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

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Інформаційна веб система керування документами на підприємстві
Назва теми

КвРІСТ. 200199.20.11.03 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 III курсу, група ІСТс20–1 _____ К.Г. Сворінь
Підпис Ініціали, прізвище

Керівник _____ С.М. Лисенко
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

«1» червня 2023 р.

Хмельницький 2023

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 ОСВІТНЯ ПРОГРАМА «ІНФОРМАЦІЙНІ СИСТЕМИ ТА ТЕХНОЛОГІЇ»

ЗАТВЕРДЖУЮ
Зав. кафедри Т.О.Говорущенко
" 11 " 01 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Сворінь Кирил Геннадійівич
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Інформаційна веб-система керування документами на підприємстві
Керівник проекту (роботи) Лисенко С.М., д.т.н., проф.
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 1.03.2023 р. № 5
2. Строк подання студентом проекту (роботи) на кафедру 01.06.2023 р.
3. Вихідні дані до проекту (роботи) Завдання на дипломне проектування
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Дослідження предметної області та постановка задачі
Проєктування програмно-технічного засобу
Програмна реалізація та тестування інформаційної веб-системи керування документами
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Інтерфейсні вікна програмного забезпечення
UML-діаграми інформаційної системи
UMLдіаграми.інформаційної системи(продовження)

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 «__01_»_____03_____2023 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	20.02.2022	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.03.2023	виконано
3	Робота над розділом 1 – дослідження предметної області та постановка задачі	10.03.2023	виконано
4	Робота над розділом 2 – Проєктування програмно-технічного засобу	20.04.2023	виконано
5	Робота над розділом 3 – Програмна реалізація та тестування інформаційної веб-системи керування документами	30.04.2023	виконано
6	Оформлення пояснювальної записки згідно вимог	11.05.2023	виконано
7	Попередній захист ВКР	26.05.2023	виконано
8	Захист ВКР на засіданні ЕК	Червень 2023 року	

Студент _____ Підпис Сворінь К.Г. Ініціали, прізвище

Керівник проекту (роботи) _____ Підпис Лисенко С.М. Ініціали, прізвище

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРІСТ. 200199.20.11.03 ПЗ	Пояснювальна записка	58		
			Графічні матеріали			
2		КвРІСТ. 200199.20.11.03 Е8	Інтерфейсні вікна програмного	1		
			забезпечення			
				1		
3		КвРІСТ. 200199.20.11.03 Е8	UML діаграми			
4		КвРІСТ. 200199.20.11.03 Е8	UML діаграми	1		
5		КвРІСТ. 200199.20.11.03 Е8	Код програмного забезпечення	13		
			інформаційної веб- системи			
			документами на підприємстві			

Зм	Арк	№ докум	Підпис	Дата	КвРІСТ. 200199.20.11.03 ПЗ		
Розробив		Сворінь К.Г.			Відомість проекту	Літера	Аркуш 1 / Аркушів 1
Перевір.		Лисенко С.М.				У	
Н. контр.		Лисенко С.М.				ХНУ, ІСТс-20-1	
Затв.		Говорущенко		31.05			

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Інформаційна веб система керування документами на підприємстві».

Автор роботи: Сворінь Кирил Геннадійович.

Керівник роботи: Лисенко Сергій Миколайович.

Пояснювальна записка: 58 с., 12 рис., 4 дод., 2 таб., 80 джерел.

Графічна частина: 3 креслення.

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ, UML ДІАГРАМИ, ФУНКЦІЙНІ ТА НЕФУНКЦІЙНІ ВИМОГИ, ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ,.

Метою роботи є розробка інформаційна система .

У цій роботі розроблена веб-системи керування документами зокрема розроблено UML-діаграми. Здійснено програмну реалізацію та розроблено інтерфейс користувача. Система була реалізована за допомогою інструменту NetBeans, який є програмним забезпеченням з відкритим кодом і інтегрованим середовищем розробки (IDE).

Підпис студента  Дата 30.05.2023

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ **4**

ВСТУП **5**

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ .. **8**

1.1 Аналіз задачі, обґрунтування вибору моделі життєвого циклу для реалізації проекту 8

1.2 Аналіз наявного програмно-апаратного забезпечення предметної області 13

1.3 Визначення вимог до системи автоматизації та розробка технічного завдання 15

1.4 Висновки 19

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ ВЕБ-СИСТЕМИ **21**

2.1 Розробка користувальницьких вимог 21

2.1.1 Механізм іменування файлів 21

2.1.3 Механізм отримання документів 22

2.1.4 Забезпечення контролю версій документів 23

2.1.5 Процес керування документами 24

2.2.1 Застосування методології Agile 25

2.2.2 Водоспадна модель 26

2.2.3 Модифікований водоспад 26

2.2.4 Забезпечення процедури збору даних 27

2.2.5 Забезпечення процедури перегляду результатів активності документообігу співробітниками 28

2.3 Нефункційні вимоги до інформаційної системи 31

2.4 Аналіз вимого до інформаційної системи 31

2.5 Проектування інформаційної веб-системи керування документами 34

2.6 Висновки 38

Зм.	Арк.	№докум.	Підпис	Дата	КвРІСТ. 200199.20.11.03 ПЗ		
Виконав		Сворінь К.Г.			Інформаційна веб-система керування документами на підприємстві	Літера	Аркуш 2 / Аркушів 58
Перевір.		Лисенко С.М.				У	
Н.контр.		Лисенко С.М.				ХНУ, ІСТс-20-1	
Затвер.		Говорущенко Т.О.		31.05			

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ ВЕБ-СИСТЕМИ КЕРУВАННЯ ДОКУМЕНТАМИ 39

3.1 Розроблення UML-діаграми для інформаційної веб-системи керування документами 39

3.1 Програмна реалізація інформаційної веб-системи керування документами 48

3.1 Інтерфейс користувача інформаційної веб-системи керування документами 50

3.2 Методи тестування ІС 59

3.6.1 Модульне тестування 60

3.6.2 Функційний тест 60

3.6.4 Тест безпеки 61

3.6.5 Тест юзабіліті ІС 61

3.7 Висновки 61

ВИСНОВКИ 62

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ 63

ДОДАТОК А Інтерфейсні вікна програмного забезпечення **74**

ДОДАТОК Б UML діаграми **75**

ДОДАТОК В UML діаграми **76**

ДОДАТОК Г Код програмного забезпечення інформаційної веб-системи керування документами на підприємстві **77**

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ІВСКД - Інформаційна веб-система керування документами

БД - база даних

ОС - операційна система

ПЗ - програмне забезпечення

ВСТУП

Системи діловодства або управління документами беруть свій початок на початку 1970-х років, коли комп'ютери почали використовувати на підприємствах як пристрої для запису та архівування або зберігання паперів [1]. Однак зі збільшенням доступності комп'ютерних технологій у західному світі історія управління документами зробила різкий поворот у 1980-х роках [2].

Оскільки значення та використання документів з часом зростало, управління документами описують як організацію та ведення паперової документації щодо певних завдань та процедур. Розвиток організації сьогодні значною мірою залежить від управління документами. Дуже важливо ефективно і безпечно розподіляти правильні папери між відповідними особами. За допомогою документообігу компанії можуть впорядкувати всю свою інформацію в усіх її формах в одному місці. Незалежно від розміру чи галузі, впорядкування бізнес-процесів та підвищення ефективності є ключовими питаннями для всіх організацій.

Система управління документами (DMS) - це використання комп'ютерних систем або програмного забезпечення для зберігання, управління та відстеження електронних документів та електронних образів паперової інформації. Крім того, його можна визначити як швидку і просту у використанні систему, яку менеджери та інші співробітники можуть використовувати для економії часу і зусиль при обробці, зберіганні, пошуку, координації оновлень, вилучення та обміну інформацією між собою [3,4].

Враховуючи той факт, що система управління документами підвищує операційну ефективність і продуктивність, система електронного документообігу (СЕД) була прийнята багатьма компаніями і секторами для підвищення продуктивності праці та ефективності транзакцій.

Управління документами пронизало всі операції в державному та приватному секторах. Це пов'язано з тим, що різні документи використовуються в повсякденних ділових операціях банків, будівельної та виробничої галузей, системи правосуддя, транспорту та логістики, страхування, освіти тощо. Урядові відомства та організації у своїй роботі також покладаються на інформацію, що міститься в різних документах. Крім того, для того, щоб більшість фірм продовжували залишатися конкурентоспроможними та операційно успішними й ефективними, автоматизація перетворилася на ядро стратегічного планування в більшості програм модернізації [5, 6, 7, 8].

Ці автоматизовані системи, з одного боку, є життєво важливими для полегшення процесів обміну інформацією між рівнями управління, операціями та зацікавленими сторонами, а з іншого боку, вони підвищують ефективність, життєздатність, видимість та конкурентні переваги організації.

Більшість країн, що розвиваються, все ще використовують традиційні системи управління паперовими документами (DMS), але також зростає обсяг електронного контенту, який зберігається на робочих станціях і серверах, наприклад, електронні листи, веб-сторінки та пакети баз даних, які зберігаються на робочих станціях користувачів і серверах.

Одним з найважливіших інструментів управління є веб-система електронного документообігу (СЕД), яка часто використовується для інтегрованого збору даних в установі або організації.

Метою кваліфікаційної роботи є забезпечення керування документами на підприємстві .

Поставлена мета досягається розв’язанням такої основної задачі: проєктування та розроблення веб-система керування документами на підприємстві, яка передбачатиме застосування різних сценаріїв передачі документів.

Об’єктом дослідження є процес керування документами на підприємстві.

Предметом дослідження є веб-система керування документами на підприємстві.

Для досягнення поставленої мети застосовуються різноманітні дослідницькі методи, такі як методи синтезу, аналізу й моделювання процесів, а також засади системного аналізу й теоретико-множинні підходи

Практичне значення має спроєктована підсистема керування документами на підприємстві яка передбачає застосування різних сценаріїв створення\передачі документів.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз задачі, обґрунтування вибору моделі життєвого циклу для реалізації проекту

Структура інформаційної веб-системи керування документами на підприємстві складається з трьох основних компонентів: серверної частини, клієнтської частини та бази даних [1-3].

Серверна частина відповідає за обробку запитів, збереження та організацію документів, контроль доступу до них, а також автоматизацію процесів.

Клієнтська частина надає користувачам можливість працювати з системою через веб-інтерфейс. База даних забезпечує збереження та організацію документів.

Функціональні особливості інформаційної веб-системи керування документами на підприємстві включають наступні функції:

Збереження та організація документів. Система дозволяє зберігати та організовувати документи у зручному для користувачів форматі.

Кожен документ може мати свої метадані (наприклад, назва, автор, дата створення тощо), що дозволяє швидко знаходити та редагувати потрібні документи [4-6].

Керування доступом. Інформаційна веб-система керування документами забезпечує контроль доступу до документів, що дозволяє обмежувати доступ до конфіденційної інформації.

Керування доступом реалізується через різні рівні доступу, ролі користувачів та групи доступу.

Автоматизація процесів. Інформаційна веб-система керування документами дозволяє автоматизувати процеси створення, редагування та підпису документів. Наприклад, можливість автоматичного створення звітів або нагадувань про терміни підпису документів [7-9].

Пошук документів. Система дозволяє швидкий та ефективний пошук документів за різними параметрами, такими як назва документа, тип, дата, автор та інші метадані. Пошук може бути здійснений як вручну, так і за допомогою автоматичних фільтрів.

Колаборація та співпраця. Інформаційна веб-система керування документами надає можливість спільно працювати з документами, в тому числі в режимі онлайн [10-13].

Користувачі можуть спільно редагувати документи, вносити коментарі та пропозиції, спілкуватись у чаті та інших режимах спілкування.

Аналітика та звітність. Інформаційна веб-система керування документами дозволяє аналізувати діяльність підприємства з використанням документів.

Наприклад, можливість побудови звітів за кількістю створених та підписаних документів за певний період часу або за різними відділами підприємства [5-9].

Пошук документів може бути здійснений у системі інформаційного керування документами швидко та ефективно за допомогою різних параметрів, включаючи назву документа, тип, дату, автора та інші метадані. Пошук можна виконувати як вручну, так і за допомогою автоматичних фільтрів.

Користувачі системи можуть спільно працювати з документами, включаючи можливість спільного редагування, внесення коментарів та пропозицій, спілкування у чаті та інших режимах спілкування.

Система інформаційного керування документами також надає можливість аналізувати діяльність підприємства [6-9].

Загалом, інформаційна веб-система керування документами на підприємстві допомагає забезпечувати ефективне та безпечне керування документами, що є важливим елементом успішної діяльності будь-якої організації [11-14].

Деякі додаткові структурні та функціональні особливості інформаційної веб-системи керування документами можуть включати:

Ролі та права доступу. Інформаційна веб-система керування документами може мати різні рівні доступу для користувачів залежно від їхньої ролі в організації та потреб. Наприклад, деякі користувачі можуть мати право на створення та редагування документів, тоді як інші - тільки на перегляд.

Інтеграція з іншими системами. Інформаційна веб-система керування документами може бути інтегрована з іншими системами, такими як електронна пошта, система електронного документообігу або система управління різними проектами [20-22].

Це дозволяє підприємству забезпечити ще більшу ефективність та автоматизацію діяльності.

Захист інформації та даних є важливою складовою інформаційної веб-системи керування документами на підприємстві [44-47].

Для цього можуть застосовуватись різноманітні методи, такі як шифрування, аутентифікація, авторизація та інші.

Крім того, система може забезпечувати відстеження історії змін документів, що дозволяє користувачам перевірити, хто та коли вніс зміни.

Це допомагає підприємствам відповідати правилам та регуляторним вимогам [76-79].

В цілому, інформаційна веб-система керування документами на підприємстві є важливим і ефективним інструментом, який дозволяє підприємствам забезпечувати ефективну організацію документообігу та збереження інформації.

Завдяки цій системі, підприємство може отримати описані нижче вагомі переваги [60-62].

Ефективність та швидкість обробки документів. Інформаційна веб-система керування документами дозволяє швидко створювати, редагувати та розповсюджувати документи в різних форматах, що забезпечує ефективну роботу з документами та зменшує час, потрібний для їх обробки.

Оптимізація документообігу. Інформаційна веб-система керування документами дозволяє забезпечити оптимальний документообіг в організації, що допомагає знизити кількість помилок та зменшити витрати на паперову документацію та її обробку [55-57].

Забезпечення безпеки та конфіденційності інформації. Інформаційна веб-система керування документами дозволяє забезпечити високий рівень захисту конфіденційної інформації та персональних даних, що є важливим для підприємств у різних сферах діяльності.

Забезпечення зручного доступу до документів. Інформаційна веб-система керування документами дозволяє забезпечити зручний та швидкий доступ до документів, що дозволяє користувачам швидко знаходити потрібну інформацію та зменшує час на пошук необхідних документів [33-35].

У підсумку, інформаційна веб-система керування документами є важливим інструментом для підприємств у будь-якій сфері діяльності, я дозволяє забезпечити ефективний документообіг, зменшити час та витрати на обробку документів, підвищити безпеку та конфіденційність інформації та забезпечити зручний доступ до необхідних документів.

Основні структурні складові інформаційної веб-системи керування документами на підприємстві можуть включати:

Централізовану базу даних. Це може бути база даних на локальному сервері або в хмарі. База даних зберігає всі документи та інформацію про них, що дозволяє користувачам зручно знаходити та редагувати документи [61-65].

Інтерфейс користувача. Для зручного та ефективного використання інформаційної веб-системи керування документами на підприємстві, важливо мати простий та легкий у використанні інтерфейс, що дозволяє користувачам швидко знаходити необхідну інформацію та забезпечує зручний доступ до функцій системи.[8-19]

Функціонал для зберігання та обробки документів. Веб-система повинна надавати можливість створення, редагування, зберігання, перегляду та розповсюдження документів у різних форматах, включаючи текстові документи, електронні таблиці, фотографії та інші.

Функціонал для керування доступом до документів. Важливо мати можливість керувати доступом до документів, залежно від ролі користувача, наприклад, керівники мають доступ до всіх документів, тоді як звичайні співробітники мають доступ тільки до своїх документів [4-8].

Функціонал для контролю версій документів. Система повинна мати можливість зберігати кілька версій одного документу та забезпечувати можливість перегляду та відновлення попередніх верс й документу.

Функціонал для автоматизованого створення документів. Система може мати можливість створювати документи автоматично за шаблонами з введенням необхідної інформації, що дозволяє зменшити час на створення документів та запобігти помилкам в їх створенні.

Функціонал для моніторингу та контролю за документообігом. Система може мати можливість відстежувати статус документу в процесі його обробки, надавати інформацію про терміни та відповідальних осіб, що дозволяє контролювати та планувати роботу з документами [29-31].

Функціонал для інтеграції з іншими системами. Інформаційна веб-система керування документами може мати можливість інтегруватися з іншими системами, такими як електронна пошта, CRM-системи або ERP-системи, що

дозволяє забезпечити єдиний простір для роботи з документами та зберігати інформацію в єдиному форматі.

Функціонал для забезпечення безпеки та конфіденційності. Система повинна мати можливість забезпечувати безпеку та конфіденційність інформації, захищаючи її від несанкціонованого доступу, зловживань та випадкової втрати даних. [10-54]

Враховуючи вищезазначені функціональні та структурні особливості, інформаційна веб-система керування документами на підприємстві дозволяє забезпечити ефективний документообіг, зменшити час та витрати на обробку документів, підвищити безпеку та конфіденційність інформації та забезпечити зручний доступ до необхідних документів.

1.2 Аналіз наявного програмно-апаратного забезпечення предметної області

На сьогоднішній день багато підприємств використовують веб-системи керування документами, щоб покращити ефективність та продуктивність своєї діяльності.

Важливо мати належне програмне та апаратне забезпечення, щоб забезпечити швидкий та безперебійний доступ до даних, а також збереження важливих документів.

Один з основних елементів програмно-апаратного забезпечення веб-системи керування документами - це база даних, де зберігаються всі наявні документи [36-39].

Важливо мати належну базу даних, яка може забезпечити безперебійний доступ до даних і має достатню місткість для збереження документів.

Для цього можуть використовуватися різноманітні бази даних, такі як MySQL, PostgreSQL, Oracle, або Microsoft SQL Server.

Також необхідно мати належні засоби зберігання даних.

Найчастіше використовуються жорсткі диски або сервери з можливістю RAID-масивів, що забезпечує надійне зберігання даних та захист від втрати.

Що стосується програмного забезпечення, то веб-система керування документами повинна мати відповідні програмні модулі для забезпечення її ефективної роботи [75-77].

Це може включати модулі для роботи з базою даних, модулі для відображення документів та їхнього збереження в різних форматах, модулі для роботи з користувачами та їхніми ролями, а також модулі для забезпечення безпеки та захисту від несанкціонованого доступу.

При виборі програмного та апаратного забезпечення для веб-системи керування документами на підприємстві необхідно зосередитися на його відповідності потребам компанії [43-49].

До того ж, важливо враховувати вартість програмного та апаратного забезпечення та його сумісність з існуючими системами на підприємстві.

Наприклад, якщо підприємство використовує операційні системи з певною підтримкою, то необхідно знайти програмне забезпечення, яке підтримує ці операційні системи.

Крім того, важливо забезпечити безперебійну роботу системи та захистити її від можливих кібератак [57-60].

Для цього можуть використовуватися різні засоби захисту, такі як файрволи, антивіруси, системи виявлення вторгнень та інші. [11-16]

В цілому, для успішної роботи веб-системи керування документами на підприємстві важливо мати належне програмне та апаратне забезпечення, що забезпечує швидкий та безперебійний доступ до даних, збереження важливих документів та захист від можливих кібератак.

Також важливо звернути увагу на вартість та сумісність програмного та апаратного забезпечення з наявними системами на підприємстві.

Інтеграція з іншими системами є одним з ключових елементів успішної роботи веб-системи керування документами на підприємстві.

Наприклад, можна поєднати її з системою електронного документообігу, щоб автоматизувати процес обробки документів, або з системою управління проектами, щоб зберігати документи відповідно до проектів.

Також важливо мати можливість обмеження доступу до документів для певних користувачів згідно з їхніми ролями та правами доступу [74-77].

Це забезпечить безпеку та конфіденційність даних та допоможе уникнути небажаних помилок при обробці документів.

Нарешті, необхідно мати можливість зробити резервне копіювання даних та їх відновлення у разі втрати або пошкодження.

Це збереже важливу інформацію та допоможе уникнути можливих проблем у майбутньому.

Взагалі, веб-система керування документами на підприємстві повинна бути не тільки швидкою та безперебійною, але також інтегрованою з іншими системами [13-15].

1.3 Визначення вимог до системи автоматизації та розробка технічного завдання

При розробці інформаційної веб-системи керування документами на підприємстві потрібно враховувати ряд вимог, що забезпечать її ефективну роботу та задоволення потреб користувачів:

1. Функціональність. Система повинна забезпечувати всі необхідні функції для ефективного керування документами, зокрема зберігання, пошук, редагування, перегляд, відправлення та контроль над документами.

