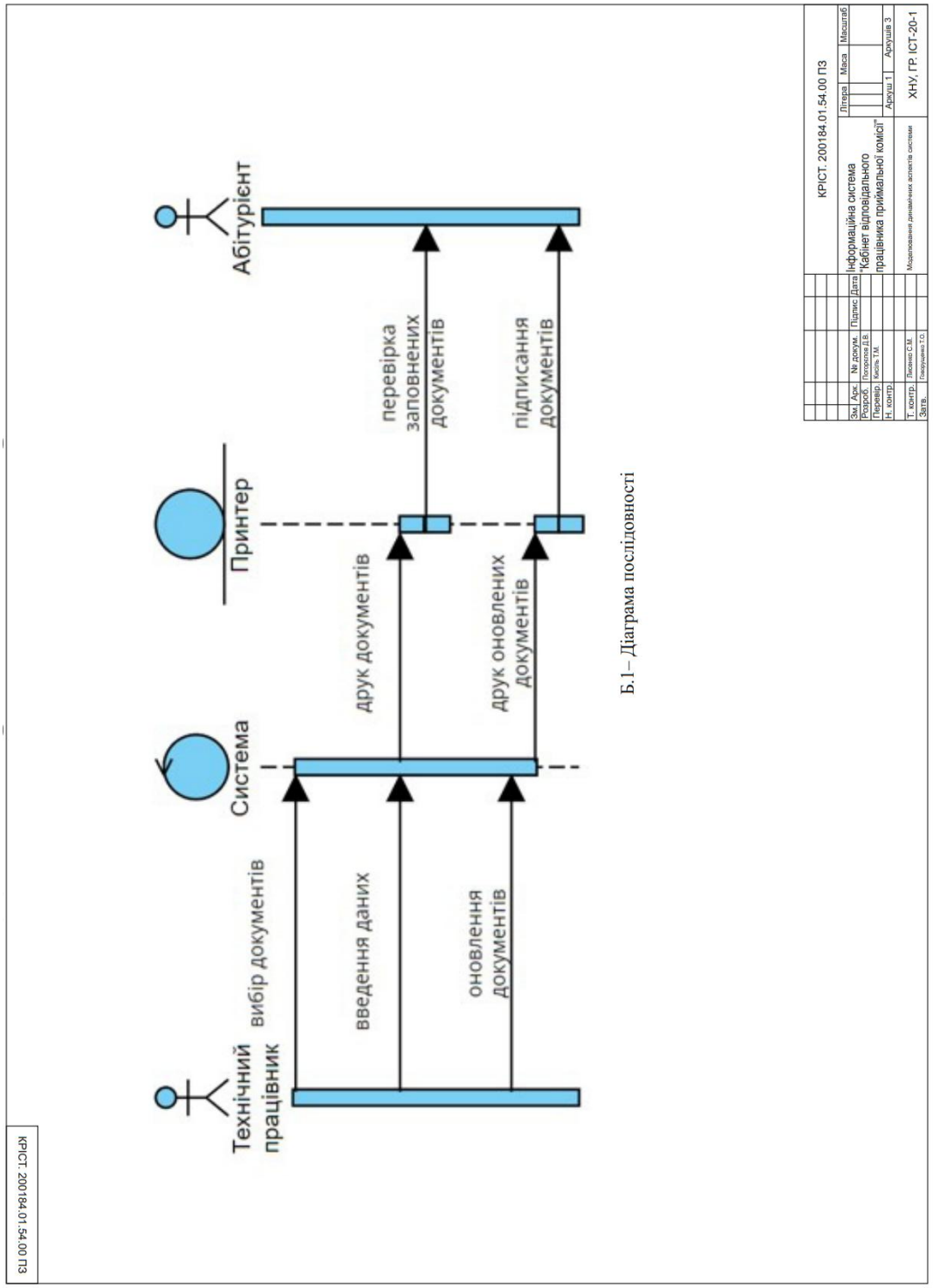
Копія креслення «Моделювання поведінки системи»



ДОДАТОК Б

(обов'язковий)

Копія креслення «Моделювання динамічних аспектів системи»