2. Зручність інтерфейсу. Система повинна мати зручний та інтуїтивно зрозумілий інтерфейс, що дозволяє користувачам швидко знаходити необхідну інформацію та працювати з документами.

3. Масштабованість. Система повинна бути здатною працювати з великою кількістю документів та користувачів, що дозволяє їй відповідати потребам розвиваючогося підприємства.

4. Надійність. Система повинна бути стійкою до витоків даних, злому безпеки та втрати даних, що дозволяє забезпечити безпеку та конфіденційність інформації.

5. Швидкість. Система повинна працювати швидко та ефективно, що дозволяє користувачам швидко знаходити та обробляти необхідну інформацію.

6. Гнучкість. Система повинна бути гнучкою та здатною пристосовуватися до потреб користувачів та відповідати на змінні вимоги підприємства [38-42].

Технічне завдання для розробки інформаційної веб-системи керування документами на підприємстві повинно містити такі пункти:

1. Опис функціональності системи, включаючи всі необхідні функції

2. Вимоги до архітектури системи, включаючи опис бази даних, серверів та програмного забезпечення, що використовуються.

3. Опис інтерфейсу користувача, включаючи дизайн та функціональність.

4. Вимоги до безпеки системи, включаючи захист від несанкціонованого доступу та від витоків даних.

5. Вимоги до масштабованості системи, включаючи здатність працювати з великим обсягом документів та користувачів.

6. Вимоги до швидкодії системи, включаючи мінімальні часи відповіді та завантаження сторінок.

7. Вимоги до технічної підтримки системи, включаючи моніторинг та резервне копіювання даних.

8. Вимоги до документації та підтримки користувачів, включаючи навчання та підтримку у разі проблем з системою.

9. Вимоги до термінів розробки та впровадження системи.

Розробка технічного завдання є важливим етапом при розробці інформаційної веб-системи керування документами на підприємстві, що дозволяє визначити всі необхідні вимоги та функціональність системи та забезпечити її ефективну та надійну роботу. [14-22]

Розробка технічного завдання є ключовим етапом при створенні веб-системи керування документами на підприємстві, оскільки воно допомагає визначити всі потреби та функціональні вимоги до системи і забезпечити її ефективну та надійну роботу.

При розробці технічного завдання необхідно враховувати усі специфічні особливості діяльності підприємства та вимоги користувачів.

Дотримання всіх вимог технічного завдання дозволить створити ефективну та надійну веб-систему керування документами на підприємстві, яка допоможе покращити продуктивність та забезпечити захист конфіденційної інформації.

Після розробки технічного завдання наступним важливим етапом є створення самої веб-системи керування документами на підприємстві. Під час розробки необхідно дотримуватися вимог технічного завдання та використовувати сучасні методики та технології розробки програмного забезпечення.

Перед початком розробки системи необхідно спочатку спроектувати її архітектуру та визначити структуру бази даних та розробити інтерфейс користувача.

При проектуванні архітектури системи важливо врахувати потреби та масштаби підприємства, щоб забезпечити ефективну та масштабовану роботу системи [29-33].

Розробка бази даних передбачає створення таблиць, які містять необхідну інформацію про документи та їх статуси. При розробці інтерфейсу користувача необхідно забезпечити зручний та інтуїтивно зрозумілий інтерфейс, щоб користувачі могли швидко та легко знаходити необхідну інформацію.

Розробка програмного забезпечення - це створення функціоналу системи, який включає можливість створення, редагування та пошуку документів, надання доступу до них відповідно до рівня доступу користувачів та інші функції.

Після створення системи необхідно перевірити її працездатність та відповідність технічним вимогам шляхом проведення тестування та заданої валідації [58-60].

Крім того, важливо забезпечити моніторинг та звітування про роботу системи, включаючи збір та аналіз статистичних даних, а також забезпечити підтримку та оновлення системи для забезпечення її безперебійної та ефективної роботи на протязі усього періоду її використання на підприємстві.

Додатково, важливим етапом є навчання користувачів системи, яке можна здійснити в режимі онлайн або офлайн, та забезпечити документацію та підтримку для користувачів.

Таким чином, розробка інформаційної веб-системи керування документами на підприємстві передбачає створення архітектури, бази даних та інтерфейсу користувача, розробку програмного забезпечення, проведення тестування та валідації системи, забезпечення підтримки та оновлення системи, навчання користувачів та підтримки їх роботи з системою [77-80].

1.4 Висновки

Аналіз відомих інформаційних веб-систем керування документами показав їх наявні переваги та недоліки.

Зокрема було зроблено висновок, що інформаційної системи керування документами можуть різнитися залежно від конкретних потреб та контексту організації. Однак, основні вимоги до такої системи можуть включати наступні аспекти.

Існують важливі функціональні вимоги:

- можливість створення;

- редагування та видалення документів;

- збереження та організація документів у структурованій формі, наприклад, за категоріями, тегами або датами, пошук та фільтрація документів за різними параметрами, такими як заголовок, ключові слова або автор, забезпечення доступу до документів з різних пристроїв та місць розташування;

- можливість керувати правами доступу до документів залежно від ролі користувача (наприклад, адміністратор, редактор, читач);

- автоматичне створення резервних копій та забезпечення захисту від втрати даних;

- захист конфіденційної інформації шляхом обмеження доступу до документів, контроль доступу до системи шляхом аутентифікації користувачів та надання їм відповідних прав доступу, захист від несанкціонованого доступу, злому або крадіжки даних;

- можливість інтеграції з іншими системами, наприклад, електронною поштою, системою управління взаємодією з клієнтами (CRM) або корпоративним порталом;

- забезпечення масштабованості системи для врахування зростання обсягу документів та користувачів;

- зручність використання, зокрема інтуїтивний та простий інтерфейс користувача;

- низькі апаратні вимоги до ІС.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ ВЕБ-СИСТЕМИ

2.1 Розробка користувальницьких вимог

Розглянемо основні практичні задачі та вимоги щодо проєктування інформаційної веб-системи керування документами на підприємстві.

Метою проекту є створення веб-рішення системи керування документами, яке буде використовуватися для ефективного управління документною системою підприємства або компанії.

Задання, що необхідно забезпечити наступний функціонал.

2.1.1 Механізм іменування файлів

Іменування файлів - Структура папок. Цей метод індексування рекомендується для простих або невеликих проектів для впорядкування документів.

Якщо вибрано цей метод індексування, важливо мати стабільну та зрозумілу угоду, щоб запобігти плутанині.

Тому його не рекомендують для складних завдань. «Деякі програми мають автоматичні угоди про іменування, які можна застосовувати, наприклад додавання поточної дати до назви файлу або додавання вказаних префіксів до цифрових послідовностей, щоб отримати щось на зразок A-0001.pdf, A-0002.pdf, A-0003 тощо .

Пошук текст за допомогою оптичного розпізнавання символів (OCR) – це механічне чи електронне перетворення зображень або письмового або написаного тексту в машинно-кодований текст, який можна редагувати та шукати.

Такий спосіб дозволяє швидко індексувати та зручно шукати будь-який формат електронного документа [14].

Документи із зображеннями обробляються далі за допомогою програм оптичного розпізнавання символів (OCR).

Тоді точність програм OCR визначатиме якість індексу.

2.1.2 Механізм повнотекстової індексації

Документи можуть бути індексовані за їх повнотекстовим вмістом, що означає, що вони переглядають усі слова в кожному документі та індексують документ за всіма цими словами. отже, їх можна отримати за будь-яким словом у вмісті [9].

Метадані є одним із методів індексування.

До документа додається додаткова інформація, наприклад унікальний ідентифікатор, дата створення або основна тема документа.

Метадані – це додаткова інформація про документ. Так, щоб його можна було використовувати для ідентифікації чогось або для того, щоб документи були легко доступними для пошуку.

2.1.3 Механізм отримання документів

Пошук документів є процедурою віднаходженням файлової інформації із неструктурованим текстом, який задовольняє інформаційну потребу у великих колекціях, які зазвичай зберігаються на комп’ютері, і це дозволяє користувачеві знайти відповідний документ із колекції незліченних ресурсів.

Коли документи зберігаються в центральній системі керування документами, залежно від того, як система організована та як користувачам

дозволено доступ до інформації, документи доступні для пошуку з централізованої системи.

Індексування є одним із важливих завдань інформаційного пошуку, яке можна застосувати до будь-якої форми даних, згенерованих з Інтернету, баз даних тощо.

Це означає, що при розгляді методів пошуку важливо перевірити різні методи індексування.

2.1.4 Забезпечення контролю версій документів

Забезпечення контролю версій документів є можливістю відстежувати та керувати кількома версіями однієї інформації.

Це досить важливий аспект ІСВКД . Щоб правильно керувати поточним розвитком цифрових даних, необхідно створювати та правильно зберігати кілька версій даних у міру їх проходження системою. ІСВКД може керувати контролем версій двома способами.

ІСВКД може заблокувати документ для інших користувачів під час використання, щоб гарантувати, що лише один користувач може отримати доступ до документа в будь-який момент часу.

Або у випадках, коли необхідно, щоб декілька користувачів одночасно працювали над одним документом, ІСВКД може забезпечити одночасне використання.

Тут система контролю версій має важливе значення.

Система контролю версій дозволяє одночасний доступ, зберігаючи головну копію документа.

Таким чином, користувачі можуть вносити зміни лише до останньої версії. Кілька користувачів можуть надіслати свої зміни до останньої версії. Цей процес також відомий як реєстрація.

Головна копія залишається незмінною протягом усього процесу роботи з системою.

Очікується, що кілька користувачів регулярно користуватимуться командою оновлення програмного забезпечення, яка оновлюватиме їхню копію документа для включення останніх змін.

Контроль версій корисний для документів, які змінюються з часом і потребують оновлення, але може знадобитися повернутися до попередньої копії або зробити посилання на неї [11].

Без такої функції, як контроль версій, адміністратору стає важко відстежити, ким і коли були внесені зміни.

Отже, контроль версій є однією з найважливіших функцій. Використовуючи функцію контролю версій, адміністратор може відстежувати зміни в різних версіях документа.

2.1.5 Процес керування документами

Робочий процес керування документами є системою, яка використовується для створення, відстеження, редагування, зберігання та керування документами, пов’язаними з бізнес-процесом.

На практиці це означає управління різними етапами документа. Передбачається, що в системі виконано всі етапи, і в системі є інформація про кожен крок.

Прикладом різних етапів є коментування документа, публікація документа, розповсюдження документа тощо.

Оскільки для цих етапів використовується керування документами, електронні листи користувачів не завантажуються так сильно під час обробки та переміщення всіх документів.

Оскільки робочий документ весь час зберігається в системі управління документами, і всі версії, зміни тощо також можна побачити звідти.

Процес документообігу переміщує документи з одного столу на інший традиційним способом, і він має свої недоліки, як втрата документів під час процесу, процес може тривати набагато довше, ніж мав би тощо.

Програмне забезпечення для керування документами створює робочі процеси, які допомагають модернізувати процес документообігу.

Документи стають цифровими і можуть оброблятися ефективніше. [34-16]

2.2 Методика розроблення інформаційної системи

Є різні методології, які візуалізують, як можуть бути виконані проекти розробки інформаційнихї систем

Це модель водоспаду, спіральна модель, гнучкі методи, поступова розробка тощо.

Усі ці моделі процесів візуалізують кроки, які зазвичай виконуються в проектах програмного забезпечення.

2.2.1 Застосування методології Agile

У методології Agile накладні витрати на традиційні процеси розробки програмного забезпечення та документацію намагаються зменшити.

Незалежно від того, який метод Agile було обрано, ключовою ідеєю є запуск швидкої інкрементальної та ітераційної розробки, де швидкі випуски переглядаються разом із клієнтом.

Причина, чому гнучка методологія не була обрана для цього проекту, полягає в тому, що гнучкий метод в основному зосереджений на кінцевих результатах роботи над проектом.

Це означає, що він уникає значної частини документації.

Швидше це вимагає багаторазових поставок і демонстрацій кінцевому користувачеві цієї системи.

2.2.2 Водоспадна модель

Водоспадну модель процесу зазвичай називають найстарішою та найвідомішою моделлю процесу [36].

Розглядаючи традиційний метод розробки програмного забезпечення,метод водоспадує жорсткою лінійною моделлю. У цій моделі є різні послідовні фази (вимоги, проектування, впровадження, перевірка, обслуговування).

Кожна окрема фаза отримує зворотний зв’язок від попередньої, тому кожна фаза має бути повністю завершена, перш ніж переходити до наступної фази, і традиційно немає процесу повернення для її зміни. [55-3]

2.2.3 Модифікований водоспад

Модифікована модель водоспаду тісно заснована на моделі розроблення ІС водоспаду [11].

Причина його існування полягає в тому, щоб мінімізувати або стерти дефекти або недоліки традиційної моделі водоспаду.

Основна зміна цієї моделі порівняно з моделлю водоспаду полягає в тому, що фази життєвого циклу модифікованої моделі водоспаду можуть накладатися.

Для розробки запропонованої системи обрано модифіковану модель Waterfall; Duo завдяки перерахованим нижче перевагам.

У модифікованій моделі водоспаду життєвий цикл дозволяється накладатися.

Він більш гнучкий у порівнянні з моделлю водоспаду.

Також можливе одночасне виконання декількох завдань.

У цій моделі застосовується менш формальний підхід до процедур, документів і перевірок. Завдяки цьому він зменшує величезний пакет документів. Завдяки цьому команда розробників має більше часу, щоб присвятити роботі над кодом, і їй не потрібно турбуватися про процедури. Це, в свою чергу, допомагає швидше завершити виріб.

Це забезпечує усунення дефектів у програмному забезпеченні на самій стадії розробки та зберігає додаткові витрати на внесення змін до програмного забезпечення перед впровадженням. Оскільки в один момент часу може бути декілька активних фаз, внесення змін до конструкції та виправлення помилок можна легко вирішити [11].

2.2.4 Забезпечення процедури збору даних

Для збору вимог користувачів до побудови системи електронного документообігу використовувалися різні методи збору даних.

Вважається, що якісне дослідження більше підходить для сфери інформаційних систем через зв’язок із розумінням людської поведінки з точки зору інформанта.

У роботі для було обрано якісний метод збору даних.

Якісні методи використовуються в інформаційній науці та практиці управління записами та інформаційними системами та сприяють звуженню інформаційної професії.

Цей аспект пояснюється рядом пояснень, таких як:

1) складність інформаційного середовища вимагає гнучкого розуміння складних соціальних конструкцій, що розвиваються, і мінливості аналізу даних;

2) якісне дослідження сприяє використанню тріангуляції для збагачення результатів досліджень;

3) він підходить для неякісного фону багатьох інформаційних професіоналів.

2.2.5 Забезпечення процедури перегляду документів

В роботі було розглянуто різні типи записів з підприємства чи компанії, щоб зібрати додаткові дані, які могли б забезпечити достатній внесок для розробки системи та триангуляції даних, зібраних під час інтерв'ю.

2.2.5 Забезпечення процедури перегляду результатів активності документообігу співробітниками

Забезпечення процедури перегляду результатів активності документообігу співробітниками передбачає систематичний перегляд або спостереження за виконанням співробітниками підприємства чи компанії виконаних завдань з документами.

Спостереження включає в себе те, як ведуться записи, який тип записів вони мають, як вони класифікують документи та як працівники підприємства чи компанії використовують процес документування.

Таке спостереження допоможе отримати більше контекстної інформації щодо роботи з документами.

Щоб досягти поставлених цілей щодо розробки веб-системи електронного документообігу, було проаналізовано існуючу систему та

прийнято об’єктно-орієнтовану методологію дизайну гіпермедіа (OOHDM) для розробки запропонованої веб-системи управління документами.

Цей підхід заснований на моделі метод для створення високоякісних і великих гіпермедійних додатків, інформаційних систем, мультимедійних презентацій тощо з різними діями, такими як концептуальний дизайн, навігаційний дизайн, абстрактний дизайн інтерфейсу та реалізація.

Користувач інформаційної системи досягає балансу між компромісами та п’ятьма критеріями, що визначають якість стратегії під час розробки підходів до стратегії на основі сценаріїв для веб-системи керування документами, використовуючи методологію гіпермедійного проектування на основі сценаріїв (SHDM).

Це дозволяє забезпечити персоналізовану візуалізацію необхідного процесу міркування на його фундаментальні когнітивні процеси з метою максимізації адаптивності, гнучкості та толерантності до складності.

Повторне агрегування в організовану, чітку та доступну «архітектуру міркування» дає змогу автору при розробці стратегії інтерактивно переосмислювати та базувати відкриття на загальних припущеннях, причинно-наслідкових зв’язках і пріоритетах. [2-26]

Цей модульний метод дозволяє використовувати різні інструменти та джерела вхідних даних, а також можливість змінювати порядок процесів.

Крім того, у поточному дослідженні запропонований метод було оцінено з використанням різних фундаментальних стандартів, таких як стандарти оцінки методології, стандарти навколишнього середовища та стандарти властивостей системи, щоб продемонструвати, як систему було створено відповідно до вимог програми.

Рівень охоплення можна приблизно виразити цілим числом, як і критерії, що використовуються для доступу до запропонованого підходу та середовища, в якому виконується впровадження.

Крім того, запропонований підхід, який використовувався в цьому дослідженні, оцінювали за допомогою структури, що складається з семи уявлень, які дотримуються різних типів і кількості фаз, які були чітко помітні в інформаційних системах документообігу.

На етапі аналізу було надано DFD, щоб представити межі системи та відрізнити цільову прикладну систему від зовнішнього середовища.

Крім того, надається сценарій для визначення інформаційних і навігаційних потреб користувачів.

Подібно до сценарію використання, сценарій — це серія дій.

Системи додатків і об’єктні моделі даних моделюються та створюються на етапі об’єктного моделювання за допомогою наборів сценаріїв спроєктованої діаграм класів.

Шляхи, які забезпечують гіпермедійну навігацію, створюються розробниками на етапі проектування навігації. діаграма класів, де розглядається кожен асоціативний зв'язок.

Після цього сценарії змінюються та використовуються для визначення потреб у навігації.

Інформаційні компоненти класів домену на діаграмі ієрархії класів згруповані в навігаційні одиниці, щоб забезпечити чітке уявлення, наприклад інтерфейс користувача, під час процесу проектування навігації.

Логічна структура та користувальницький інтерфейс наразі генеруються на етапі реалізації з використанням ряду системних налаштувань, включаючи різні СУБД, такі як RDBMS або OODBMS, і засоби розробки, такі як CGI, HTML і JAVA. [64-17]

2.3 Нефункційні вимоги до інформаційної системи

Нефункційні вимоги визначають різні атрибути системи, такі як безпека, надійність, продуктивність, зручність обслуговування, масштабованість, зручність використання, обробка помилок, доступність і ефективність. Ось як можна перефразувати деякі зазначені вимоги:

Продуктивність: Система керування документами повинна бути здатна підтримувати багато користувачів одночасно, особливо з урахуванням того, що передавання інформації зазвичай включає переважно текстові дані, що сприяє високій продуктивності.

Безпека. Адміністративний розділ системи має бути доступним тільки для уповноважених працівників Адміністрації з відповідними правами доступу. Неавторизованим користувачам повинен бути заборонений доступ до сторінки адміністрування. Контроль доступу та авторизація повинні бути використані для обмеження доступу до ресурсів та функцій в системі.

Доступність. Система повинна бути доступною з будь-якого комп'ютера з локальним підключенням, незалежно від часу, коли користувач бажає скористатися системою. Передбачення високої доступності є важливим аспектом для забезпечення незавершеного доступу користувачів до системи.

Інтуїтивність інтерфейсу користувача.: Система повинна мати інтерактивний, динамічний та послідовний графічний інтерфейс користувача, що не відволікає увагу. Використання списків та розкривних вікон для вводу даних робить процес зручним та ефективним для користувача.

2.4 Аналіз вимого до інформаційної системи

Запропонована інформаційна веб-система керування документами на підприємстві базується на залученні користувачів та інших користувачів на веб-

платформі для легкого доступу до документів, спільного використання, пошуку, покращеної безпеки, підвищення продуктивності та розвитку з колегами та клієнтами через мережу.

Модель розроблено з основним фокусом на сприянні зберіганню та витягу документів із бази даних, що стосується процесу адміністратора, який отримує, затверджує та видає документи для використання працівниками та іншими клієнтами.

Схема потоку даних пропонованої інформаційної веб-системи керування документами системи зображена на рисунку 2.1.

Адміністратор, співробітники відділу та секретар є зовнішніми суб'єктами системи, які взаємодіють разом.

Звіт про виконання та зворотний зв’язок – це потоки даних, які служать для збору кількох частин інформації.

Створення прав (новий персонал) і надання доступу адміністратором, і зворотній зв'язок - це дії, які виконуються з певних причин.

Це дозволяє адміністратору створювати облікові записи та надавати права секретарям, співробітникам відділу та іншим службовцям на платформі.

Секретар форми фіксує, отримує, сканує та завантажує документи, пам’ятки та інформаційні бюлетені, які містять інформацію про засідання ради викладачів, відділів, семінари та участь в інших заходах, що відбуваються в університетському середовищі.

Співробітники департаменту можуть переглядати звіт, переглядати, шукати та обмінюватися документами та звітами лише за допомогою особистих даних у системі або на будь-якому пристрої з доступом до Інтернету.

Співробітники департаменту також можуть надати свої коментарі через той же засіб.

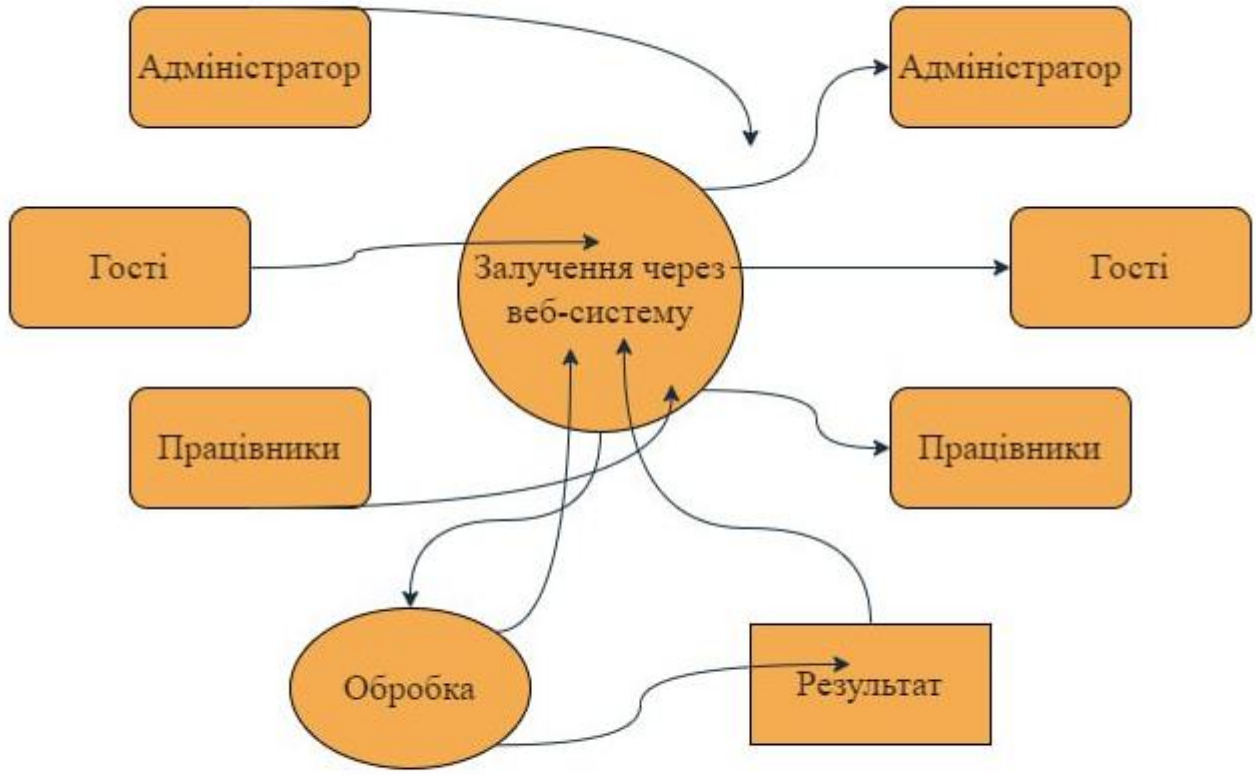


Рисунок 2.1 – Схема потоку даних пропонованої інформаційної веб-системи керування документами

Так само секретар завантажує документ, а співробітники можуть переглядати та роздруковувати документ для власного використання.

Запропоноване рішення пропонованої інформаційної веб-системи керування документами спрямоване на вирішення постійних проблем, спричинених неналежним керуванням документами в організації.

Ця система запроваджує систематичне управління документами та враховує деякі елементи, пов’язані з правильним керуванням та розташуванням документів для підтримки роботи, що виконується, і просування. [5-10]

Крім того, система сприяє використанню електронних методів для відстеження документообігу та простого пошуку документів.

Також ІС враховує, як часто документи обробляються в електронному вигляді, сприяючи розвитку та обміну контентом всередині компанії.

Деякі з функціональних можливостей, які надає ця система, включають повнотекстовий пошук, багатокористувацький доступ, створення вмісту, зберігання та структуроване зберігання, повний пошук документів, робочий процес документів і контроль версій документів.

2.5 Проектування інформаційної веб-системи керування документами

На рисунку 2.2 показано архітектура інформаційної веб-системи керування документами, яка була розроблена для забезпечення оптимального відображення її структури.

Клієнтський інтерфейс, проміжне програмне забезпечення та сховище бази даних складають основну його частину.

Рівень проміжного програмного забезпечення отримує та обробляє всі повідомлення.

Для реалізації цього використовувалася:

- програма XAMPP. HTML використовувався для створення інтерфейсів користувача (передній рівень на схемі),
- Apache служив проміжним програмним забезпеченням,
- MySQL служив базою даних, до якої можна отримати доступ через використаний інструмент PHP MyAdmin.

Варіант використання, показаний у таблиці 2.1, де подано перелік вимог до системи.

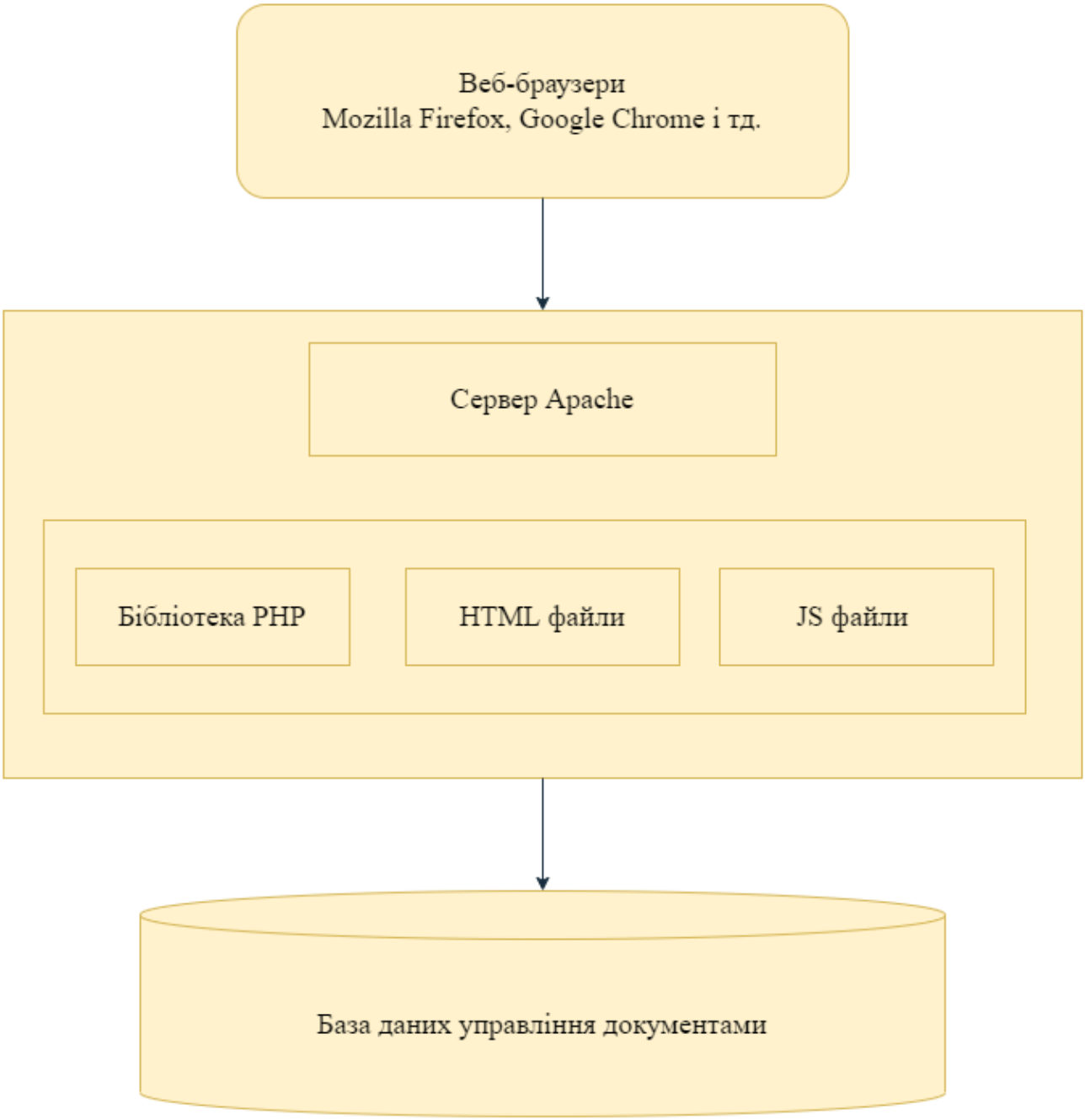


Рисунок 2.2 – Схема архітектури інформаційної веб-системи керування документами

Таблиця 2.1 – Вимоги до варіантів використання для запропонованої системи керування документами

Назва випадку використання	Опис	Актори
Логін	Щоб отримати доступ, користувач вводить в систему ім’я користувача та пароль	Секретар апарату відділу
Захоплення документів і сканування зображень	Усі документи, що надходять до відділу, фіксуються та скануються системою, а фізичні копії зберігаються в шафах	Секретар
Вставте новий запис	Основні відомості, включаючи дату, тип документа, джерело та призначення, реєструються разом з ідентифікованими документами.	Секретар
Внутрішній документообіг	У цьому розділі демонструється процес передачі документів у відділі.	Секретар апарату відділу
Назва випадку використання	Опис	Актори
Створити новий документ	Ця функція дозволяє користувачам створювати вміст (наприклад, листи, записки та звіти) за допомогою розроблених шаблонів документів, схожих на Microsoft Word, а потім ділитися цим вмістом з іншими співробітниками відділу.	Апарат відділу

Кінець таблиці 2.1 – Вимоги до варіантів використання для запропонованої системи керування документами

Переглянути наявні документи	Таким чином відкриваються створені та збережені документи у файлових сховищах.	Апарат відділу
Версії документа.	При цьому одночасно працювати над документом може лише одна особа розділ Блокування створюється, коли користувач реєструється, і потім знімається, коли користувач виходить.	Апарат відділу
Обмін документами	Користувач може дозволити іншим співробітникам переглядати, коментувати та ділитися документа за допомогою цього атрибута	Апарат відділу
Назва випадку використання	Опис	Актори
Базовий пошук і розширений пошук.	Користувач може отримати або відновити документи за необхідними критеріями пошуку	Апарат відділу
Журнали аудиту записів	Це відображає повні дані про кожен вхідний документ, отриманий	Секретар

	системою	
Панель адміністратора	здійснює технічне обслуговування системи та моніторинг продуктивності.	Системний адміністратор

2.6 Висновки

В розділі подано користувальницькі вимоги, зокрема, описано механізм іменування файлів, механізм отримання документів, забезпечення контролю версій документів, процес керування документами.

В розділ описано основні аспекти застосування методології Agile при проєктуванні ІС.

Описано процес забезпечення процедури збору даних та забезпечення процедури перегляду результатів активності документообігу співробітниками.

В розділі також надано нефункційні вимоги до інформаційної системи.

2.4 Аналіз вимого до інформаційної системи

Подано процес та аспекти проектування інформаційної веб-системи керування документами.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ ВЕБ-СИСТЕМИ КЕРУВАННЯ ДОКУМЕНТАМИ

3.1 Розроблення UML-діаграми для інформаційної веб-системи керування документами

Пропонована інформаційна система документів зображена на рисунках 3.1-3.6 відповідно.

Специфікації для нової системи були розроблені з використанням Unified Modeling Language Tools.

Діаграма варіантів використання окреслює систему, а також секретаря, персонал відділу, адміністратора та багато ролей, які кожен учасник відіграє в системі.

Діаграма варіантів використання - це графічний інструмент, який використовується для моделювання взаємодії між системою та її зовнішніми агентами. Її основна мета - визначити, проаналізувати та конкретизувати потреби користувачів системи.

Діаграма варіантів використання включає акторів, варіанти використання та зв'язки між ними. Актори - це сутності, які взаємодіють з системою, наприклад, користувачі, зовнішні системи або компоненти.

Варіанти використання представляють конкретні функціональні можливості системи, які можуть виконуватися акторами.

Основні завдання діаграми варіантів використання включають наступне:

1. Ідентифікація функціональних вимог: Діаграма варіантів використання допомагає виявити всі можливі варіанти використання системи і визначити їх функціональні вимоги.

2. Встановлення взаємозв'язків між акторами та варіантами використання: Діаграма дозволяє візуалізувати взаємодію акторів з різними варіантами використання і показати взаємозв'язки між ними.
3. Уточнення вимог: Діаграма варіантів використання допомагає уточнити вимоги до системи, виявити можливі протиріччя і прогалини у функціоналі.
4. Комунікація зі стейкхолдерами: Діаграма варіантів використання може бути ефективним інструментом для комунікації із зацікавленими сторонами та визначення їхніх потреб і очікувань від системи.

Призначення діаграми класів (class diagram) полягає в моделюванні структури об'єктно-орієнтованої системи шляхом відображення класів, їх атрибутів, методів та взаємозв'язків між класами. Вона надає графічний огляд основних складових системи і допомагає в розумінні її архітектури та взаємозв'язків між класами.

Основні завдання діаграми класів включають:

1. Візуалізація структури системи: Діаграма класів дозволяє представити класи системи, їх атрибути (змінні) та методи (функції), що дозволяє зрозуміти загальну структуру системи та її компонентів.

2. Моделювання взаємозв'язків між класами: Діаграма класів відображає взаємозв'язки між класами, такі як асоціації (зв'язки), агрегації, композиції, спадкування тощо. Це дозволяє виявити, як класи взаємодіють між собою та як вони спільно працюють для досягнення бізнес-цілей.

3. Уточнення атрибутів і методів класів: Діаграма класів допомагає уточнити атрибути та методи, які притаманні кожному класу. Це сприяє кращому розумінню функціональності класів та їх взаємозв'язків.

4. Підтримка проектування та рефакторингу: Діаграма класів є важливим інструментом для планування та проектування системи перед її

реалізацією. Вона дозволяє аналізувати та вдосконалювати архітектуру системи, а також виявляти можливі проблеми та вдосконалення

Об’єкти і класи з їх характеристиками та методами показано на діаграмі класів на рисунку 3.2.

Призначення діаграми послідовності (sequence diagram) полягає в моделюванні послідовності взаємодії об'єктів у системі в рамках певного сценарію або функціонального процесу.

Діаграма допомагає візуалізувати, як об'єкти взаємодіють між собою та як вони обмінюються повідомленнями впродовж певного часу.

Основні завдання діаграми послідовності включають:

1. Відображення послідовності взаємодії: Діаграма послідовності дозволяє показати порядок та послідовність виконання повідомлень між об'єктами. Це допомагає зрозуміти, як об'єкти спілкуються та взаємодіють для досягнення певної функціональності.

2. Моделювання взаємодії об'єктів: Діаграма послідовності відображає, як об'єкти взаємодіють між собою шляхом обміну повідомленнями. Вона дозволяє показати, як об'єкти викликають методи один від одного та як вони реагують на отримані повідомлення.

3. Виявлення паралельних та асинхронних процесів: Діаграма послідовності може показати паралельне виконання декількох об'єктів або асинхронні повідомлення, які не блокують виконання інших операцій. Це допомагає розуміти синхронізацію та координацію дій між об'єктами.

4. Валідація та аналіз системи: Діаграма послідовності може бути використана для валідації та аналізу системи. Вона дозволяє перевірити правильність взаємодії між об'єктами.

Призначення діаграми діяльності (activity diagram) полягає в моделюванні послідовності дій або процесу в системі з використанням різних діаграмних елементів, таких як стани, дії, розгалуження, злиття та зв'язки між ними. Ця

діаграма допомагає візуалізувати логіку та поведінку системи або функціонального процесу.

Основні завдання діаграми діяльності включають:

1. Моделювання послідовності дій: Діаграма діяльності дозволяє показати послідовність дій або процесу в системі. Вона використовує різні елементи, такі як стани та дії, для відображення виконуваних дій та їх порядку в рамках процесу.
2. Виявлення розгалужень та злиття: Діаграма діяльності дозволяє візуалізувати розгалуження (у разі коли процес має різні варіанти виконання) та злиття (об'єднання різних шляхів виконання). Це допомагає зрозуміти умови та логіку виконання дій в системі.
3. Моделювання паралельних процесів: Діаграма діяльності може відображати паралельне виконання декількох процесів або дій. Вона дозволяє показати, які дії виконуються одночасно та як вони взаємодіють між собою.
4. Валідація та аналіз системи: Діаграма діяльності може бути використана для валідації та аналізу системи. Вона дозволяє перевірити правильність послідовності дій та виявити можливі проблеми або неправильність у поведінці системи.

Кілька дій, які користувач може виконувати після того, як він отримав доступ до системи, описані на рис. 3.3.

Діаграма послідовності на рис. 3.4 ілюструє, як процеси взаємодіють один з одним. Він відображає послідовність часу, яка недостатньо чітко визначена в інших діаграмах, наприклад потік повідомлень, послідовність подій і дії між об’єктами.

Контекстна діаграма UML, зображена на рис. 3.5, показує елементи функціонального моделювання, які можна використовувати незалежно як корисний інструмент.

Це дозволяє групі або одній людині створити модель високого рівня запропонованої системи, яка визначає межі цільової системи та те, як вона взаємодіє з ключовими компонентами свого оточення.

Діаграма варіантів системи – це окрема ілюстрація з цікавою системою в центрі та об’єктами, які складають її середовище, з яким вона взаємодіє, з обох боків.

Діаграма взаємодії (interaction diagram) має наступні призначення:

Моделювання взаємодії об'єктів: Діаграма взаємодії дозволяє візуалізувати взаємодію між об'єктами або компонентами в системі. Вона показує, як об'єкти обмінюються повідомленнями та взаємодіють один з одним для досягнення певних цілей.

Показ послідовності повідомлень: Діаграма взаємодії відображає послідовність відправки та отримання повідомлень між об'єктами. Вона показує, які повідомлення відправляються, в якому порядку та як вони впливають на стан об'єктів.

Моделювання взаємодії зачинених систем: Діаграма взаємодії допомагає моделювати взаємодію зачинених систем, де об'єкти взаємодіють один з одним на основі певних правил та протоколів. Вона показує, як об'єкти спілкуються та виконують певні дії для досягнення спільної мети.

Аналіз та валідація системи: Діаграма взаємодії допомагає аналізувати та валідувати взаємодію в системі. Вона дозволяє перевірити правильність та працездатність взаємодії між об'єктами, виявити можливі проблеми або неправильність в логіці взаємодії.

Комунікація зі зацікавленими сторонами: Діаграма взаємодії є ефективним засобом комунікації з розробниками, зацікавленими сторонами та іншими учасниками проекту.

На цій діаграмі варіантів використання у нас є два основних актори: "Керування користувачами" та "Керування документами".

Актор "Керування користувачами" відповідає за дії, пов'язані з користувачами, такі як реєстрація користувачів, оновлення їхньої інформації та видалення користувачів.

Актор "Керування документами" відповідає за дії, пов'язані з документами, включаючи створення, редагування, видалення, перегляд і пошук необхідних документів.

Стрілки між акторами і варіантами використання відображають взаємодію або відносини між ними.

Наприклад, актор "Керування користувачами" взаємодіє з такими варіантами використання:

- "Реєстрація користувача".
- "Оновлення користувача".
- і "Видалення користувача".

Аналогічно, актор "Керування документами" взаємодіє з варіантами використання:

- "Створення документа".
- "Редагування документа".
- "Видалення документа".
- "Перегляд документа".
- "Пошук документа".

Діаграми варіантів використання корисні для візуалізації функціональних вимог і взаємодії між акторами та варіантами використання в системі.

Вони надають високорівневий огляд функціональності системи і допомагають зрозуміти поведінку системи з точки зору користувача.