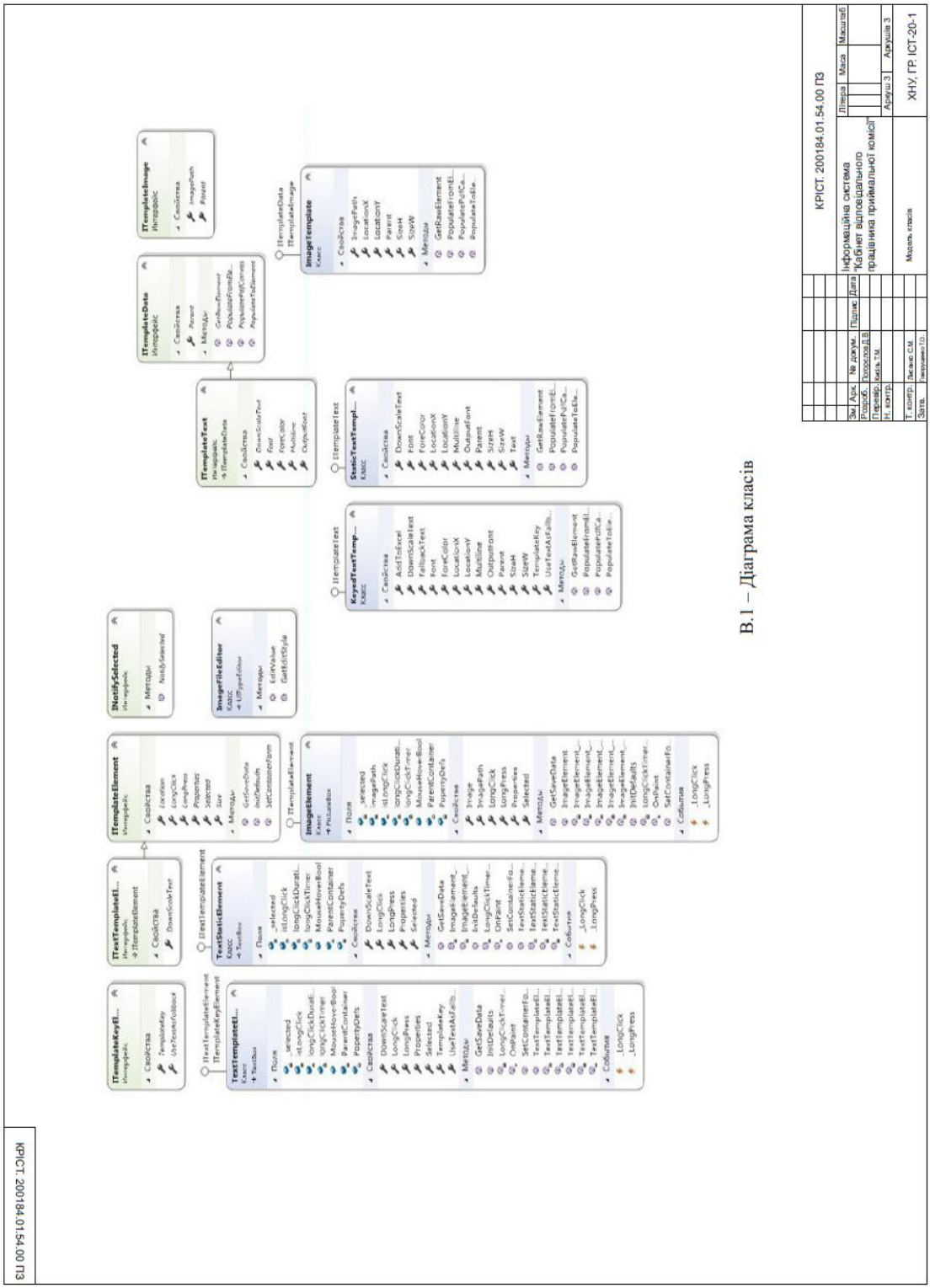


Б.1 – Діаграма послідовності

ДОДАТОК В

(обов'язковий)

Копія креслення «Модель класів»



В.1 – Діаграма класів

ДОДАТОК Г

(обов'язковий)