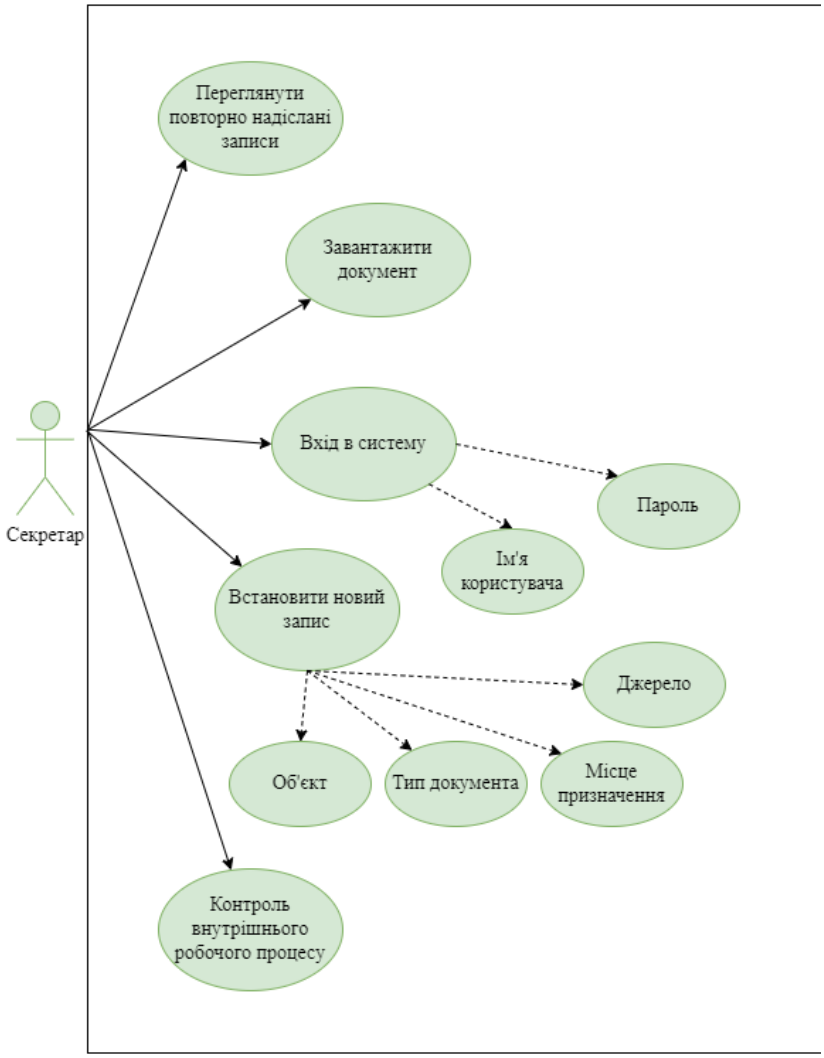
Як і діаграми послідовності, діаграми співпраці є діаграмами взаємодії. Інформація передається подібно до діаграм послідовності.

Натомість вони підкреслюють функціональну роль предметів.

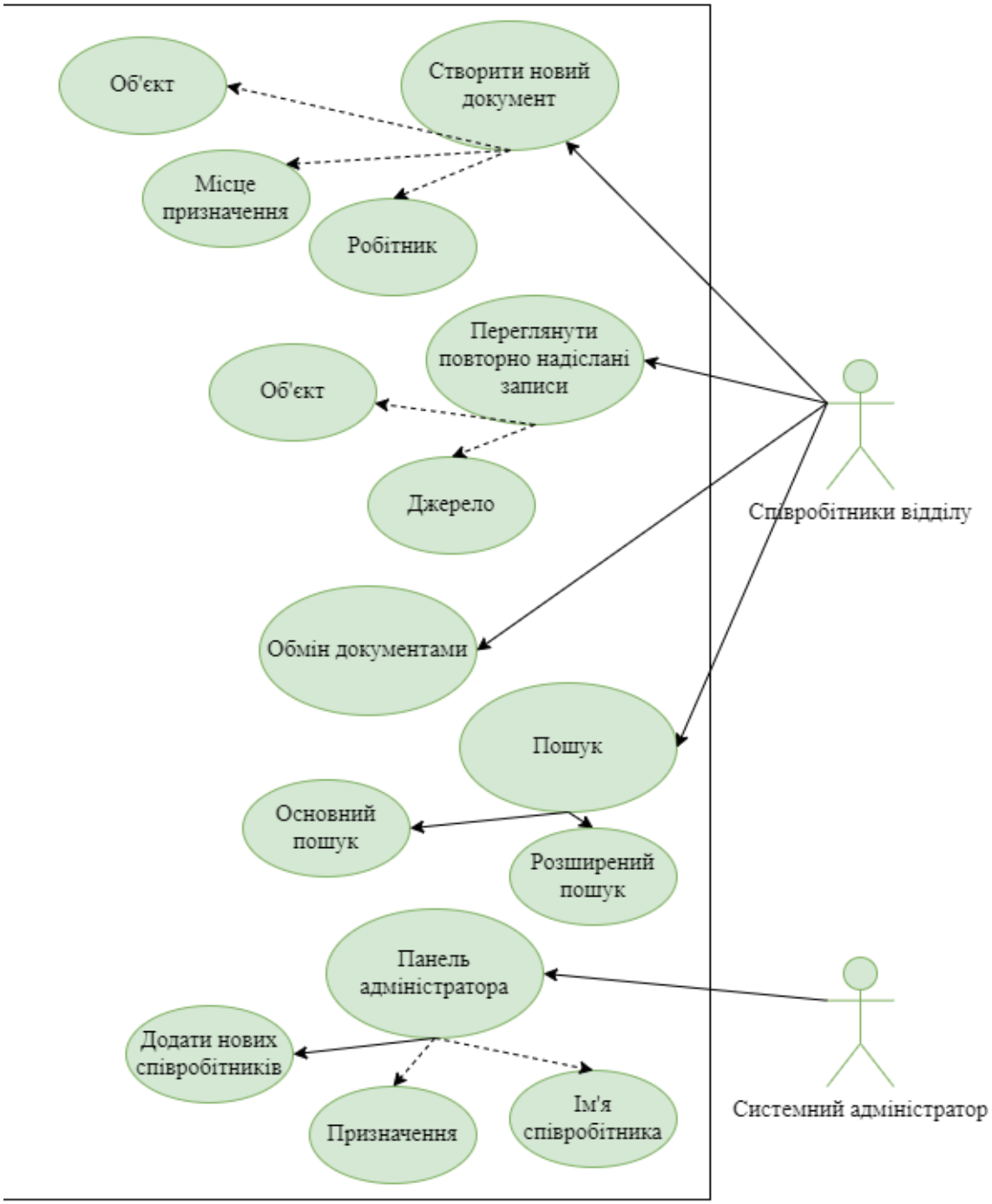
На рисунку 3.6 показана схема з акторами, об’єктами та наконечниками стрілок, яка представляє навігацію відповідно до використаних методів.

Діаграма на малюнку показує багато структур співпраці, які використовуються для характеристики кожної системної діяльності.

Він відображає дії, виконані для додавання нового елемента, створення нового документа, перегляду нещодавно доданих документів, проведення пошуку та керування внутрішнім робочим процесом.



а)



б)

Рисунок 3.1 – Діаграма варіантів використання інформаційної веб-системи керування документами: а) для актору «Секретар»; б) для акторів «Співробітник відділу» та «Системний адміністратор»

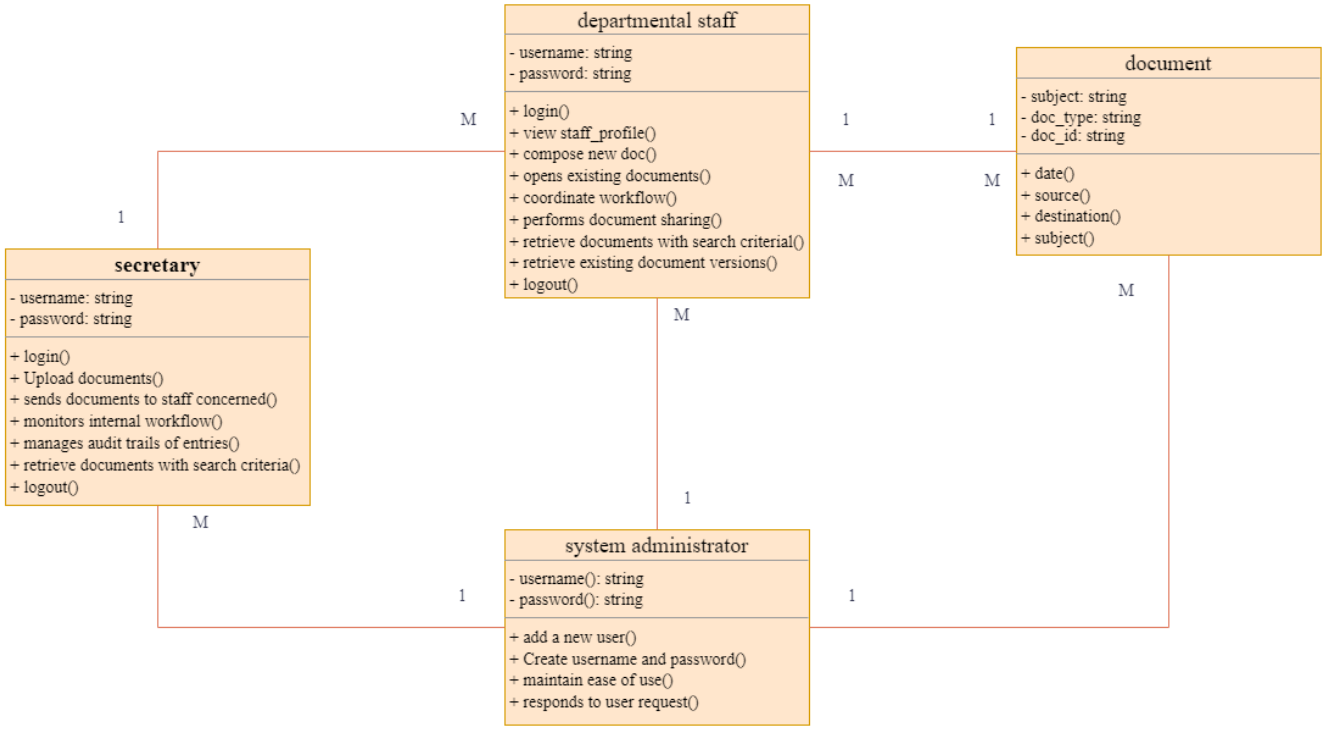


Рисунок 3.2 – Узагальнена діаграма класів інформаційної веб-системи керування документами

На діаграмі класів присутні маємо кілька основних класів: "Документ", "Користувач", "Тека", "Роль", "Менеджер документів" і "Менеджер користувачів".

Клас "Документ" представляє документ з такими атрибутами, як id, title, content, created_at і updated_at.

Клас "User" представляє користувача з такими атрибутами, як id, name та email.

Клас "Папка" представляє папку, яка може містити документи. Вона має такі атрибути, як id та name.

Клас "Роль" представляє роль користувача в системі і має такі атрибути, як ідентифікатор та ім'я.

Клас "DocumentManager" відповідає за керування документами та теками. Він має асоціації з класами "Документ" і "Папка", що дозволяє йому отримувати доступ до документів і папок та маніпулювати ними.

Клас "UserManager" відповідає за керування користувачами. Він має зв'язок з класом "User", що дозволяє йому додавати та видаляти користувачів.

Асоціації між класами відображають взаємозв'язки між ними. Наприклад, клас "DocumentManager" має асоціації з класами "Document" і "Folder", що вказує на те, що він може взаємодіяти з документами і папками та керувати ними. Аналогічно, клас "UserManager" має асоціацію з класом "User", що відображає його здатність керувати користувачами.

3.1 Програмна реалізація інформаційної веб-системи керування документами

Запропонована система була реалізована за допомогою інструменту NetBeans, який є програмним забезпеченням з відкритим кодом і інтегрованим середовищем розробки (IDE).

Програмне забезпечення NetBeans забезпечує середовище кодування для написання програм HTML, PHP, CSS і JavaScript і Java.

Графічний інтерфейс користувача було розроблено з використанням HTML, CSS і JavaScript.

Проміжне програмне забезпечення було розроблено з використанням Xampp (сервер HTTP Apache, PHP, MySQL і PHPMyAdmin).

Воно забезпечує масштабованість, балансування навантаження, обробку запитуваних транзакцій.

Внутрішню частину розроблено за допомогою інструменту бази даних SQLite.

Інструмент розробки бази даних використовувався через його надійність для збереження великих даних і його сумісність з багатьма мовами програмування, такими як PHP, JAVA та Python.

Visual Studio Code — це кросплатформний безкоштовний текстовий редактор із відкритим кодом, розроблений Microsoft.

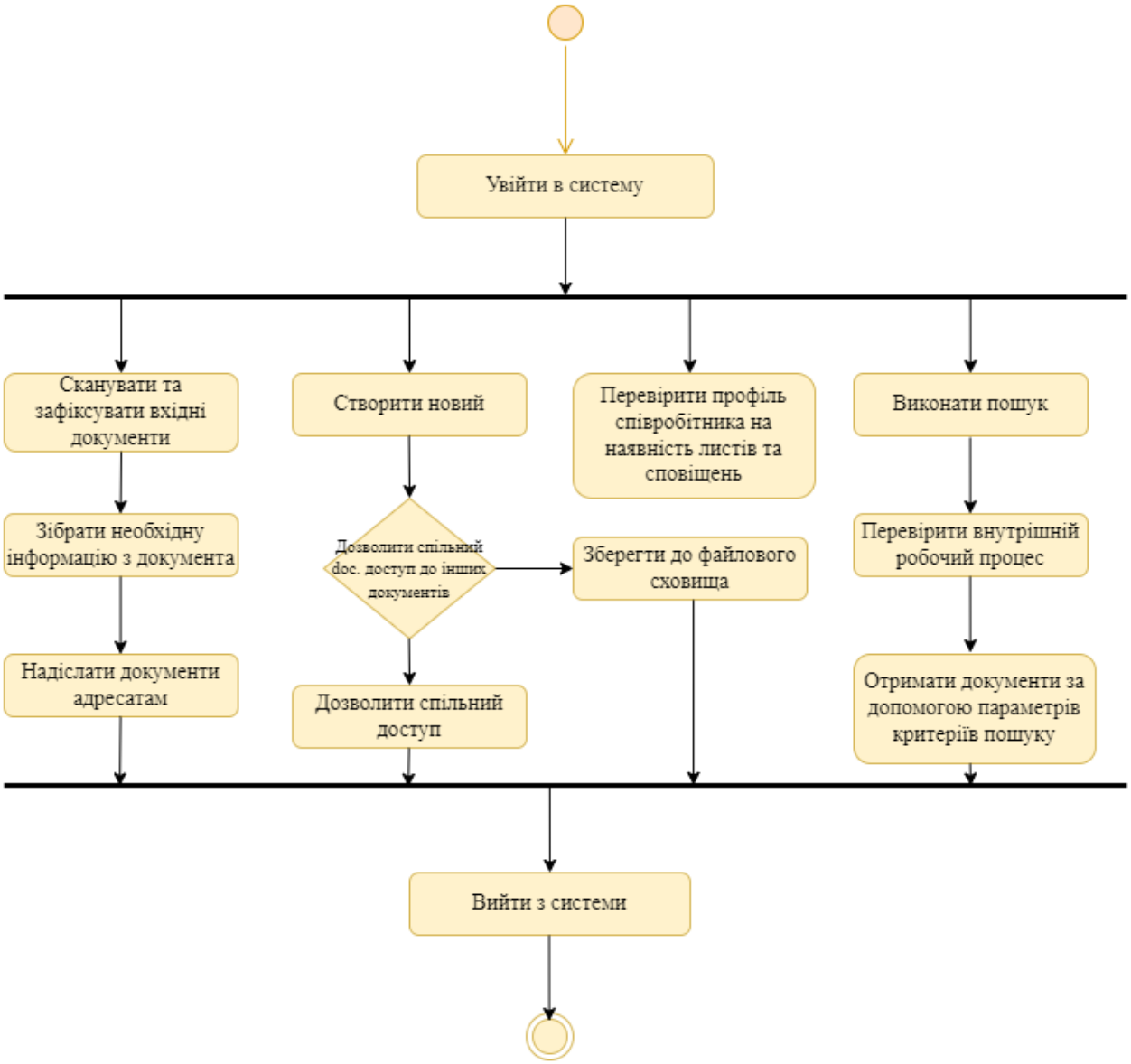


Рисунок 3.3 – Діаграма діяльності інформаційної веб-системи керування документами

Він побудований на основі Electron і розширюється за допомогою розширень, які можна переглядати в самому текстовому редакторі.

Це програмне забезпечення використовується як редактор PHP, оскільки воно швидко запускається та інтегрується з командним рядком.

Веб-сайт, який служить проміжним програмним забезпеченням і інтерфейсом рівня доступу до даних, відображається та обробляється веб-браузерами.

Програмне забезпечення інформаційної системи корпоративного документообігу було реалізована на персональному комп'ютері з наступними характеристиками:

- 32 ГБ оперативної пам'яті;
- процесор Intel Core (TM) i5-12500H, що працює на частоті 2,5 ГГц,
- 64-розрядна операційна система
- SSD диск на 1000 ГБ.

Для розповсюдження та обробки різних типів документів в організації запропонована веб-система управління документами є незалежною від платформи.

Інформаційна веб-система керування документами може бути доступна на локальному веб-сервері або віддалено розміщена в хмарі, як показано на рисунку 3.7.

3.1 Інтерфейс користувача інформаційної веб-системи керування документами

Запропонована система отримала інтуїтивний інтерфейс користувача.

Головна сторінка входу, показана на рис. 3.7, відображає сторінку, на якій працюють лише аутентифіковані користувачі, які можуть отримати авторизований доступ:

- секретар;
- співробітники відділу
- адміністративні службовці.

Автентифікований користувач може отримати доступ до системи лише за допомогою імені користувача та пароля, наданих йому адміністратором інформаційної системи.

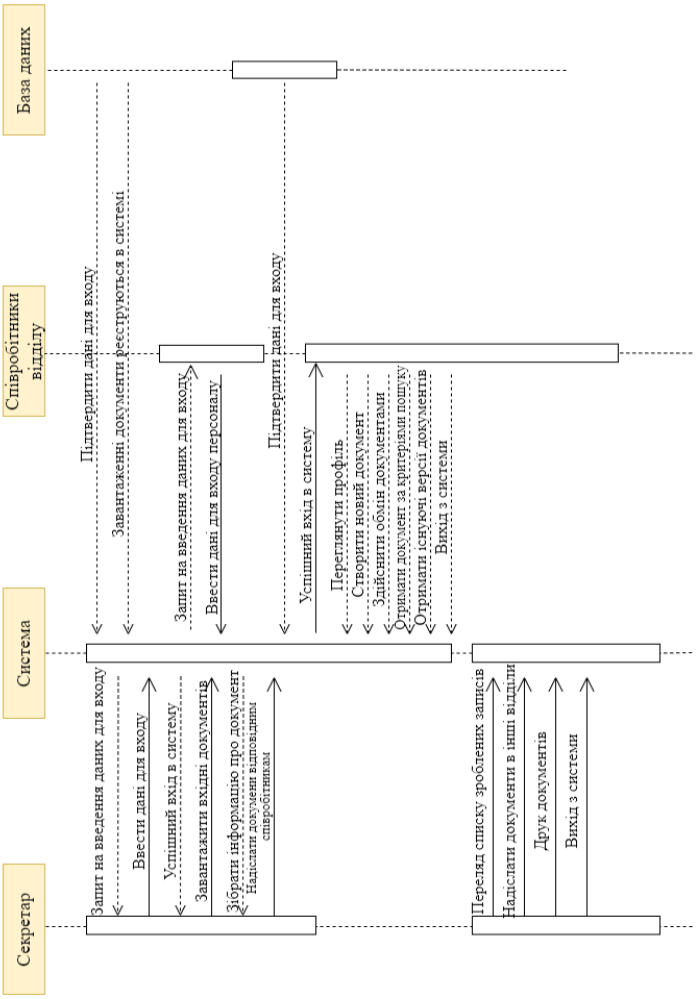


Рисунок 3.4 – Діаграма послідовності інформаційної веб-системи керування документами

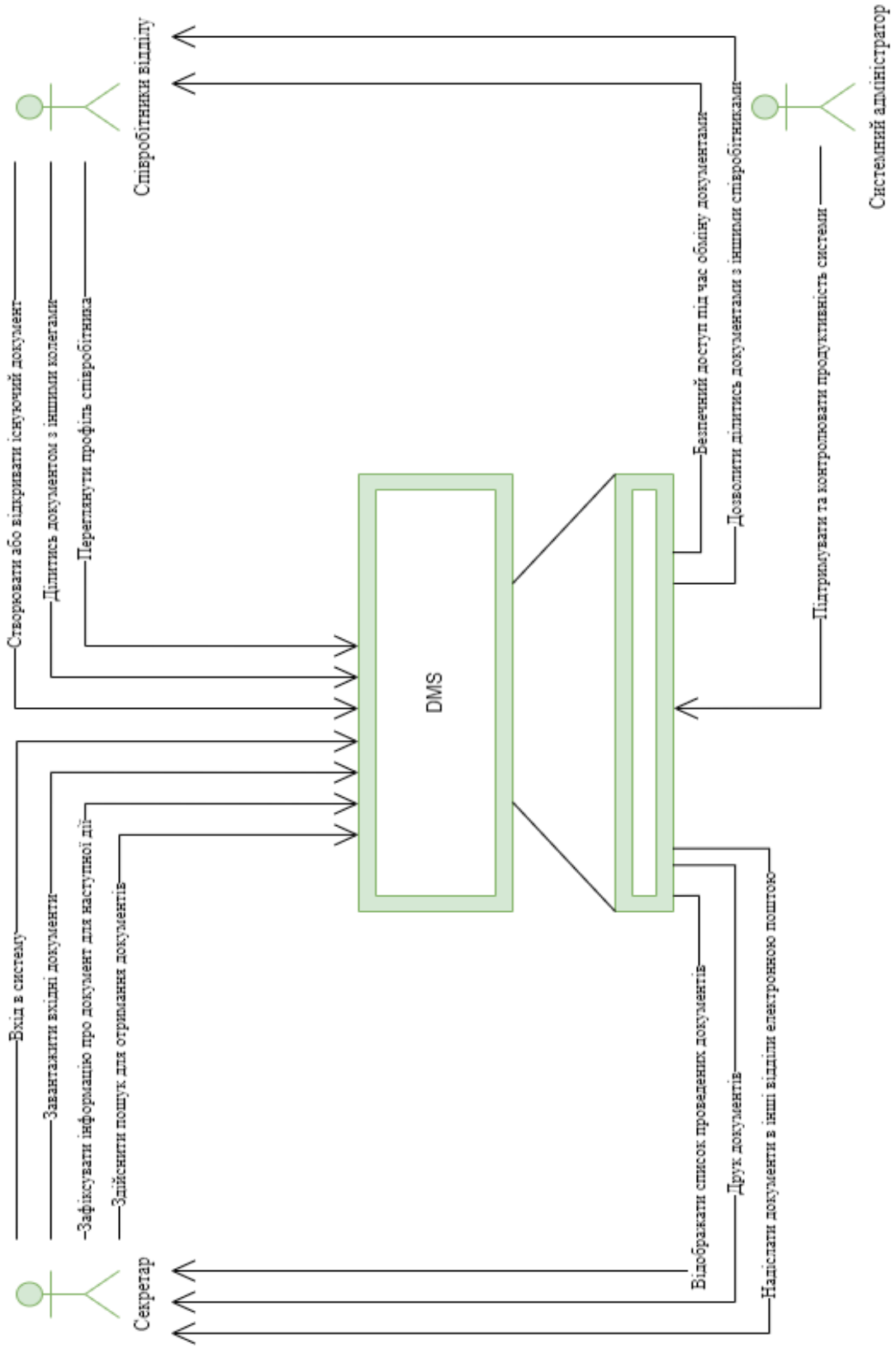


Рисунок 3.5. – Діаграма контекстної моделі інформаційної веб-системи керування документами

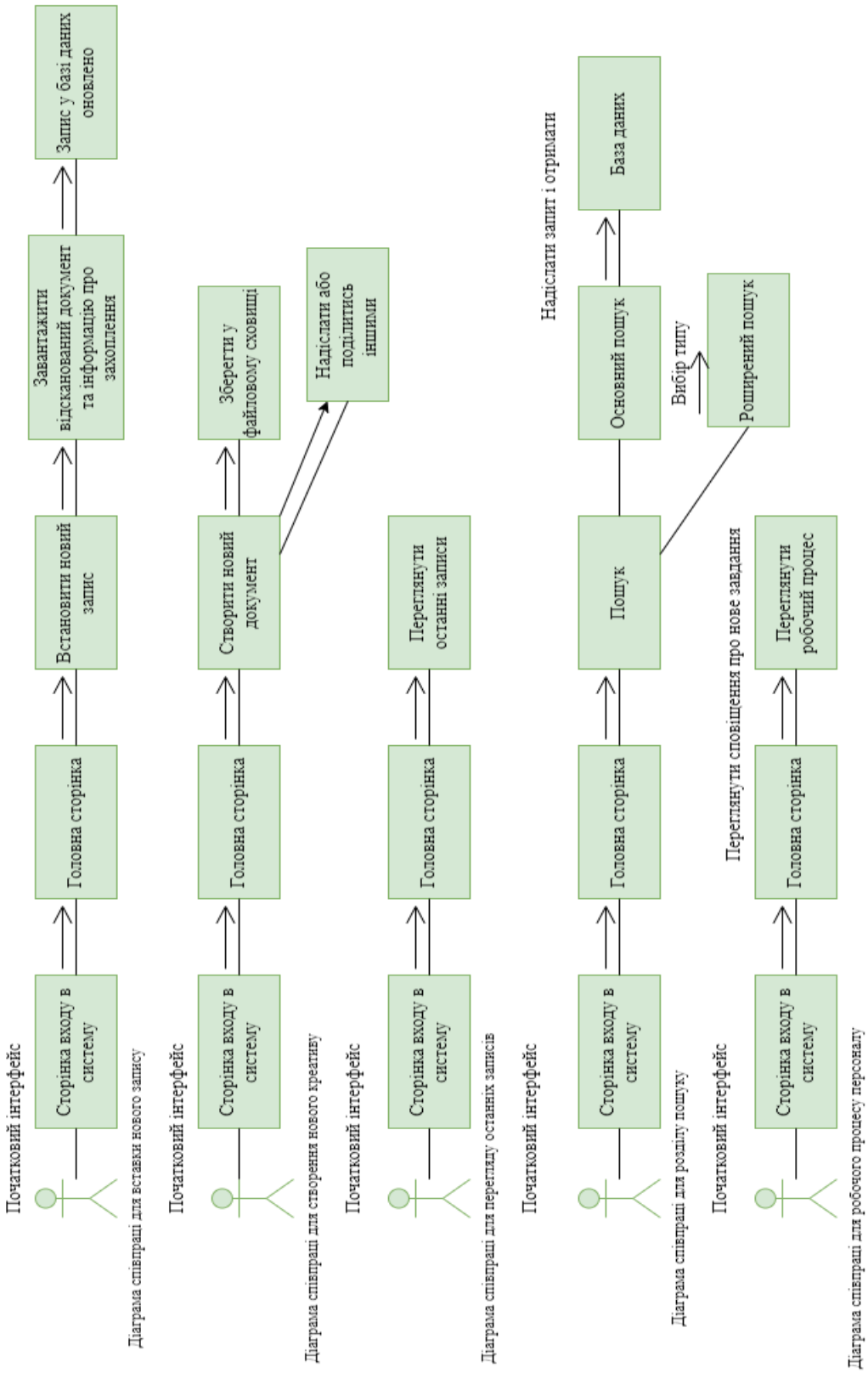


Рисунок 3.6 – Діаграма взаємодії запропонованої інформаційної веб-системи керування документами

Ім’я користувача та пароль дають їм доступ до інформаційної панелі.

Коли користувачі вводять правильну інформацію для входу, система розпізнає їхню роль (призначену під час додавання до системи) і відповідно перенаправляє їхню інформаційну панель.

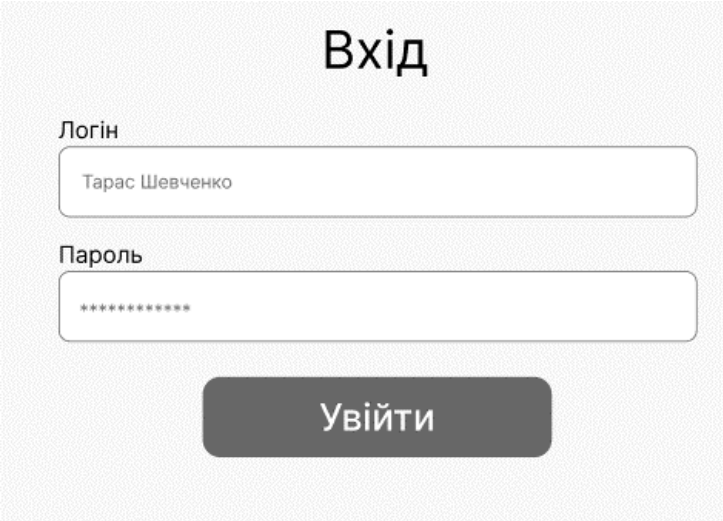


Рисунок 3.7 – Графічний інтерфейс користувача, на якому показано сторінку входу до веб-системи керування документами

Домашня сторінка, як показано на рис. 3.8 .

Вона є першою сторінкою, яку бачить користувач після входу в інформаційну систему.

Вона також служить головною сторінкою, яка спрямовує користувачів на інші сторінки системи.

Відповідно до інформаційної панелі, що відображається на сторінці, у формі є чотири розділи: вставити новий запис, створити новий документ, переглянути останній документ і знайти документ.

На рисунку 3.9 показано, що користувач може імпортувати документи, які надходять до відділу, використовуючи сторінку вставки нового запису.

Користувач записує всі відповідні дані про документ, включаючи його тип, тему або назву, автора, дату тощо.

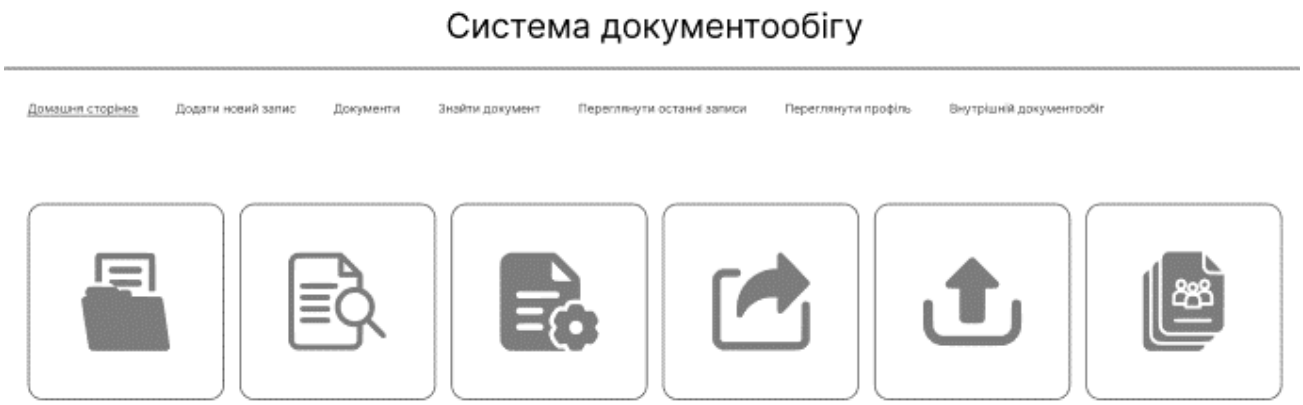


Рисунок 3.8 – Графічний інтерфейс користувача, на якому показано домашню сторінку системи документів

Користувач завантажує документ і зберігає його в системі після збору всіх необхідних даних.

Крім того, сторінка створення нового документа в системі, показана на рисунку 3.10.

Дана можливість пропонує платформу динамічного вмісту, де користувач може додавати текст до попередньо розроблених документів за потреби, що виникла.

У цій області користувачі можуть створювати документи та миттєво надсилати їх іншим співробітникам у організації.

Користувач зберігає та передає документ одержувачу в ІС після створення вмісту.

Сторінка інформаційної системи «Перегляд останніх документів», зображена на рисунку 3.11.

Вона відображає всі надіслані користувачем записи документів, що дозволяє користувачеві надіслати їх вказаним одержувачам.

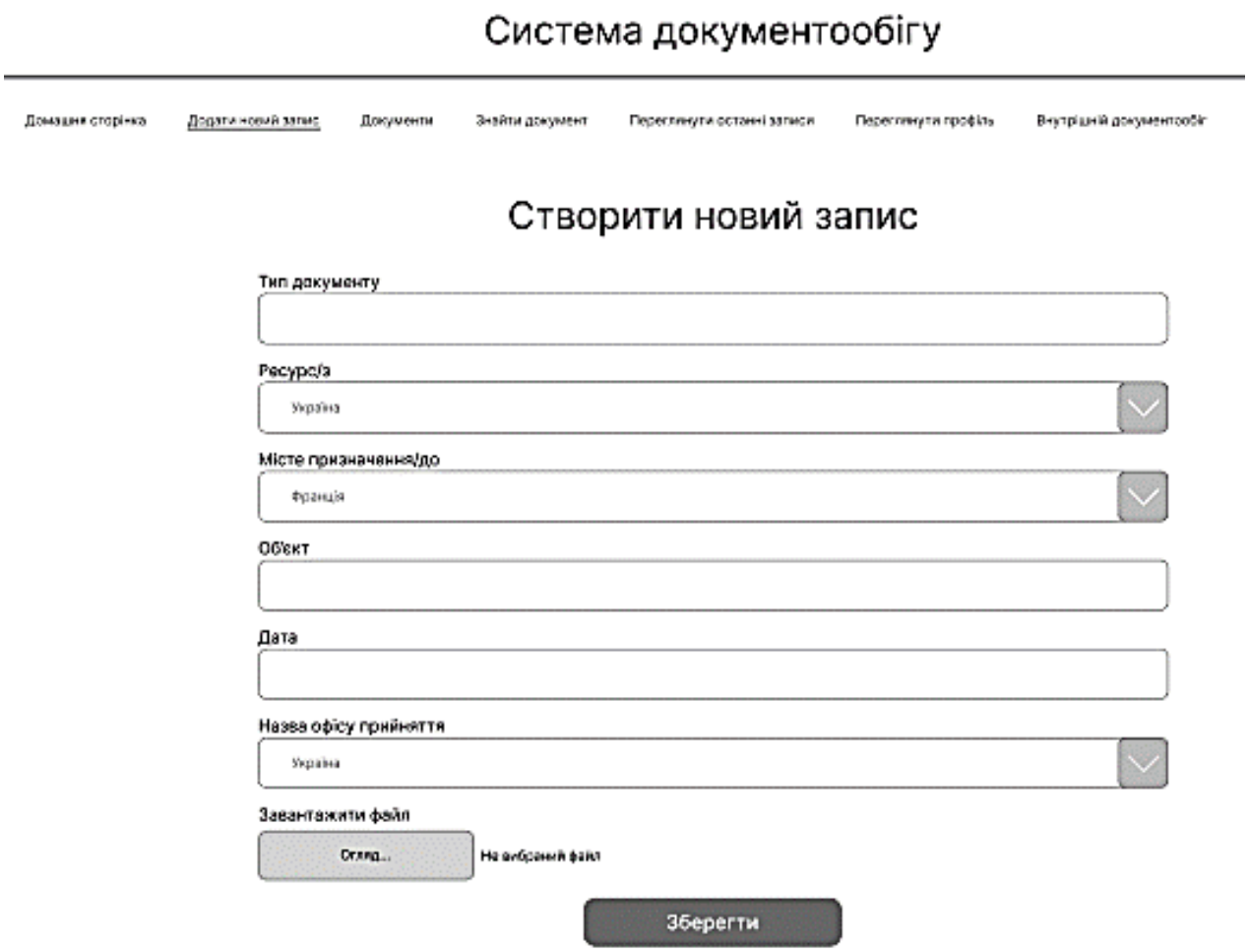


Рисунок 3.9 – Графічний інтерфейс користувача, на якому показано сторінку нового запису системи документів

Тоді як користувач може шукати та отримувати документи, що зберігаються в інформаційній системі, за допомогою екрана «Пошук документа», показаного на рисунку 3.12.

Система пропонує два типи категорій пошуку:

- базовий;

- розширений пошук.

Текстове поле, у якому користувач може ввести тему документа для пошуку, відображається в основному пошуку.

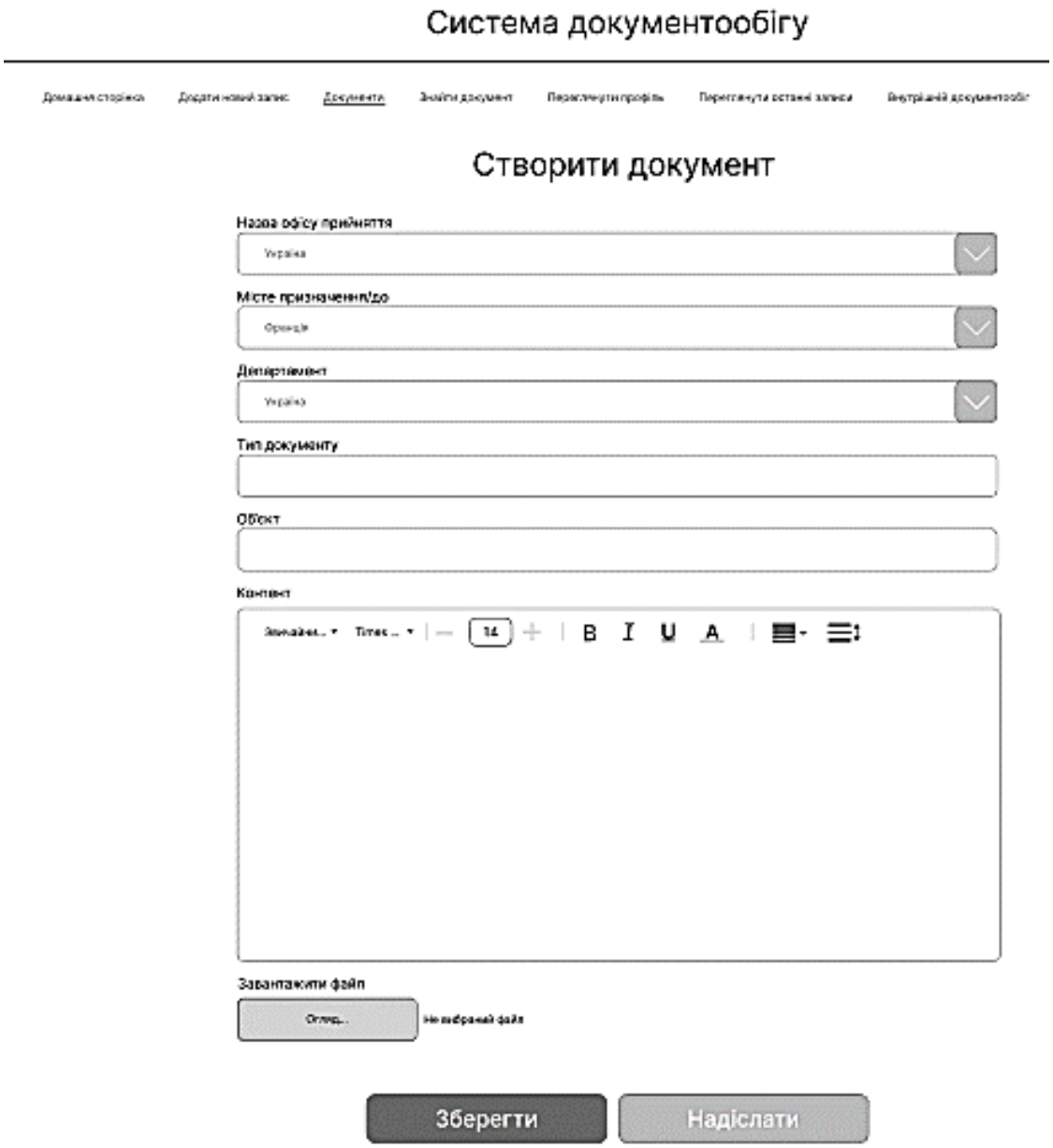


Рисунок 3.10 – Графічний інтерфейс користувача, на якому показано сторінку «Створити новий документ» у системі документів

Потім користувач може вводити різні текстові поля в розширений пошук, щоб обмежити кількість результатів, які повертаються після пошуку необхідного документу.