Лістинг основних модулів системи

Модуль редагування шаблонів

```
using iText.Kernel.Pdf;
using iText.Kernel.Pdf.Canvas;
using iText.Layout;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Windows.Forms;
using WFAdmissionDocuments.Code;
using WFAdmissionDocuments.Code.TemplateStuff;
using WFAdmissionDocuments.Code.TemplateStuff.TemplateData;
using WFAdmissionDocuments.Code.TemplateStuff.TemplateData.Interfaces;
using WFAdmissionDocuments.Templates;
using WFAdmissionDocuments.Templates.TemplateElements;

namespace WFAdmissionDocuments
{
    public partial class TemplateEditForm : Form, INotifySelected
    {
        public bool FileEdit { get; set; } = false;
        public string FileToOpen { get; set; }

        #region Edit Part
        public event EventHandler ElementSelected;
        public TemplateElementPropertiesForm PropertiesForm;
        protected Type _templateAddType;
        protected TemplateEditMode _templateEditMode =
TemplateEditMode.EditElement;
        protected TemplateEditMode TemplateEditMode {
            get => _templateEditMode;
            set
            {
                if (value != _templateEditMode)
                {
                    _templateEditMode = value;
                    UpdateEditMode();
                }
            }
        }
        protected ITemplateElement _selectedElement;
        public ITemplateElement SelectedElement
        {
            get => _selectedElement; set
            {
                if (value != _selectedElement)
                {
                    _selectedElement = value;
                    ElementSelected?.Invoke(_selectedElement, null);
                }
```

```
            }
        }

        public TemplateEditForm()
        {
            InitializeComponent();
        }

        private void AddEditTemplateForm_Load(object sender, EventArgs e)
        {
            //pdfGenerator.PageWidth = 210; // in millimeters
            //pdfGenerator.PageHeight = 297; // in millimeters

            var size = new Size
            {
                Width = SizeUtils.MillimetersToPixels(210),
                Height = SizeUtils.MillimetersToPixels(297),
            };

            PdfSourcePanel.Size = size;
            InterceptPanel.Size = size;

            PropertiesForm = new TemplateElementPropertiesForm(this);
            PropertiesForm.VisibleChanged += PropertiesForm_VisibleChanged;

            InitializeEmpty();

            if (FileEdit && !string.IsNullOrWhiteSpace(FileToOpen) &&
File.Exists(FileToOpen))
            {
                newToolStripMenuItem.Visible = false;
                openToolStripMenuItem.Visible = false;

                OpenFile(FileToOpen);
            }
        }

        private void PropertiesForm_VisibleChanged(object sender, EventArgs e)
        {
            propertiesToolStripMenuItem.Checked = PropertiesForm.Visible;
        }

        private void InitializeEmpty()
        {
            PTextTemplateRadioButton.Checked = false;
            PTextRadioButton.Checked = false;
            PImageRadioButton.Checked = false;

            TemplateEditMode = TemplateEditMode.EditElement;
            SelectedElement = null;
            TemplateProperties = null;
            PdfSourcePanel.Controls.Clear();

            InterceptPanel.SendToBack();
            InterceptPanel.BackgroundImage = null;
            InterceptPanel.Capture = false;
        }

        //TODO: should do it recursively, fix later
        private Bitmap DrawControlsToBitmap(Control parentControl)
        {
            Bitmap bitmap = new Bitmap(parentControl.Width,
parentControl.Height);
```

```
            using (Graphics graphics = Graphics.FromImage(bitmap))
            {
                parentControl.DrawToBitmap(bitmap, new Rectangle(0, 0,
PdfSourcePanel.Width, PdfSourcePanel.Height));

                for (int i = parentControl.Controls.Count - 1; i >= 0; i--)
                {
                    Control control = parentControl.Controls[i];

                    control.DrawToBitmap(bitmap, control.Bounds);
                }
            }
            return bitmap;
        }

        private void UpdateEditMode()
        {
            if (TemplateEditMode == TemplateEditMode.AddElement)
            {
                Bitmap bmp = DrawControlsToBitmap(PdfSourcePanel); //new
Bitmap(PdfSourcePanel.Width, PdfSourcePanel.Height);

                //Draw the control onto the Bitmap
                //PdfSourcePanel.DrawToBitmap(bmp, new Rectangle(0, 0,
PdfSourcePanel.Width, PdfSourcePanel.Height));

                InterceptPanel.BackgroundImage = bmp;
                InterceptPanel.BringToFront();
                InterceptPanel.Capture = true;
                InterceptPanel.Cursor = Cursors.Cross;
            }
            else
            if (TemplateEditMode == TemplateEditMode.EditElement)
            {
                InterceptPanel.SendToBack();
                InterceptPanel.BackgroundImage.Dispose();
                InterceptPanel.BackgroundImage = null;
                InterceptPanel.Capture = false;

                var checkBoxType = typeof(CheckBox);
                foreach (var child in flowLayoutPalletePanel.Controls)
                {
                    if (checkBoxType.IsAssignableFrom(child.GetType())) {
                        (child as CheckBox).Checked = false;
                    }
                }
            }
            else
            if (TemplateEditMode == TemplateEditMode.MoveElement)
            {
                Bitmap bmp = DrawControlsToBitmap(PdfSourcePanel);
                //Bitmap bmp = new Bitmap(PdfSourcePanel.Width,
PdfSourcePanel.Height);
                //PdfSourcePanel.DrawToBitmap(bmp, new Rectangle(0, 0,
PdfSourcePanel.Width, PdfSourcePanel.Height));

                InterceptPanel.BackgroundImage = bmp;
                InterceptPanel.BringToFront();
                InterceptPanel.Capture = true;
                InterceptPanel.Cursor = Cursors.SizeAll;
            }
            else
            if (TemplateEditMode == TemplateEditMode.ResizeElement)
```

```
            {
                Bitmap bmp = DrawControlsToBitmap(PdfSourcePanel);
                //Bitmap bmp = new Bitmap(PdfSourcePanel.Width,
PdfSourcePanel.Height);
                //PdfSourcePanel.DrawToBitmap(bmp, new Rectangle(0, 0,
PdfSourcePanel.Width, PdfSourcePanel.Height));

                InterceptPanel.BackgroundImage = bmp;
                InterceptPanel.BringToFront();