Рисунок 3.11 – Графічний інтерфейс користувача, на якому показано сторінку «Переглянути останні записи» системи документів

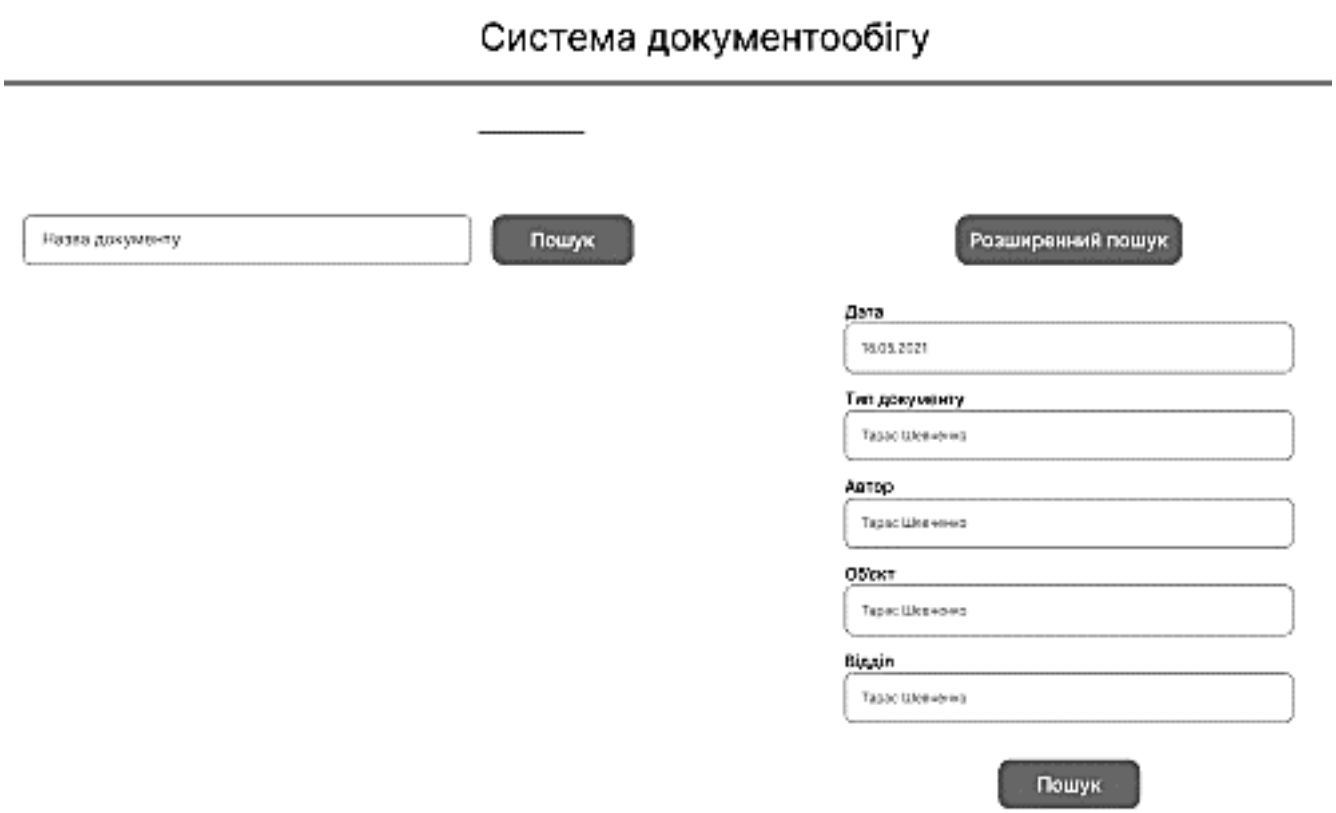


Рисунок 3.12 – Графічний інтерфейс користувача, на якому показано сторінку пошуку документів у системі документів

Подібно до сторінки «Зовнішній процес», сторінка «Внутрішній робочий процес», зображена на рисунку 3.13.

Рисунок 3.13 показує робочий процес і переміщення документів для кожного документа, що передається як у відділ, так і за його межі.

3.2 Методи тестування ІС

Для перевірки працездатності та зручності документообігу використовувалися різні методи та інструменти тестування.

Основна мета цих тестів — виявити обмеження системи та виміряти її повні можливості.

У кваліфікаційній роботі було застосовано наступні методи тестування розробленої ІС.

Рисунок 3.12 – Графічний інтерфейс користувача, на якому показано внутрішній потік системи документів

3.6.1 Модульне тестування

Цей тест було проведено, щоб перевірити, чи внутрішня логіка функціонує належним чином і вхідні дані програми створюють дійсні результати, які порівнюються з очікуваними результатами.

Це робиться після завершення окремої одиниці.

Ці тести виконуються на рівні компонентів і конкретного бізнес-процесу, програми та конфігурації системи.

3.6.2 Функційний тест

Функціональні тести забезпечують систематичну демонстрацію того, що перевірені функції доступні відповідно до бізнес-технічних вимог, системної документації системи управління документами.

3.6.3 Тест інформаційної системи

Системні тести були зосереджені на поведінці системи керування документами.

Користувацькі сценарії були виконані проти системи, а також відображення екрана та тестування повідомлень про помилки.

Загалом система протестувала інтегровану систему та підтвердила, що вона відповідає вимогам, визначеним у документі вимог.

3.6.4 Тест безпеки

Основною метою тестування безпеки є збереження доступності, цілісності та конфіденційності інформації:

1. Наявність гарантує, що авторизований користувач має доступ до інформації та пов’язаних активів, коли це необхідно.
2. Цілісність — це збереження точності та повноти інформації та методів обробки.
3. Конфіденційність гарантує, що інформація доступна лише авторизованим користувачам.

3.6.5 Тест юзабіліті ІС

Перевірка зручності використання зосереджена на перевірці зручності для користувача.

У цьому тесті тестування перевіряється зручність використання ІС, щоб користувач системи міг легко зрозуміти розроблену систему.

3.7 Висновки

В розділі здійснено проєктування архітектури інформаційної технології , зокрема розроблено UML-діаграми для інформаційної веб-системи керування документами.

Здійснено програмну реалізацію інформаційної веб-системи керування документами.

Розроблено інтерфейс користувача інформаційної веб-системи керування документами.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було спроєктовано та реалізовано інформаційну веб-систему керування документами на підприємстві.

В першому розділі проведено аналіз відомих інформаційних веб-систем керування документами показав їх наявні переваги та недоліки.

Зокрема було зроблено висновок, що інформаційної системи керування документами можуть різнитися залежно від конкретних потреб та контексту організації

В другому розділі подано користувальницькі вимоги, зокрема, описано механізм іменування файлів, механізм отримання документів, забезпечення контролю версій документів, процес керування документами.

В розділ описано основні аспекти застосування методології Agile при проєктуванні ІС.

Описано процес забезпечення процедури збору даних та забезпечення процедури перегляду результатів активності документообігу співробітниками.

В розділі також надано нефункційні вимоги до інформаційної системи та виконано аналіз вимого до інформаційної системи.

Подано процес та аспекти проектування інформаційної веб-системи керування документами.

В третьому розділі здійснено проєктування архітектури інформаційної технології, зокрема розроблено UML-діаграми для інформаційної веб-системи керування документами. Здійснено програмну реалізацію інформаційної веб-системи керування документами. Розроблено інтерфейс користувача інформаційної веб-системи керування документами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Heidmann, E.F., von Kurnatowski, L., Meinecke, A., Schreiber, A. Visualization of Evolution of Component-Based Software Architectures in Virtual Reality. *Proceedings of the 2020 Working Conference on Software Visualization (VISSOFT),* Adelaide, Australia, 28 September–2 October 2020. IEEE: Piscataway, NJ, USA, 2020. pp. 12–21.
2. Mekni, M., Buddhavarapu, G., Chinthapatla, S., Gangula, M. Software Architectural Design in Agile Environments. J. Comput. Commun. 2017, 6, 171–189.
3. Kil, B.-H., Park, J.-S., Ryu, M.-H., Park, C.-Y., Kim, Y.-S., Kim, J.-D. Cloud-Based Software Architecture for Fully Automated Point-of-Care Molecular Diagnostic Device. Sensors 2020. 21, 6980.
4. Lagsaiar, L., Shahrour, I., Aljer, A., Soulhi, A. Modular Software Architecture for Local Smart Building Servers. *Sensors*. 2020. 21, 5810.
5. Ungurean, I., Gaitan, N.C. A software architecture for the Industrial Internet of Things—A conceptual model. Sensors 2020. 20, 5603.
6. Piao, Y.C.K., Ezzati-Jivan, N., Dagenais, M.R. Distributed Architecture for an Integrated Development Environment, Large Trace Analysis, and Visualization. *Sensors* 2020. 21, 5560.
7. Dickerson, C.E., Wilkinson, M., Hunsicker, E., Ji, S., Li, M., Bernard, Y., Bleakley, G., Denno, P. Architecture definition in complex system design using model theory. *IEEE Syst. J.* 2020. 15, 1847–1860.
8. Yang, C., Liang, P., Avgeriou, P., Eliasson, U., Heldal, R., Pelliccione, P. Architectural assumptions and their management in industry—An exploratory study. *Proceedings of the European Conference on Software Architecture*, Canterbury, UK, 11–15 September 2017, Springer: Berlin/Heidelberg, Germany, 2017, pp. 191–207.

9. Harrison, N., Avgeriou, P. Pattern-based architecture reviews. *IEEE Softw.* 2010, 28, 66–71.
10. Yang, T., Jiang, Z., Shang, Y., Norouzi, M. Systematic review on next-generation web-based software architecture clustering models. Comput. Commun. 2020. 167, 63–74.
11. Allian, A.P., Sena, B., Nakagawa, E.Y. Evaluating variability at the software architecture level: An overview. Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, Limassol, Cyprus, 8–12 April 2019, pp. 2354–2361.
12. Sedaghatbaf, A., Azgomi, M.A. SQME: A framework for modeling and evaluation of software architecture quality attributes. Softw. Syst. Model. 2019, 18, 2609–2632.
13. Venters, C.C., Capilla, R., Betz, S., Penzenstadler, B., Crick, T., Crouch, S., Nakagawa, E.Y., Becker, C., Carrillo, C. Software sustainability: Research and practice from a software architecture viewpoint. J. Syst. Softw. 2018, 138, 174–188.
14. Link, D., Behnamghader, P., Moazeni, R., Boehm, B. Recover and RELAX: Concern-oriented software architecture recovery for systems development and maintenance. Proceedings of the 2019 IEEE/ACM International Conference on Software and System Processes (ICSSP), Montreal, QC, Canada, 25–26 May 2019, IEEE: Piscataway, NJ, USA, 2019, pp. 64–73.
15. Medvidovic, N., Taylor, R.N. Software architecture: Foundations, theory, and practice. Proceedings of the 2010 ACM/IEEE 32nd International Conference on Software Engineering, Cape Town, South Africa, 1–8 May 2010, IEEE: Piscataway, NJ, USA, 2010, Volume 2, pp. 471–472.
16. Kazman, R., Bass, L., Klein, M., Lattanze, T., Northrop, L. A basis for analyzing software architecture analysis methods. Softw. Qual. J. 2005, 13, 329–355.

17. Aboutaleb, H., Monsuez, B. Measuring complexity of system/software architecture using Higraph-based model. Proceedings of the International Multiconference of Engineers and Computer Scientists, Hong Kong, China, 15–17 March 2017, Newswood Limited: Hong Kong, China, 2017, Volume 1, pp. 92–96.
18. Garcés, L., Oquendo, F., Nakagawa, E.Y. Towards a taxonomy of software mediators for systems-of-systems. Proceedings of the VII Brazilian Symposium on Software Components, Architectures, and Reuse, Sao Carlos, Brazil, 17–21 September 2018, pp. 53–62.
19. Kanade, A. Event-Based Concurrency: Applications, Abstractions, and Analyses. In Advances in Computers, Elsevier: Amsterdam, The Netherlands, 2019, Volume 112, pp. 379–412.
20. Rodriguez-Priego, E., García-Izquierdo, F.J., Rubio, Á.L. Modeling issues: A survival guide for a non-expert modeler. Proceedings of the International Conference on Model Driven Engineering Languages and Systems, Oslo, Norway, 3–8 October 2010, Springer: Berlin/Heidelberg, Germany, 2010, pp. 361–375.
21. Medvidovic, N., Taylor, R.N. A classification and comparison framework for software architecture description languages. IEEE Trans. Softw. Eng. 2000, 26, 70–93.
22. Jensen, K., Kristensen, L.M. Introduction to Modelling and Validation. In Coloured Petri Nets, Springer: Berlin/Heidelberg, Germany, 2009, pp. 1–12.
23. Emadi, S., Shams, F. Mapping Annotated Use Case and Sequence Diagrams to a Petri Net Notation for Performance Evaluation. Proceedings of the Second International Conference on Computer and Electrical Engineering (ICCEE'09), Dubai, United Arab Emirates, 28–30 December 2009, pp. 67–81.

24. Sievi-Korte, O., Richardson, I., Beecham, S. Software architecture design in global software development: An empirical study. J. Syst. Softw. 2019, 158, 110400.
25. Jaiswal, M. Software Architecture and Software Design. Int. Res. J. Eng. Technol. (IRJET) 2019, 6, 2452–2454.
26. Hofmeister, C., Kruchten, P., Nord, R.L., Obbink, H., Ran, A., America, P. Generalizing a model of software architecture design from five industrial approaches. Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05), Pittsburgh, PA, USA, 6–10 November 2005, IEEE: Piscataway, NJ, USA, 2005, pp. 77–88.
27. Wu, X.-W., Li, C., Wang, X., Yang, H.-J. A creative approach to reducing ambiguity in scenario-based software architecture analysis. Int. J. Autom. Comput. 2019, 16, 248–260.
28. Iacob, M.-E. Architecture analysis. In Enterprise Architecture at Work, Springer: Berlin/Heidelberg, Germany, 2017, pp. 215–252.
29. Khatchatoorian, A.G., Jamzad, M. Architecture to improve the accuracy of automatic image annotation systems. IET Comput. Vis. 2020. 14, 214–223.
30. Júnior, A.A., Misra, S., Soares, M.S. A systematic mapping study on software architectures description based on ISO/IEC/IEEE 42010: 2011. Proceedings of the International Conference on Computational Science and Its Applications, Saint Petersburg, Russia, 1–4 July 2019, Springer: Berlin/Heidelberg, Germany, 2019, pp. 17–30.
31. Cabac, L., Haustermann, M., Mosteller, D. Software development with Petri nets and agents: Approach, frameworks and tool set. Sci. Comput. Program. 2018, 157, 56–70.
32. Siefke, L., Sommer, V., Wudka, B., Thomas, C. Robotic Systems of Systems Based on a Decentralized Service-Oriented Architecture. Robotics 2020. 9, 78.

33. Hasselbring, W. Software architecture: Past, present, future. In The Essence of Software Engineering, Springer: Cham, Switzerland, 2018, pp. 169–184.
34. Breivold, H.P., Crnkovic, I. A systematic review on architecting for software evolvability. Proceedings of the 2010 21st Australian Software Engineering Conference, Auckland, New Zealand, 6–9 April 2010, IEEE: Piscataway, NJ, USA, 2010, pp. 13–22.
35. Barcelos, R.F., Travassos, G.H. Evaluation Approaches for Software Architectural Documents: A Systematic Review. Proceedings of the CIbSE 2006, La Plata, Argentina, 24–28 April 2006, pp. 433–446.
36. Roy, B., Graham, T.N. Methods for evaluating software architecture: A survey. Sch. Comput. TR 2008, 545, 82.
37. Mattsson, M., Grahn, H., Mårtensson, F. Software architecture evaluation methods for performance, maintainability, testability, and portability. Proceedings of the Second International Conference on the Quality of Software Architectures, Västerås, Sweden, 27–29 June 2006, Citeseer: Princeton, NJ, USA, 2006.
38. Christensen, H.B., Hansen, K.M. An empirical investigation of architectural prototyping. J. Syst. Softw. 2010, 83, 133–142.
39. Babar, M.A., Gorton, I. Comparison of scenario-based software architecture evaluation methods. Proceedings of the 11th Asia-Pacific Software Engineering Conference, Busan, Korea, 30 November–3 December 2004, IEEE: Piscataway, NJ, USA, 2004, pp. 600–607.
40. Maranzano, J.F., Rozsypal, S.A., Zimmerman, G.H., Warnken, G.W., Wirth, P.E., Weiss, D.M. Architecture reviews: Practice and experience. IEEE Softw. 2005, 22, 34–43.
41. Nakamura, T., Basili, V.R. Metrics of software architecture changes based on structural distance. Proceedings of the 11th IEEE International Software

Metrics Symposium (METRICS'05), Como, Italy, 19–22 September 2005, IEEE: Piscataway, NJ, USA, 2005, p. 24.

42. Knodel, J., Naab, M. How to Evaluate Software Architectures: Tutorial on Practical Insights on Architecture Evaluation Projects with Industrial Customers. Proceedings of the 2017 IEEE International Conference on Software Architecture Workshops (ICSAW), Gothenburg, Sweden, 5–7 April 2017, IEEE: Piscataway, NJ, USA, 2017, pp. 183–184.
43. Babar, M.A., Gorton, I. Software architecture review: The state of practice. Computer 2009, 42, 26–32.
44. Tekinerdoğan, B., Akşit, M. Classifying and evaluating architecture design methods. In Software Architectures and Component Technology, Springer: Berlin/Heidelberg, Germany, 2002, pp. 3–27.
45. Klein, M.H., Kazman, R., Bass, L., Carriere, J., Barbacci, M., Lipson, H. Attribute-based architecture styles. Proceedings of the Working Conference on Software Architecture, San Antonio, TX, USA, 22–24 February 1999, Springer: Berlin/Heidelberg, Germany, 1999, pp. 225–243.
46. Arvanitou, E.M., Ampatzoglou, A., Chatzigeorgiou, A., Galster, M., Avgeriou, P. A mapping study on design-time quality attributes and metrics. J. Syst. Softw. 2017, 127, 52–77.
47. Fünfrocken, M., Otte, A., Vogt, J., Wolniak, N., Wieker, H. Assessment of ITS architectures. IET Intell. Transp. Syst. 2018, 12, 1096–1102.
48. Babar, M.A., Shen, H., Biffl, S., Winkler, D. An Empirical Study of the Effectiveness of Software Architecture Evaluation Meetings. IEEE Access 2019, 7, 79069–79084.
49. Savold, R., Dagher, N., Frazier, P., McCallam, D. Architecting cyber defense: A survey of the leading cyber reference architectures and frameworks. Proceedings of the 2017 IEEE 4th International Conference on

Cyber Security and Cloud Computing (CSCloud), New York, NY, USA, 26–28 June 2017, IEEE: Piscataway, NJ, USA, 2017, pp. 127–138.

50. de Jong, P., van der Werf, J.M.E., van Steenbergen, M., Bex, F., Brinkhuis, M. Evaluating design rationale in architecture. Proceedings of the 2019 IEEE International Conference on Software Architecture Companion (ICSA-C), Hamburg, Germany, 25–26 March 2019, IEEE: Piscataway, NJ, USA, 2019, pp. 145–152.
51. Shahbazian, A., Lee, Y.K., Le, D., Brun, Y., Medvidovic, N. Recovering architectural design decisions. Proceedings of the 2018 IEEE International Conference on Software Architecture (ICSA), Seattle, WA, USA, 30 April–4 May 2018, IEEE: Piscataway, NJ, USA, 2018, pp. 95–9509.
52. Krusche, S., Bruegge, B. CSEPM-a continuous software engineering process metamodel. Proceedings of the 2017 IEEE/ACM 3rd International Workshop on Rapid Continuous Software Engineering (RCoSE), Buenos Aires, Argentina, 22–23 May 2017, IEEE: Piscataway, NJ, USA, 2017, pp. 2–8.
53. Arcelli, D. Exploiting queuing networks to model and assess the performance of self-adaptive software systems: A survey. Procedia Comput. Sci. 2020. 170, 498–505.
54. Palensky, P., van der Meer, A.A., Lopez, C.D., Joseph, A., Pan, K. Cosimulation of intelligent power systems: Fundamentals, software architecture, numerics, and coupling. IEEE Ind. Electron. Mag. 2017, 11, 34–50.
55. Szmuc, W., Szmuc, T. Towards Embedded Systems Formal Verification Translation from SysML into Petri Nets. Proceedings of the 2018 25th International Conference" Mixed Design of Integrated Circuits and System"(MIXDES), Gdynia, Poland, 21–23 June 2018, IEEE: Piscataway, NJ, USA, 2018, pp. 420–423.

56. Coulin, T., Detante, M., Mouchère, W., Petrillo, F. Software Architecture Metrics: A literature review. arXiv 2019, arXiv:1901.09050.
57. Soares, M.A.C., Parreiras, F.S. A literature review on question answering techniques, paradigms and systems. J. King Saud Univ.-Comput. Inf. Sci. 2020. 32, 635–646.
58. Düllmann, T.F., Heinrich, R., van Hoorn, A., Pitakrat, T., Walter, J., Willnecker, F. CASPA: A platform for comparability of architecture-based software performance engineering approaches. Proceedings of the 2017 IEEE International Conference on Software Architecture Workshops (ICSAW), Gothenburg, Sweden, 5–7 April 2017, IEEE: Piscataway, NJ, USA, 2017, pp. 294–297.
59. Walter, J., Stier, C., Koziolek, H., Kounev, S. An expandable extraction framework for architectural performance models. Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion, L'Aquila, Italy, 22–26 April 2017, pp. 165–170.
60. Singh, H. Secure SoftwareArchitecture and Design: Security Evaluation for Hybrid Approach. INROADS Int. J. Jaipur Natl. Univ. 2019, 8, 82–88.
61. Sujay, V., Reddy, M.B. Advanced Architecture-Centric Software Maintenance. i-Manager's J. Softw. Eng. 2017, 12, 1.
62. Hassan, A., Oussalah, M.C. Evolution Styles: Multi-View/Multi-Level Model for Software Architecture Evolution. *JSW* 2018, 13, 146–154.
63. Dobrica, L., Niemela, E. A survey on software architecture analysis methods. *IEEE Trans. Softw. Eng.* 2002, 28, 638–653.
64. Plauth, M., Feinbube, L., Polze, A. A performance evaluation of lightweight approaches to virtualization. Cloud Comput. 2017, 2017, 14.
65. Abrahão, S., Insfran, E. Evaluating software architecture evaluation methods: An internal replication. *Proceedings of the 21st International*

Conference on Evaluation and Assessment in Software Engineering, Karlskrona, Sweden, 15–16 June 2017, pp. 144–153.

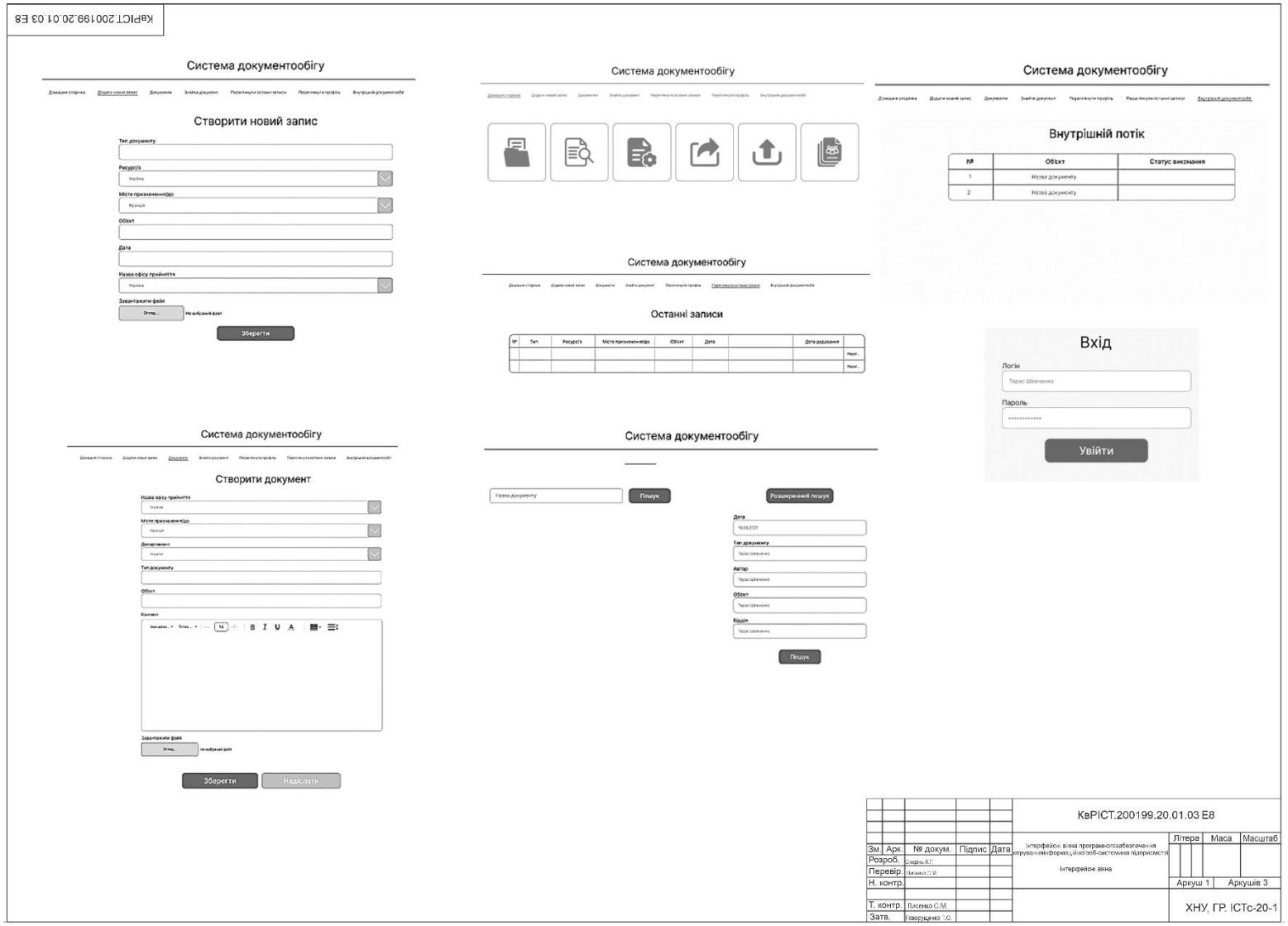
66. Alsaqaf, W., Daneva, M., Wieringa, R. Quality requirements in large-scale distributed agile projects—A systematic literature review. *Proceedings of the International Working Conference on Requirements Engineering: Foundation for Software Quality*, Essen, Germany, 27 February–2 March 2017, Springer: Berlin/Heidelberg, Germany, 2017, pp. 219–234.
67. Kasauli, R., Knauss, E., Horkoff, J., Liebel, G., Neto, F.G.d. Requirements engineering challenges and practices in large-scale agile system development. *J. Syst. Softw.* 2020. 172, 110851.
68. Zhang, D., Yu, F.R., Yang, R. A machine learning approach for software-defined vehicular ad hoc networks with trust management. Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018, *IEEE*: Piscataway, NJ, USA, 2018, pp. 1–6.
69. Cruz-Benito, J., Garcia-Penalvo, F.J., Theron, R. Analyzing the software architectures supporting HCI/HMI processes through a systematic review of the literature. *Telemat. Inform*. 2019, 38, 118–132.
70. Poularakis, K., Qin, Q., Nahum, E.M., Rio, M., Tassiulas, L. Flexible SDN control in tactical ad hoc networks. *Ad Hoc Netw*. 2019, 85, 71–80.
71. Li, X.-Y., Liu, Y., Lin, Y.-H., Xiao, L.-H., Zio, E., Kang, R. A generalized petri net-based modeling framework for service reliability evaluation and management of cloud data centers. *Reliab. Eng. Syst. Saf.* 2020. 207, 107381.
72. Varshosaz, M., Beohar, H., Mousavi, M.R. Basic behavioral models for software product lines: Revisited. *Sci. Comput. Program*. 2018, 168, 171–185.

73. Ozkaya, M. Do the informal & formal software modeling notations satisfy practitioners for software architecture modeling? Inf. Softw. Technol. 2018, 95, 15–33.
74. Bhat, M., Shumaiev, K., Hohenstein, U., Biesdorf, A., Matthes, F. The evolution of architectural decision making as a key focus area of software architecture research: A semi-systematic literature study. *Proceedings of the 2020 IEEE International Conference on Software Architecture (ICSA),* Salvador, Brazil, 16–20 March 2020. IEEE: Piscataway, NJ, USA, 2020. pp. 69–80.
75. Seifermann, S., Heinrich, R., Reussner, R. Data-driven software architecture for analyzing confidentiality. *Proceedings of the 2019 IEEE International Conference on Software Architecture (ICSA),* Hamburg, Germany, 25–29 March 2019, IEEE: Piscataway, NJ, USA, 2019, pp. 1–10.
76. Landauer, C., Bellman, K.L. An architecture for self-awareness experiments. Proceedings of the 2017 IEEE International Conference on Autonomic Computing (ICAC), Columbus, OH, USA, 17–21 July 2017, IEEE: Piscataway, NJ, USA, 2017, pp. 255–262.
77. Ferraiuolo, A., Xu, R., Zhang, D., Myers, A.C., Suh, G.E. Verification of a practical hardware security architecture through static information flow analysis. *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems,* Xi'an, China, 1–8 April 2017, pp. 555–568.
78. van Engelenburg, S., Janssen, M., Klievink, B. Design of a software architecture supporting business-to-government information sharing to improve public safety and security. *J. Intell. Inf. Syst.* 2019, 52, 595–618.
79. Bánáti, A., Kail, E., Karóczkai, K., Kozlovszky, M. Authentication and authorization orchestrator for microservice-based software architectures. *Proceedings of the 2018 41st International Convention on Information and*

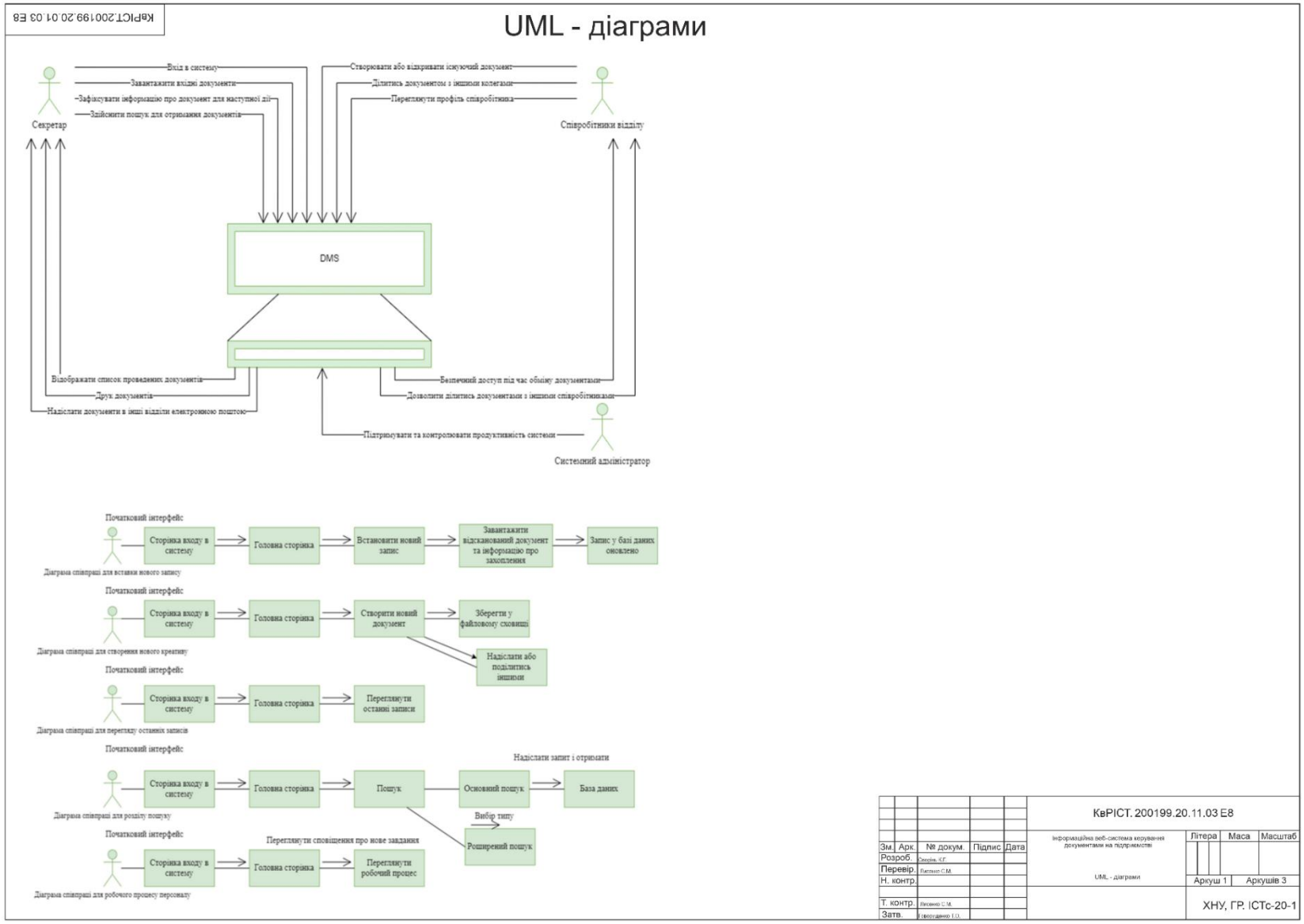
Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 21–25 May 2018, IEEE: Piscataway, NJ, USA, 2018, pp. 1180–1184.

80. Tuma, K., Scandariato, R., Balliu, M. Flaws in flows: Unveiling design flaws via information flow analysis. *Proceedings of the 2019 IEEE International Conference on Software Architecture (ICSA),* Hamburg, Germany, 25–29 March 2019, IEEE: Piscataway, NJ, USA, 2019, pp. 191–

ДОДАТОК А
(Обов 'язковий)
Копія креслення «Інтерфейсні вікна програмного забезпечення»



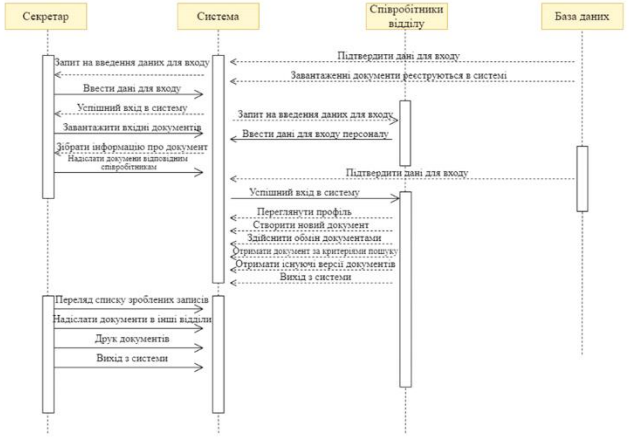
ДОДАТОК Б
(Обов 'язковий)
UML діаграми

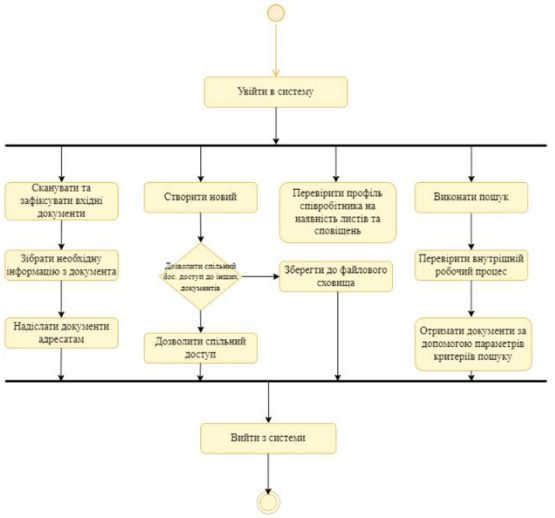


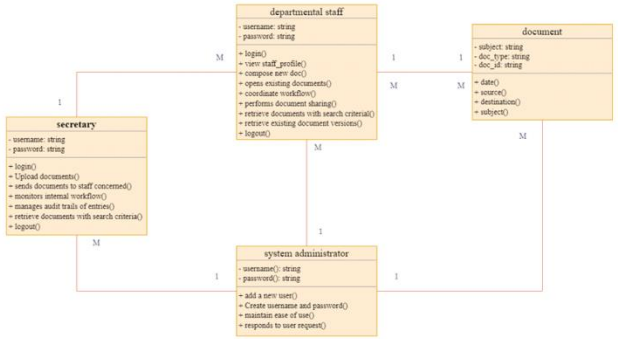
ДОДАТОК В
(Обов 'язковий)
UML діаграми

КвРІСТ. 200199.20.11.03 Е8

UML - діаграми







					КвРІСТ. 200199.20.11.03 Е8		
Зм.	Арк.	№ докум.	Підпис	Дата	Інформаційна веб-система керування документами на підприємстві	Літера	Маса	Масштаб
Розроб.		Сімоньо К.І.						
Перевір.		Лисенко С.М.			UML - діаграми	Аркуш 1	Аркушів 3	
Н. контр.								
Т. контр.		Вусенко С.М.				ХНУ, ГР. ІСТс-20-1		
Затв.		Голосуденко І.С.						

ДОДАТОК Г

(Обов`язковий)

Код програмного забезпечення інформаційної веб-системи керування документами на підприємстві

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>netcoreapp2.2</TargetFramework>
      <CodeAnalysisRuleSet>..\..\.ruleset</CodeAnalysisRuleSet>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Microsoft.Azure.Cosmos" Version="3.6.0" />
    <PackageReference Include="Microsoft.Azure.Storage.Blob" Version="11.1.3" />
    <PackageReference Include="StyleCop.Analyzers" Version="1.1.118">
      <PrivateAssets>all</PrivateAssets>
      <IncludeAssets>runtime; build; native; contentfiles; analyzers</IncludeAssets>
    </PackageReference>
    <PackageReference Include="WindowsAzure.Storage" Version="9.3.3" />
  </ItemGroup>

</Project>



<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>netcoreapp2.2</TargetFramework>
      <CodeAnalysisRuleSet>..\..\.ruleset</CodeAnalysisRuleSet>
  </PropertyGroup>

</Project>



<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>netcoreapp2.2</TargetFramework>
      <CodeAnalysisRuleSet>..\..\.ruleset</CodeAnalysisRuleSet>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="StyleCop.Analyzers" Version="1.1.118">
      <PrivateAssets>all</PrivateAssets>
      <IncludeAssets>runtime; build; native; contentfiles; analyzers</IncludeAssets>
    </PackageReference>
  </ItemGroup>

  <ItemGroup>
    <ProjectReference
Include="..\DocumentManagment.Data\DocumentManagment.DataAccess.csproj" />
    <ProjectReference
Include="..\DocumentManagment.Domain\DocumentManagment.Domain.csproj" />
  </ItemGroup>

</Project>



<Project Sdk="Microsoft.NET.Sdk.Web">
```

```
  <PropertyGroup>
    <TargetFramework>netcoreapp2.2</TargetFramework>
      <CodeAnalysisRuleSet>..\..\.ruleset</CodeAnalysisRuleSet>
    <AspNetCoreHostingModel>InProcess</AspNetCoreHostingModel>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Microsoft.AspNetCore.App" />
    <PackageReference Include="Microsoft.AspNetCore.Razor.Design" Version="2.2.0"
PrivateAssets="All" />
    <PackageReference Include="StyleCop.Analyzers" Version="1.1.118">
      <PrivateAssets>all</PrivateAssets>
      <IncludeAssets>runtime; build; native; contentfiles; analyzers</IncludeAssets>
    </PackageReference>
  </ItemGroup>

  <ItemGroup>
    <ProjectReference
Include="..\DocumentManagment.Data\DocumentManagment.DataAccess.csproj" />
    <ProjectReference
Include="..\DocumentManagment.Services\DocumentManagment.Services.csproj" />
  </ItemGroup>

</Project>

<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>netcoreapp2.2</TargetFramework>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="AutoFixture" Version="4.11.0" />
    <PackageReference Include="Moq" Version="4.13.1" />
    <PackageReference Include="xunit" Version="2.4.1" />
    <PackageReference Include="xunit.runner.visualstudio" Version="2.4.1">
      <PrivateAssets>all</PrivateAssets>
      <IncludeAssets>runtime; build; native; contentfiles; analyzers</IncludeAssets>
    </PackageReference>
  </ItemGroup>

  <ItemGroup>
    <ProjectReference
Include="..\..\src\DocumentManagment.Data\DocumentManagment.DataAccess.csproj" />
  </ItemGroup>

</Project>

<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>netcoreapp2.2</TargetFramework>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="AutoFixture" Version="4.11.0" />
    <PackageReference Include="Moq" Version="4.13.1" />
    <PackageReference Include="xunit" Version="2.4.1" />
    <PackageReference Include="xunit.runner.visualstudio" Version="2.4.1">
      <PrivateAssets>all</PrivateAssets>
      <IncludeAssets>runtime; build; native; contentfiles; analyzers</IncludeAssets>
    </PackageReference>
  </ItemGroup>
```

```
  <ItemGroup>
    <ProjectReference
Include="..\..\src\DocumentManagment.Services\DocumentManagment.Services.csproj" />
  </ItemGroup>

</Project>

<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>netcoreapp2.2</TargetFramework>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Microsoft.AspNetCore.Http.Abstractions" Version="2.2.0"
/>
    <PackageReference Include="Microsoft.AspNetCore.Mvc.Abstractions" Version="2.2.0"
/>
    <PackageReference Include="Microsoft.AspNetCore.Mvc.Core" Version="2.2.0" />
    <PackageReference Include="Moq" Version="4.13.1" />
    <PackageReference Include="xunit" Version="2.4.1" />
    <PackageReference Include="xunit.runner.visualstudio" Version="2.4.1">
      <PrivateAssets>all</PrivateAssets>
      <IncludeAssets>runtime; build; native; contentfiles; analyzers</IncludeAssets>
    </PackageReference>
  </ItemGroup>

  <ItemGroup>
    <ProjectReference
Include="..\..\src\DocumentManagment.Web\DocumentManagment.Web.csproj" />
  </ItemGroup>