                InterceptPanel.Capture = true;
            }
        }

        private void InterceptPanel_MouseClick(object sender, MouseEventArgs e)
        {
            Point position = e.Location;
            if (_templateAddType != null)
            {
                Control newElement =
(Control)Activator.CreateInstance(_templateAddType);

                newElement.Location = position;

                InitElement(newElement as ITemplateElement);

                _templateAddType = null;
                TemplateEditMode = TemplateEditMode.EditElement;
            }
        }

        private void InitElement(ITemplateElement newElement)
        {
            (newElement as ITemplateElement).InitDefaults();
            (newElement as ITemplateElement).SetContainerForm(this);
            (newElement as ITemplateElement).LongPress += Element_LongPress;

            PdfSourcePanel.Controls.Add(newElement as Control);
            PdfSourcePanel.Controls.SetChildIndex(newElement as Control, 0);
        }

        private void SetAddElementSelected(object element, Type type)
        {
            if ((element as CheckBox).Checked)
            {
                TemplateEditMode = TemplateEditMode.AddElement;
                _templateAddType = type;
                foreach (var checkBox in flowLayoutPalletePanel.Controls)
                {
                    if (checkBox is CheckBox cb && cb != element)
                    {
                        cb.Checked = false;
                    }
                }
            }
            else
            {
                _templateAddType = null;
                TemplateEditMode = TemplateEditMode.EditElement;
            }
        }

        private void PImageRadioButton_CheckedChanged(object sender, EventArgs
e)
```

```
        {
            SetAddElementSelected(sender, typeof(ImageElement));
        }

        private void PTextRadioButton_CheckedChanged(object sender, EventArgs e)
        {
            SetAddElementSelected(sender, typeof(TextStaticElement));
        }

        private void PTextTemplateRadioButton_CheckedChanged(object sender,
EventArgs e)
        {
            SetAddElementSelected(sender, typeof(TextTemplateElement));
        }

        private void SwitchPropertiesFormVisible()
        {
            if (PropertiesForm.Visible)
            {
                PropertiesForm.Hide();
                propertiesToolStripMenuItem.Checked = false;
            }
            else
            {
                PropertiesForm.Show();
                propertiesToolStripMenuItem.Checked = true;
            }
        }

        private void propertiesToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            SwitchPropertiesFormVisible();
        }

        //public void NotifyHidden(Form form)
        //{
            //if (form == PropertiesForm)
            //{
            //    propertiesToolStripMenuItem.Checked = false;
            //}
        //}

        public void NotifySelected(ITemplateElement element)
        {
            foreach(var control in PdfSourcePanel.Controls)
            {
                if(control is ITemplateElement el)
                {
                    el.Selected = false;
                }
            }
            element.Selected = true;
            SelectedElement = element;
        }

        private void SwitchPalleteVisible()
        {
            if (palleteGroupBox.Visible)
            {
                palleteGroupBox.Hide();
                palleteToolStripMenuItem.Checked = false;
            }
```

```
            else
            {
                palleteGroupBox.Show();
                palleteToolStripMenuItem.Checked = true;
            }
        }
        private void palleteToolStripMenuItem_Click(object sender, EventArgs e)
        {
            SwitchPalleteVisible();
        }

        private void InterceptPanel_MouseDown(object sender, MouseEventArgs e)
        {
            if (_templateEditMode == TemplateEditMode.MoveElement)
            {
                InterceptPanel.SetDrawSquare(e.Location, SelectedElement.Size);
            }
            else
            if (_templateEditMode == TemplateEditMode.ResizeElement)
            {
                InterceptPanel.SetDrawSquare(SelectedElement.Location,
e.Location);
            }
        }


        private void InterceptPanel_MouseMove(object sender, MouseEventArgs e)
        {
            if (_templateEditMode == TemplateEditMode.MoveElement)
            {
                InterceptPanel.SetDrawSquare(e.Location, SelectedElement.Size);
            }
            else
            if (_templateEditMode == TemplateEditMode.ResizeElement)
            {
                InterceptPanel.SetDrawSquare(SelectedElement.Location,
e.Location);
            }
        }

        private void InterceptPanel_MouseLeave(object sender, EventArgs e)
        {
            StopInterceptMouseMove();
        }

        private void InterceptPanel_MouseUp(object sender, MouseEventArgs e)
        {
            StopInterceptMouseMove();
        }

        private void StopInterceptMouseMove()
        {
            if (_templateEditMode == TemplateEditMode.MoveElement)
            {
                SelectedElement.Location = InterceptPanel.ResultPosition;
            }
            else
            if (_templateEditMode == TemplateEditMode.ResizeElement)
            {
                SelectedElement.Size = InterceptPanel.ResultSize;
            }

            TemplateEditMode = TemplateEditMode.EditElement;
```

```
            InterceptPanel.StopDraw();
        }

        private void Element_LongPress(object sender, EventArgs e)
        {
            InterceptPanel.Invoke(new Action(() =>
            {
                SelectedElement = sender as ITemplateElement;
                TemplateEditMode = TemplateEditMode.MoveElement;
            }));
        }

        private void deleteToolStripMenuItem_Click(object sender, EventArgs e)
        {
            var element = SelectedElement as Control;
            if(element != null)
            {
                PdfSourcePanel.Controls.Remove(element);
            }
        }
        #endregion

        #region Template Part
        public PdfTemplateProperties TemplateProperties { get; set;}
        public Dictionary<string, string> TemplateKeyDatas { get; set; } = new
Dictionary<string, string>();

        private void newToolStripMenuItem_Click(object sender, EventArgs e)
        {
            InitializeEmpty();
        }

        private void closeToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Close();
        }
        private void saveToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Save();
        }

        #region Save
        private bool Save()
        {
            string name;
            string path;

            if (TemplateProperties == null || !TemplateProperties.IsSaved)
            {
                var templatesPath =
Path.Combine(Directory.GetCurrentDirectory(), Constants.TemplatesDirectory);
                if (!Directory.Exists(templatesPath))
Directory.CreateDirectory(templatesPath);

                using (SaveFileDialog saveFileDialog = new SaveFileDialog())
                {
                    saveFileDialog.Filter = "Template json files
(*.tmjson)|*.tmjson";
                    saveFileDialog.FileName = "defaultTemplate.tmjson";
                    saveFileDialog.FilterIndex = 0;
                    saveFileDialog.RestoreDirectory = true;
                    saveFileDialog.OverwritePrompt = true;
                    saveFileDialog.InitialDirectory = templatesPath;
```

```
                    if (saveFileDialog.ShowDialog() == DialogResult.OK)
                    {
                        name = Path.GetFileName(saveFileDialog.FileName);
                        path = Path.GetDirectoryName(saveFileDialog.FileName);
                    }
                    else return false;

                }

                TemplateProperties = TemplateLibrary.CreateNewTemplate(name);
                TemplateProperties.RootFolderPath = path;
                TemplateProperties.PageSize = new SizeF(210, 297);
            }
            else
            {
                name = TemplateProperties.Name;
                path = TemplateProperties.RootFolderPath;
            }

            if (TemplateProperties.PageSize == SizeF.Empty)
            {
                TemplateProperties.PageSize = new SizeF(210, 297);
            }

            try
            {
                PopulateTemplateElements(TemplateProperties);
                //UpdateImageElementPaths(TemplateProperties);
                SaveTemplateFile(TemplateProperties);
                UpdateImageElementPaths(TemplateProperties);
            }
            catch (Exception ex)
            {
                return false;
            }

            return true;
        }

        private void UpdateImageElementPaths(PdfTemplateProperties
templateObject)
        {
            var templateFolderPath = templateObject.RootFolderPath;
            //string destImageFolderPath = Path.Combine(templateFolderPath,
Constants.ImagesDirectory);

            foreach (var element in templateObject.Elements)
            {
                if (element is ITemplateImage el && el.Parent != null &&
el.Parent is ImageElement imEl)
                {
                    imEl.ImagePath = Path.Combine(templateFolderPath,
el.ImagePath);
                }
            }
        }

        private void PopulateTemplateElements(PdfTemplateProperties
templateProperties)
        {
            templateProperties.Elements.Clear();
            foreach (var control in PdfSourcePanel.Controls)
```

```
            {
                var templateItem = control as ITemplateElement;
                var data = templateItem.GetSaveData();
                data.PopulateFromElement(templateItem);

                templateProperties.Elements.Add(data);
            }
        }

        private void UpdateImages(PdfTemplateProperties templateObject, string
newTemplateDirectory)
        {
            //move stuff to templates directory

            var templateFolderPath = templateObject.RootFolderPath;
            //update image elements
            //should contain image paths to template`s Images directory
            List<string> imagePathsOld = templateObject.ImagesPathOld;

            List<string> imagePathsNew = new List<string>();
            foreach (var element in templateObject.Elements)
            {
                if (element is ITemplateImage el)
                {
                    imagePathsNew.Add(el.ImagePath);
                }
            }
            //delete old images
            var imagePathNotExist = imagePathsOld.Where(i =>
!imagePathsNew.Contains(i));
            foreach (var imagePath in imagePathNotExist)
            {
                if (File.Exists(imagePath)) File.Delete(imagePath);
            }

            //move images
            string destImageFolderPath = Path.Combine(newTemplateDirectory,
Constants.ImagesDirectory);
            if (!Directory.Exists(destImageFolderPath))
Directory.CreateDirectory(destImageFolderPath);

            imagePathsOld.Clear();
            foreach (var element in templateObject.Elements)
            {
                if (element is ITemplateImage el)
                {
                    var imagepath = el.ImagePath;
                    if (!Path.IsPathRooted(imagepath))
                        imagepath = Path.Combine(templateFolderPath, imagepath);
                    if (!string.IsNullOrEmpty(imagepath) &&
!imagepath.StartsWith(destImageFolderPath))
                    {
                        var destFilePath = Path.Combine(destImageFolderPath,
Guid.NewGuid() + Path.GetExtension(imagepath));

                        File.Copy(imagepath, destFilePath, true);

                        templateObject.ImagesPathOld.Add(destFilePath);
                        el.ImagePath = destFilePath.Remove(0,
newTemplateDirectory.Length).Trim('\\', '/');
                    }
                    else
                    if(imagepath.StartsWith(destImageFolderPath))
```

```
                    {
                        el.ImagePath = imagepath.Remove(0,
newTemplateDirectory.Length).Trim('\\', '/');
                    }
                }
            }
        }

        private void SaveTemplateFile(PdfTemplateProperties templateObject)
        {
            var templateFolderPath = templateObject.RootFolderPath;
            var templateDirectory =
Path.Combine(Directory.GetCurrentDirectory(), Constants.TemplatesDirectory);
            var templatesFile =
Path.ChangeExtension(Path.Combine(templateDirectory, TemplateProperties.Name),
".tmjson");

            if (!Directory.Exists(templateDirectory))
            {
                Directory.CreateDirectory(templateDirectory);
            }

            if (!File.Exists(templatesFile))
            {
                File.Create(templatesFile);
            }

            UpdateImages(templateObject, templateDirectory);

            var settings = new JsonSerializerSettings
            {
                TypeNameHandling = TypeNameHandling.All
            };
            string serializedData = JsonConvert.SerializeObject(templateObject,
settings);
            File.WriteAllText(templatesFile, serializedData,
System.Text.Encoding.Unicode);

            templateObject.RootFolderPath = templateDirectory;
            templateObject.IsSaved = true;
        }
        #endregion

        private void OpenFile(string file)
        {
            string data = File.ReadAllText(file, System.Text.Encoding.Unicode);

            var settings = new JsonSerializerSettings
            {
                TypeNameHandling = TypeNameHandling.All
            };

            var templateFolderPath = Path.GetDirectoryName(file);


            PdfTemplateProperties serializedData = null;
            try
            {
                serializedData =
JsonConvert.DeserializeObject<PdfTemplateProperties>(data, settings);
            }
            catch (Exception ex)
            {
```

```
                serializedData =
TemplateLibrary.CreateNewTemplate(Path.GetFileName(file));
            }


            serializedData.RootFolderPath = templateFolderPath;
            serializedData.IsSaved = true;

            PdfSourcePanel.Controls.Clear();
            serializedData.Elements.Reverse();
            foreach (var element in serializedData.Elements)
            {
                var control = element.GetRawElement();
                InitElement(control);

                element.PopulateToElement(control);
                //PdfSourcePanel.Controls.Add(control as Control);

                if (control is ImageElement el)
                {
                    el.ImagePath = Path.Combine(templateFolderPath,
el.ImagePath);

                    serializedData.ImagesPathOld.Add(el.ImagePath);
                    if (File.Exists(el.ImagePath))
                    {
                        using (Stream stream = File.OpenRead(el.ImagePath))
                        {
                            System.Drawing.Image image =
System.Drawing.Image.FromStream(stream);
                            el.Image = image;
                        }
                    }

                }
            }

            TemplateProperties = serializedData;
        }

        private void openToolStripMenuItem_Click(object sender, EventArgs e)
        {
            var templatesPath = Path.Combine(Directory.GetCurrentDirectory(),
Constants.TemplatesDirectory);
            if (!Directory.Exists(templatesPath))
Directory.CreateDirectory(templatesPath);

            string file;
            using (OpenFileDialog openFileDialog = new OpenFileDialog())
            {
                openFileDialog.Filter = "Template json files
(*.tmjson)|*.tmjson";
                openFileDialog.FilterIndex = 0;
                openFileDialog.RestoreDirectory = true;
                openFileDialog.InitialDirectory = templatesPath;

                if (openFileDialog.ShowDialog() == DialogResult.OK)
                {
                    file = openFileDialog.FileName;
                }
                else return;
            }
```

```
            OpenFile(file);
        }

        private void printToolStripMenuItem_Click(object sender, EventArgs e)
        {
            string file;
            using (SaveFileDialog saveFileDialog = new SaveFileDialog())
            {
                saveFileDialog.Filter = "Pdf files (*.pdf)|*.pdf";
                saveFileDialog.FilterIndex = 0;
                saveFileDialog.RestoreDirectory = true;
                saveFileDialog.OverwritePrompt = true;

                if (saveFileDialog.ShowDialog() == DialogResult.OK)
                {
                    file = saveFileDialog.FileName;
                }
                else return;
            }
            /*
            To set the size and position of elements in iTextSharp using
millimeters, you need to convert millimeters to points because iTextSharp uses
points as the measurement unit. The conversion factor is that 1 millimeter
equals 2.83465 points.
            */

            if (TemplateProperties == null)
            {
                TemplateProperties =
TemplateLibrary.CreateNewTemplate(string.Empty);
            }
            if (!TemplateProperties.IsSaved)
            {
                PopulateTemplateElements(TemplateProperties);
            }

            PdfUtils.PrintPdf(TemplateProperties, file, TemplateKeyDatas);
        }

        #endregion

        private void settingsToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if(TemplateProperties == null)
            {
                TemplateProperties =
TemplateLibrary.CreateNewTemplate(string.Empty);
            }

            TemplateSettingsForm settingsForm = new TemplateSettingsForm();
            settingsForm.SetSettingsData(TemplateProperties.Settings);

            settingsForm.ShowDialog();
        }

        private void TemplateEditForm_FormClosing(object sender,
FormClosingEventArgs e)
        {
            var message = "Do you want to save changes?";
            var title = "Save Changes";

            DialogResult result = MessageBox.Show(
                message,
```

```
                title,
                MessageBoxButtons.YesNoCancel,
                MessageBoxIcon.Question);

            switch (result)
            {
                case DialogResult.Yes:
                    if (!Save())
                    {
                        e.Cancel = true;
                        return;
                    }
                    break;
                case DialogResult.No:
                    break;
                case DialogResult.Cancel:
                    e.Cancel = true;
                    return;
            }

            PropertiesForm.SetClose = true;
            PropertiesForm.Close();

            //var dialogResult = MessageBox.Show("Save changes?", "Save",
MessageBoxButtons.YesNoCancel);

            //if(dialogResult == DialogResult.Yes)
            //{
            //    Save();
            //    DialogResult = DialogResult.OK;
            //}
            //else
            //if (dialogResult == DialogResult.Cancel)
            //{
            //    e.Cancel = true;
            //}
        }
    }
}
```

Модуль створення набору документів

```
using System;
using System.Windows.Forms;

namespace WFAdmissionDocuments
{
    partial class DocumentSetsListForm
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
```

```
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            System.Windows.Forms.TreeNode treeNode1 = new
System.Windows.Forms.TreeNode("Узел7");
            this.documentSetTreeView = new System.Windows.Forms.TreeView();
            this.buttonNewDocumentSet = new System.Windows.Forms.Button();
            this.buttonNewTemplatePage = new System.Windows.Forms.Button();
            this.buttonLoadPage = new System.Windows.Forms.Button();
            this.buttonRemove = new System.Windows.Forms.Button();
            this.buttonSettings = new System.Windows.Forms.Button();
            this.buttonEdit = new System.Windows.Forms.Button();
            this.buttonAddPage = new System.Windows.Forms.Button();
            this.buttonPrint = new System.Windows.Forms.Button();
            this.buttonUp = new System.Windows.Forms.Button();
            this.panelUpDown = new System.Windows.Forms.Panel();
            this.buttonDown = new System.Windows.Forms.Button();
            this.buttonReload = new System.Windows.Forms.Button();
            this.panelUpDown.SuspendLayout();
            this.SuspendLayout();
            //
            // documentSetTreeView
            //
            this.documentSetTreeView.Anchor =
((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Bottom)
            | System.Windows.Forms.AnchorStyles.Left)
            | System.Windows.Forms.AnchorStyles.Right)));
            this.documentSetTreeView.Location = new System.Drawing.Point(12,
12);
            this.documentSetTreeView.Name = "documentSetTreeView";
            treeNode1.Name = "Узел7";
            treeNode1.Text = "Узел7";
            this.documentSetTreeView.Nodes.AddRange(new
System.Windows.Forms.TreeNode[] {
            treeNode1});
            this.documentSetTreeView.Size = new System.Drawing.Size(468, 426);
            this.documentSetTreeView.TabIndex = 0;
            this.documentSetTreeView.AfterSelect += new
System.Windows.Forms.TreeViewEventHandler(this.documentSetTreeView_AfterSelect);
            //
            // buttonNewDocumentSet
            //
            this.buttonNewDocumentSet.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
            this.buttonNewDocumentSet.Location = new System.Drawing.Point(486,
12);
            this.buttonNewDocumentSet.Name = "buttonNewDocumentSet";
            this.buttonNewDocumentSet.Size = new System.Drawing.Size(129, 23);
            this.buttonNewDocumentSet.TabIndex = 1;
```

```
            this.buttonNewDocumentSet.Text = "New document set";
            this.buttonNewDocumentSet.UseVisualStyleBackColor = true;
            this.buttonNewDocumentSet.Click += new
System.EventHandler(this.buttonNewDocumentSet_Click);
            //
            // buttonNewTemplatePage
            //
            this.buttonNewTemplatePage.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
            this.buttonNewTemplatePage.Location = new System.Drawing.Point(486,
82);
            this.buttonNewTemplatePage.Name = "buttonNewTemplatePage";
            this.buttonNewTemplatePage.Size = new System.Drawing.Size(129, 23);
            this.buttonNewTemplatePage.TabIndex = 2;
            this.buttonNewTemplatePage.Text = "New page";
            this.buttonNewTemplatePage.UseVisualStyleBackColor = true;
            this.buttonNewTemplatePage.Click += new
System.EventHandler(this.buttonNewTemplatePage_Click);
            //
            // buttonLoadPage
            //
            this.buttonLoadPage.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
            this.buttonLoadPage.Location = new System.Drawing.Point(486, 111);
            this.buttonLoadPage.Name = "buttonLoadPage";
            this.buttonLoadPage.Size = new System.Drawing.Size(129, 23);
            this.buttonLoadPage.TabIndex = 3;
            this.buttonLoadPage.Text = "Load page";
            this.buttonLoadPage.UseVisualStyleBackColor = true;
            this.buttonLoadPage.Click += new
System.EventHandler(this.buttonLoadPage_Click);
            //
            // buttonRemove
            //
            this.buttonRemove.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
            this.buttonRemove.Location = new System.Drawing.Point(486, 179);
            this.buttonRemove.Name = "buttonRemove";
            this.buttonRemove.Size = new System.Drawing.Size(129, 23);
            this.buttonRemove.TabIndex = 5;
            this.buttonRemove.Text = "Remove";
            this.buttonRemove.UseVisualStyleBackColor = true;
            this.buttonRemove.Click += new
System.EventHandler(this.buttonRemove_Click);
            //
            // buttonSettings
            //
            this.buttonSettings.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
            this.buttonSettings.Location = new System.Drawing.Point(486, 208);
            this.buttonSettings.Name = "buttonSettings";
            this.buttonSettings.Size = new System.Drawing.Size(129, 23);
            this.buttonSettings.TabIndex = 6;
            this.buttonSettings.Text = "Settings";
            this.buttonSettings.UseVisualStyleBackColor = true;
            this.buttonSettings.Click += new
System.EventHandler(this.buttonSettings_Click);
            //
            // buttonEdit
```

```
            //
            this.buttonEdit.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
            this.buttonEdit.Location = new System.Drawing.Point(486, 150);
            this.buttonEdit.Name = "buttonEdit";
            this.buttonEdit.Size = new System.Drawing.Size(129, 23);
            this.buttonEdit.TabIndex = 7;
            this.buttonEdit.Text = "Edit";
            this.buttonEdit.UseVisualStyleBackColor = true;
            this.buttonEdit.Click += new
System.EventHandler(this.buttonEdit_Click);
            //
            // buttonAddPage
            //
            this.buttonAddPage.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
            this.buttonAddPage.Location = new System.Drawing.Point(486, 41);
            this.buttonAddPage.Name = "buttonAddPage";
            this.buttonAddPage.Size = new System.Drawing.Size(129, 23);
            this.buttonAddPage.TabIndex = 8;
            this.buttonAddPage.Text = "Add page(s)";
            this.buttonAddPage.UseVisualStyleBackColor = true;
            this.buttonAddPage.Click += new
System.EventHandler(this.buttonAddPage_Click);
            //
            // buttonPrint
            //
            this.buttonPrint.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
            this.buttonPrint.Location = new System.Drawing.Point(486, 237);
            this.buttonPrint.Name = "buttonPrint";
            this.buttonPrint.Size = new System.Drawing.Size(129, 23);
            this.buttonPrint.TabIndex = 9;
            this.buttonPrint.Text = "To pdf";
            this.buttonPrint.UseVisualStyleBackColor = true;
            this.buttonPrint.Click += new
System.EventHandler(this.buttonPrint_Click);
            //
            // buttonUp
            //
            this.buttonUp.AutoSizeMode =
System.Windows.Forms.AutoSizeMode.GrowAndShrink;
            this.buttonUp.Dock = System.Windows.Forms.DockStyle.Left;
            this.buttonUp.Font = new System.Drawing.Font("Microsoft Sans Serif",
15.75F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(204)));
            this.buttonUp.Location = new System.Drawing.Point(0, 0);
            this.buttonUp.Name = "buttonUp";
            this.buttonUp.Size = new System.Drawing.Size(59, 31);
            this.buttonUp.TabIndex = 10;
            this.buttonUp.Text = "⇧";
            this.buttonUp.UseVisualStyleBackColor = true;
            this.buttonUp.Click += new System.EventHandler(this.buttonUp_Click);
            //
            // panelUpDown
            //
            this.panelUpDown.Controls.Add(this.buttonDown);
            this.panelUpDown.Controls.Add(this.buttonUp);
            this.panelUpDown.Location = new System.Drawing.Point(486, 266);
```

```
            this.panelUpDown.Name = "panelUpDown";
            this.panelUpDown.Size = new System.Drawing.Size(129, 31);
            this.panelUpDown.TabIndex = 11;
            this.panelUpDown.Visible = false;
            //
            // buttonDown
            //
            this.buttonDown.AutoSizeMode =
System.Windows.Forms.AutoSizeMode.GrowAndShrink;
            this.buttonDown.Dock = System.Windows.Forms.DockStyle.Right;
            this.buttonDown.Font = new System.Drawing.Font("Microsoft Sans
Serif", 15.75F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.buttonDown.Location = new System.Drawing.Point(70, 0);
            this.buttonDown.Name = "buttonDown";
            this.buttonDown.Size = new System.Drawing.Size(59, 31);
            this.buttonDown.TabIndex = 11;
            this.buttonDown.Text = "⇩";
            this.buttonDown.UseVisualStyleBackColor = true;
            this.buttonDown.Click += new
System.EventHandler(this.buttonDown_Click);
            //
            // buttonReload
            //
            this.buttonReload.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
            this.buttonReload.Location = new System.Drawing.Point(486, 415);
            this.buttonReload.Name = "buttonReload";
            this.buttonReload.Size = new System.Drawing.Size(129, 23);
            this.buttonReload.TabIndex = 12;
            this.buttonReload.Text = "Reload";
            this.buttonReload.UseVisualStyleBackColor = true;
            this.buttonReload.Click += new
System.EventHandler(this.buttonReload_Click);
            //
            // DocumentSetsListForm
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(627, 450);
            this.Controls.Add(this.buttonReload);
            this.Controls.Add(this.panelUpDown);
            this.Controls.Add(this.buttonPrint);
            this.Controls.Add(this.buttonAddPage);
            this.Controls.Add(this.buttonEdit);
            this.Controls.Add(this.buttonSettings);
            this.Controls.Add(this.buttonRemove);
            this.Controls.Add(this.buttonLoadPage);
            this.Controls.Add(this.buttonNewTemplatePage);
            this.Controls.Add(this.buttonNewDocumentSet);
            this.Controls.Add(this.documentSetTreeView);
            this.MinimumSize = new System.Drawing.Size(643, 488);
            this.Name = "DocumentSetsListForm";
            this.Text = "DocumentSetsListForm";
            this.Load += new
System.EventHandler(this.DocumentSetsListForm_Load);
            this.panelUpDown.ResumeLayout(false);
            this.ResumeLayout(false);

        }
```

```
        #endregion

        private System.Windows.Forms.TreeView documentSetTreeView;
        private System.Windows.Forms.Button buttonNewDocumentSet;
        private System.Windows.Forms.Button buttonNewTemplatePage;
        private System.Windows.Forms.Button buttonLoadPage;
        private System.Windows.Forms.Button buttonRemove;
        private System.Windows.Forms.Button buttonSettings;
        private System.Windows.Forms.Button buttonEdit;
        private System.Windows.Forms.Button buttonAddPage;
        private System.Windows.Forms.Button buttonPrint;
        private Button buttonUp;
        private Panel panelUpDown;
        private Button buttonDown;
        private Button buttonReload;
    }
}
```

Модуль введення та зберігання даних

```
partial class DataEnterTemplateForm
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(800, 450);
            this.Text = "DataEnterTemplateForm";
        }

        #endregion
    }


namespace WFAdmissionDocuments
{
```

```
    partial class DataEnterDocumentSetForm
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(800, 450);
            this.Text = "DataEnterDocumentSetForm";
        }

        #endregion
    }
}
```

UNICHECK
by Turnitin

Ім'я користувача:
Кафедра КІ

ID перевірки:
1016376061

Дата перевірки:
19.06.2024 18:49:42 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
19.06.2024 18:50:38 EEST

ID користувача:
100005591

Назва документа: Погорєлов_Інформаційна система "Кабінет відповідального працівника приймальної комі...

Кількість сторінок: 63 **Кількість слів:** 9669 **Кількість символів:** 79396 **Розмір файлу:** 5.31 MB **ID файлу:** 1016184081

3.48%
Схожість

Найбільша схожість: 0.75% з Інтернет-джерелом (http://elar.khmnu.edu.ua/jspui/bitstream/123456789/13851/1/%d0%94.

3.16% Джерела з Інтернету 210 ..... Сторінка 65

1.46% Джерела з Бібліотеки 140 ..... Сторінка 66

0.15% Цитат

Цитати 1 ..... Сторінка 67

Посилання 1 ..... Сторінка 67

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи 1

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 5%**

	Документ		Сумарний збіг по Базі Даних	
ID: 131623 Назва: БКР Інформаційна система “Кабінет відповідального працівника приймальної Додано в БД: 2024-06-19 Автора: Д. В. Погорєлов Керівники: Т. М. Кисіль Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	69249	548	1183 (2%)	16 (3%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Погорєлов Дмитро Вікторович

Тема: Інформаційна система "Кабінет відповідального працівника_ приймальної комісії"

Спеціальність: 126 «Інформаційні системи та технології»

Обсяг кваліфікаційної роботи:

Кількість листів креслень ___3___ Кількість сторінок записки ___82___

1. Короткий зміст роботи та прийнятих рішень: Метою роботи є розробка програмного продукту, який зможе спростити роботу приймальної комісії, автоматизувати процес заповнення документації, як то договір про навчання та договір про надання платної освітньої послугу, тощо

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: В першому розділі кваліфікаційної роботи проведено дослідження предметної області (проаналізовані існуючі системи електронного документообігу та автоматичного заповнення документів як іноземні так і вітчизняні) та виконано постановку задачі дослідження. В другому розділі кваліфікаційної роботи проведено моделювання інформаційної системи "Кабінет відповідального працівника приймальної комісії"; побудовано ряд UML-діаграм, які описують функціонал даної системи та динамічні аспекти системи; наведено діаграми класів даної системи. В третьому розділі кваліфікаційної роботи виконано реалізацію даної інформаційної системи, виконано її функцій не тестування та тестування інтерфейсу.

4. Позитивні сторони роботи: висока практична цінність роботи.

5. Негативні сторони роботи: недостатня увага приділена опису інструкції користувача

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: В загальному робота виконана на достатньому рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: добре/С

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) _____

Федуна Микола Васильович, к. т. н.,
доц. кафедри АКІТаР, ХНУ

"___"_____2024 р. _____(підпис)

Завідувачу кафедри КІІС
д-р.техн.наук, проф. Говорущенко Т. О.

Погорєлова Дмитра Вікторовича
ПІБ здобувача вищої освіти

ФІТ, 4 курсу, групи ІСТ-20-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Інформаційна система "Кабінет відповідального працівника приймальної комісії"

Автор: Погорєлов Дмитро Вікторович

Спеціальність: 126– Інформаційні системи та технології

Освітня програма: освітньо-професійна

Науковий керівник: Кисіль Тетяна Миколаївна, к.ф.-м.н, доцент

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
3) окремі виявлені збіги є загальновживаними фразами, про що свідчить посилання системи на збіг з 10-40 джерелами на один фрагмент речення.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 3,48% і адресується до 350 першоджерела, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ Т. М. Кисіль

Гарант ОП _____ Є. Г. Гнатчук

Завідувач кафедри КІІС _____ Т. О. Говорущенко