</Project>

using System;
using System.Linq;
using System.Linq.Expressions;
using System.Reflection;

namespace DocumentManagment.DataAccess.Extensions
{
    public static class ListOrderExtension
    {
        public static IQueryable<TDocument> SortBy<TDocument>(this
IQueryable<TDocument> source, string orderByProperty,
            bool desc)
        {
            string command = desc ? "OrderByDescending" : "OrderBy";
            var type = typeof(TDocument);
            var property = type.GetProperty(orderByProperty, BindingFlags.IgnoreCase |
BindingFlags.Public | BindingFlags.Instance);
            var parameter = Expression.Parameter(type, "p");
            var propertyAccess = Expression.MakeMemberAccess(parameter, property);
            var orderByExpression = Expression.Lambda(propertyAccess, parameter);
            var resultExpression = Expression.Call(typeof(Queryable), command, new
Type[] { type, property.PropertyType },
                source.Expression, Expression.Quote(orderByExpression));
            return source.Provider.CreateQuery<TDocument>(resultExpression);
        }
    }
}


using System;
```

```
namespace DocumentManagment.DataAccess.Models
{
    public class Document
    {
        public Document()
        {
        }

        public Document(string userId, string name, int fileSize)
        {
            this.Id = Guid.NewGuid().ToString();
            this.UserId = userId;
            this.Name = name;
            this.FileSize = fileSize;
        }

        public string UserId { get; set; }

        public string Id { get; set; }

        public string Name { get; set; }

        public int FileSize { get; set; }
    }
}

namespace DocumentManagment.DataAccess.Models
{
    public class OrderCriteria
    {
        public OrderCriteria(string field, bool isDesc)
        {
            this.Field = field;
            this.IsDesc = isDesc;
        }

        public string Field { get; set; }

        public bool IsDesc { get; set; }
    }
}

using Microsoft.Azure.Cosmos;

namespace DocumentManagment.DataAccess.Repository
{
    public abstract class BaseRepository
    {
        protected const string DatabaseId = "document-managment";

        protected BaseRepository(CosmosClient client)
        {
            this.Client = client;
        }

        protected abstract string CollectionId { get; }

        protected CosmosClient Client { get; }
    }
}

using DocumentManagment.DataAccess.Extensions;
using DocumentManagment.DataAccess.Models;
using Microsoft.Azure.Cosmos;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Threading.Tasks;
using Microsoft.Azure.Cosmos.Linq;

namespace DocumentManagment.DataAccess.Repository
{
    public class DocumentRepository : BaseRepository, IDocumentRepository
    {
        public DocumentRepository(CosmosClient cosmosClient)
            : base(cosmosClient)
        {
        }

        protected override string CollectionId => "documents";

        public async Task<bool> DocumentExistsAsync(string userId, string id)
        {
            var container = this.Client.GetContainer(DatabaseId, this.CollectionId);

            var query = container.GetItemLinqQueryable<Document>()
                .Where(c => c.UserId == userId &&
                            c.Id == id);

            var results = await this.ExecuteQueryAsync(query);

            return results.Any();
        }

        public async Task<List<Document>> GetDocumentsAsync(string userId,
OrderCriteria criteria)
        {
            var container = this.Client.GetContainer(DatabaseId, this.CollectionId);

            var query = container.GetItemLinqQueryable<Document>(true)
                .Where(c => c.UserId == userId)
                .SortBy(criteria.Field, criteria.IsDesc);

            var results = await this.ExecuteQueryAsync(query);

            return results;
        }

        public async Task<Document> GetDocumentAsync(string partitionKey, string id)
        {
            var container = this.Client.GetContainer(DatabaseId, this.CollectionId);

            var result = await container.ReadItemAsync<Document>(id, new
PartitionKey(partitionKey));

            return result.Resource;
        }

        public async Task<Document> CreateItemAsync(Document document, string
partitionKey)
        {
            var container = this.Client.GetContainer(DatabaseId, this.CollectionId);

            var result = await container.CreateItemAsync(document, new
PartitionKey(partitionKey));

            return result.Resource;
        }

        public async Task<Document> DeleteItemAsync(string id, string partitionKey)
        {
            var container = this.Client.GetContainer(DatabaseId, this.CollectionId);
```

```
            var result = await container.DeleteItemAsync<Document>(id, new
PartitionKey(partitionKey));

            return result.Resource;
        }

        protected async Task<List<TDocument>>
ExecuteQueryAsync<TDocument>(IQueryable<TDocument> query)
        {
            var documents = new List<TDocument>();

            var feedIterator = query.ToFeedIterator();

            while (feedIterator.HasMoreResults)
            {
                var nextResults = await feedIterator.ReadNextAsync();
                documents.AddRange(nextResults);
            }

            return documents;
        }
    }
}

using DocumentManagment.DataAccess.Models;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace DocumentManagment.DataAccess.Repository
{
    public interface IDocumentRepository
    {
        Task<bool> DocumentExistsAsync(string userId, string id);

        Task<List<Document>> GetDocumentsAsync(string userId, OrderCriteria criteria);

        Task<Document> GetDocumentAsync(string partitionKey, string id);

        Task<Document> CreateItemAsync(Document document, string partitionKey);

        Task<Document> DeleteItemAsync(string id, string partitionKey);
    }
}



using Microsoft.WindowsAzure.Storage;
using Microsoft.WindowsAzure.Storage.Blob;
using System;
using System.IO;
using System.Threading.Tasks;

namespace DocumentManagment.DataAccess.Storage
{
    public class BlobStorage : IBlobStorage
    {
        private readonly CloudStorageAccount storageAccount;

        public BlobStorage(string connectionString)
        {
            this.storageAccount = CloudStorageAccount.Parse(connectionString);
        }

        public async Task<Uri> CreateContainerAsync(string containerName)
        {
```

```
            var cloudBlobContainer = this.GetContainerByName(containerName);
            await cloudBlobContainer.CreateIfNotExistsAsync().ConfigureAwait(false);

            return cloudBlobContainer?.Uri;
        }

        public async Task DeleteContainerAsync(string containerName)
        {
            var cloudBlobContainer = this.GetContainerByName(containerName);
            await cloudBlobContainer.DeleteIfExistsAsync().ConfigureAwait(false);
        }

        public async Task<bool> DeleteBlobAsync(string containerName, string location)
        {
            var container = this.GetContainerByName(containerName);
            var blob = container.GetBlockBlobReference(location);
            return await blob.DeleteIfExistsAsync();
        }

        public async Task<Uri> CreateBlobFromStreamAsync(string containerName, Stream
file, string location)
        {
            var container = this.GetContainerByName(containerName);
            var cloudBlockBlob = container.GetBlockBlobReference(location);
            await cloudBlockBlob.UploadFromStreamAsync(file).ConfigureAwait(false);
            return cloudBlockBlob.Uri;
        }

        public Task<Stream> GetContentOfBlobFromContainerAsync(string containerName,
string blobName)
        {
            var container = this.GetContainerByName(containerName);
            var cloudBlockBlob = container.GetBlockBlobReference(blobName);
            return cloudBlockBlob.OpenReadAsync();
        }

        private CloudBlobClient GetBlobClient()
        {
            return this.storageAccount.CreateCloudBlobClient();
        }

        private CloudBlobContainer GetContainerByName(string name)
        {
            var cloudBlobClient = this.GetBlobClient();
            return cloudBlobClient.GetContainerReference(name);
        }
    }
}



using System;
using System.IO;
using System.Threading.Tasks;

namespace DocumentManagment.DataAccess.Storage
{
    public interface IBlobStorage
    {
        Task<Uri> CreateContainerAsync(string containerName);

        Task DeleteContainerAsync(string containerName);

        Task<bool> DeleteBlobAsync(string containerName, string location);
```

```
        Task<Uri> CreateBlobFromStreamAsync(string containerName, Stream file, string
location);

        Task<Stream> GetContentOfBlobFromContainerAsync(string containerName, string
blobName);
    }
}



namespace DocumentManagment.Domain.Model
{
    public class User
    {
        public string Id { get; set; }
    }
}

using DocumentManagment.Domain.Model;

namespace DocumentManagment.Services.Providers
{
    public interface IUserProvider
    {
        User User { get; }
    }
}

using DocumentManagment.Domain.Model;

namespace DocumentManagment.Services.Providers
{
    public class UserProvider : IUserProvider
    {
        public User User => new User { Id = "123" };
    }
}

using DocumentManagment.DataAccess.Models;
using DocumentManagment.DataAccess.Repository;
using DocumentManagment.DataAccess.Storage;
using DocumentManagment.Services.Providers;
using System.Collections.Generic;
using System.IO;
using System.Threading.Tasks;

namespace DocumentManagment.Services
{
    public class DocumentService : IDocumentService
    {
        private readonly IBlobStorage blobStorage;
        private readonly IDocumentRepository documentRepository;
        private readonly IUserProvider userProvider;

        public DocumentService(
            IBlobStorage blobStorage,
            IDocumentRepository documentRepository,
            IUserProvider userProvider)
        {
            this.blobStorage = blobStorage;
            this.documentRepository = documentRepository;
            this.userProvider = userProvider;
        }

        public async Task<IEnumerable<Document>> GetDocumentsAsync(OrderCriteria
criteria)
```

```
        {
            var userId = this.userProvider.User.Id;
            var documents = await this.documentRepository.GetDocumentsAsync(userId,
criteria);

            return documents;
        }

        public async Task<Document> SaveDocumentAsync(Stream file, string fileName,
int size)
        {
            var userId = this.userProvider.User.Id;
            var doc = new Document(userId, fileName, size);

            await this.blobStorage.CreateContainerAsync(userId);
            await this.blobStorage.CreateBlobFromStreamAsync(userId, file, doc.Id);

            var result = await this.documentRepository.CreateItemAsync(doc, userId);

            return result;
        }

        public async Task<Document> GetDocumentAsync(string id)
        {
            var userId = this.userProvider.User.Id;
            var document = await this.documentRepository.GetDocumentAsync(userId, id);
            return document;
        }

        public Task<Stream> LoadDocumentStreamAsync(string location)
        {
            var userId = this.userProvider.User.Id;
            return this.blobStorage.GetContentOfBlobFromContainerAsync(userId,
location);
        }

        public async Task<bool> DeleteDocumentAsync(string id)
        {
            var userId = this.userProvider.User.Id;
            var isDocumentExist = await
this.documentRepository.DocumentExistsAsync(userId, id);

            if (!isDocumentExist)
            {
                return false;
            }

            await this.documentRepository.DeleteItemAsync(id, userId);
            await this.blobStorage.DeleteBlobAsync(userId, id);

            return true;
        }
    }
}

using DocumentManagment.DataAccess.Models;
using System.Collections.Generic;
using System.IO;
using System.Threading.Tasks;
using DocumentManagment.Domain.Model;

namespace DocumentManagment.Services
{
    public interface IDocumentService
    {
        Task<IEnumerable<Document>> GetDocumentsAsync(OrderCriteria criteria);
```

```
        Task<Document> SaveDocumentAsync(Stream file, string fileName, int size);

        Task<Document> GetDocumentAsync(string id);

        Task<Stream> LoadDocumentStreamAsync(string location);

        Task<bool> DeleteDocumentAsync(string id);
    }
}

using AutoFixture;
using DocumentManagment.DataAccess.Models;
using DocumentManagment.DataAccess.Repository;
using Microsoft.Azure.Cosmos;
using Moq;
using System.Threading;
using System.Threading.Tasks;
using Xunit;

namespace DocumentManagment.DataAccess.Tests.Unit
{
    public class DocumentRepositoryTests
    {
        private readonly DocumentRepository _documentRepository;
        private readonly Mock<CosmosClient> _cosmosClientMock;
        private readonly Mock<Container> _containerMock;
        private readonly Mock<ItemResponse<Document>> _responseMock;

        public DocumentRepositoryTests()
        {
            _cosmosClientMock = new Mock<CosmosClient>(MockBehavior.Strict);
            _containerMock = new Mock<Container>();
            _documentRepository = new DocumentRepository(_cosmosClientMock.Object);
            _responseMock = new Mock<ItemResponse<Document>>();
        }

        [Fact]
        public async Task GetDocumentTask_ReturnsCorrectResult()
        {
            // Arrange
            var fixture = new Fixture();
            var userId = fixture.Create<string>();
            var id = fixture.Create<string>();
            var size = fixture.Create<int>();

            var expectedDocument = fixture.Build<Document>()
                .With(x => x.UserId, userId)
                .With(x => x.Id, id)
                .With(x => x.FileSize, size)
                .Create();

            _responseMock.Setup(p => p.Resource).Returns(expectedDocument);

            _containerMock.Setup(m =>
                    m.ReadItemAsync<Document>(id, new PartitionKey(userId), null,
default(CancellationToken)))
                .Returns(Task.FromResult(_responseMock.Object));

            _cosmosClientMock.Setup(
                    x => x.GetContainer("document-managment", "documents"))
                .Returns(_containerMock.Object);

            //Act
            var actualResult = await _documentRepository.GetDocumentAsync(userId, id);
```

```
            //Assert
            Assert.Equal(userId, actualResult.UserId);
        }
    }
}

using AutoFixture;
using DocumentManagment.DataAccess.Models;
using DocumentManagment.DataAccess.Repository;
using DocumentManagment.DataAccess.Storage;
using DocumentManagment.Domain.Model;
using DocumentManagment.Services.Providers;
using Moq;
using System;
using System.IO;
using System.Threading.Tasks;
using Xunit;

namespace DocumentManagment.Services.Tests.Unit
{
    public class DocumentServiceTests
    {
        private readonly Mock<IBlobStorage> _blobStorageMock;
        private readonly Mock<IDocumentRepository> _documentRepositoryMock;
        private readonly Mock<IUserProvider> _userProviderMock;

        public DocumentServiceTests()
        {
            _blobStorageMock = new Mock<IBlobStorage>();
            _documentRepositoryMock = new Mock<IDocumentRepository>();
            _userProviderMock = new Mock<IUserProvider>();
        }

        [Fact]
        public async Task GetDocumentAsync_ReturnsCorrectResult()
        {
            //Assert
            var expectedDocument = CreateDocument();

            _userProviderMock.Setup(p => p.User).Returns(new User { Id =
expectedDocument.UserId });

            _documentRepositoryMock.Setup(m =>
m.GetDocumentAsync(expectedDocument.UserId, expectedDocument.Id))
                .Returns(Task.FromResult(expectedDocument));
            var documentService = CreateService();

            //Act
            var actualDocument = await
documentService.GetDocumentAsync(expectedDocument.Id);

            //Arrange
            Assert.Equal(expectedDocument.Id, actualDocument.Id);
        }

        [Fact]
        public async Task SaveDocumentAsync_ReturnsCorrectResult()
        {
            //Assert
            var expectedDocument = CreateDocument();

            _userProviderMock.Setup(p => p.User).Returns(new User { Id =
expectedDocument.UserId });

            _blobStorageMock.Setup(m =>
m.CreateContainerAsync(expectedDocument.UserId));
```

```
            _blobStorageMock.Setup(m =>
                m.CreateBlobFromStreamAsync(It.IsAny<string>(), It.IsAny<Stream>(),
expectedDocument.Id));

            _documentRepositoryMock.Setup(m => m.CreateItemAsync(It.IsAny<Document>(),
expectedDocument.UserId))
                .Returns(Task.FromResult(expectedDocument));

            var documentService = CreateService();

            //Act
            var actualDocument =
                await documentService.SaveDocumentAsync(null, expectedDocument.Name,
expectedDocument.FileSize);

            //Arrange
            Assert.Equal(expectedDocument.Id, actualDocument.Id);
        }

        [Fact]
        public async Task
DeleteDocumentAsync_ReturnsCorrectResult_WhenDocumentExists()
        {
            //Assert
            var expectedDocument = CreateDocument();

            _userProviderMock.Setup(p => p.User).Returns(new User { Id =
expectedDocument.UserId });

            _blobStorageMock.Setup(m => m.DeleteBlobAsync(It.IsAny<string>(),
expectedDocument.Id));

            _documentRepositoryMock.Setup(m => m.DeleteItemAsync(expectedDocument.Id,
expectedDocument.UserId));
            _documentRepositoryMock.Setup(m =>
m.DocumentExistsAsync(expectedDocument.UserId, expectedDocument.Id))
                .Returns(Task.FromResult(true));
            var documentService = CreateService();

            //Act
            var actualResult =
                await documentService.DeleteDocumentAsync(expectedDocument.Id);

            //Arrange
            Assert.True(actualResult);
        }

        [Fact]
        public async Task
DeleteDocumentAsync_ReturnsCorrectResult_WhenDocumentDoseNotExists()
        {
            //Assert
            var expectedDocument = CreateDocument();

            _userProviderMock.Setup(p => p.User).Returns(new User { Id =
expectedDocument.UserId });

            _blobStorageMock.Setup(m => m.DeleteBlobAsync(It.IsAny<string>(),
expectedDocument.Id));

            _documentRepositoryMock.Setup(m => m.DeleteItemAsync(expectedDocument.Id,
expectedDocument.UserId));
            _documentRepositoryMock.Setup(m =>
m.DocumentExistsAsync(expectedDocument.UserId, expectedDocument.Id))
                .Returns(Task.FromResult(true));
            var documentService = CreateService();
```

```
            //Act
            var actualResult =
                await documentService.DeleteDocumentAsync(Guid.NewGuid().ToString());

            //Arrange
            Assert.False(actualResult);
        }

        private static Document CreateDocument()
        {
            var fixture = new Fixture();
            var userId = fixture.Create<string>();
            var id = fixture.Create<string>();
            var size = fixture.Create<int>();

            return fixture.Build<Document>()
                .With(x => x.UserId, userId)
                .With(x => x.Id, id)
                .With(x => x.FileSize, size)
                .Create();
        }

        private IDocumentService CreateService()
        {
            return new DocumentService(_blobStorageMock.Object,
_documentRepositoryMock.Object,
                _userProviderMock.Object);
        }
    }
}



using DocumentManagment.Web.ModelBinders;
using DocumentManagment.Web.Models;
using Microsoft.AspNetCore.Mvc.ModelBinding;
using Moq;
using System.Threading.Tasks;
using Xunit;

namespace DocumentManagment.Web.Tests.Unit
{
    public class OrderCriteriaModelBinderTest
    {
        private readonly Mock<IValueProvider> _mockValueProvider = new
Mock<IValueProvider>();

        [Fact]
        public async Task BindModelAsync_DataFormatIsValid_ContextResultModelIsOrder()
        {
            // Arrange
            _mockValueProvider.Setup(m => m.GetValue("order")).Returns(new
ValueProviderResult("Location.desc"));

            var context = CreateModelBindingContext(_mockValueProvider.Object);
            var binder = new OrderCriteriaModelBinder();

            // Act
            await binder.BindModelAsync(context);

            // Assert
            var resultModel = context.Result.Model as Order;

            Assert.Equal("Location", resultModel?.Field);
            Assert.Equal(true, resultModel?.IsDesc);
```

```
        }

        [Fact]
        public async Task
BindModelAsync_DataFormatIsEmpty_ContextResultModelHasDefaulOrder()
        {
            // Arrange
            _mockValueProvider.Setup(m => m.GetValue("order")).Returns(new
ValueProviderResult(""));

            var context = CreateModelBindingContext(_mockValueProvider.Object);
            var binder = new OrderCriteriaModelBinder();

            // Act
            await binder.BindModelAsync(context);

            // Assert
            var resultModel = context.Result.Model as Order;

            Assert.Equal("Name", resultModel?.Field);
            Assert.Equal(false, resultModel?.IsDesc);
        }

        [Fact]
        public async Task
BindModelAsync_DataFormatIsInvalid_ContextResultModelIsNull()
        {
            // Arrange
            _mockValueProvider.Setup(m => m.GetValue("order")).Returns(new
ValueProviderResult("size"));

            var context = CreateModelBindingContext(_mockValueProvider.Object);
            var binder = new OrderCriteriaModelBinder();

            // Act
            await binder.BindModelAsync(context);

            // Assert
            var resultModel = context.Result.Model as Order;
            Assert.Null(resultModel);
        }

        private static ModelBindingContext CreateModelBindingContext(IValueProvider
valueProvider)
        {
            return new DefaultModelBindingContext {ValueProvider = valueProvider};
        }
    }
}
```

UNICHECK

Ім'я користувача:
Кафедра КІ

ID перевірки:
1015311092

Дата перевірки:
30.05.2023 09:50:32 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
30.05.2023 10:14:55 EEST

ID користувача:
100005591

Назва документа: **Сворінь_Інформаційна веб система керування документами на підприємстві**

Кількість сторінок: **77** Кількість слів: **11904** Кількість символів: **97518** Розмір файлу: **987.01 KB** ID файлу: **1014982266**

3.2%
Схожість

Найбільша схожість: **0.9% з джерелом з Бібліотеки** (ID файлу: **1014981430**)

2.43% Джерела з Інтернету	41	Сторінка 79
2.52% Джерела з Бібліотеки	62	Сторінка 79

0% Цитат

Цитати	1	Сторінка 80
Посилання	1	Сторінка 80

0%
Вилучень

Немає вилучених джерел

Tue May 30 08:47:52 EEST 2023, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 7%**

ID: 114269 Назва: БКР Інформаційна веб система керування документами на підприємстві Додано в БД: 2023-05-30 Автора: Сворінь К.Г. Керівники: С.М. Лисенко Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	67355	1046	1116 (2%)	12 (1%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Сворінь Кирил Геннадійович

Тема: Інформаційна веб система керування документами на підприємстві

Спеціальність: 126 «Інформаційні системи та технології»

Обсяг кваліфікаційної роботи:

Кількість листів креслень ___3___ Кількість сторінок записки ___58___

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є створення інформаційної веб-системи керування документами на підприємстві.

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 проведено аналіз відомих інформаційних веб-систем керування документами показав їх наявні переваги та недоліки.

Зокрема було зроблено висновок, що інформаційної системи керування документами можуть різнитися залежно від конкретних потреб та контексту організації

В другому розділі подано користувальницькі вимоги, зокрема, описано механізм іменування файлів, механізм отримання документів, забезпечення контролю версій документів, процес керування документами.

В розділ описано основні аспекти застосування методології Agile при проєктуванні ІС.

Описано процес забезпечення процедури збору даних та забезпечення процедури перегляду результатів активності документообігу співробітниками.

В розділі також надано нефункційні вимоги до інформаційної системи та виконано аналіз вимого до інформаційної системи.

Подано процес та аспекти проектування інформаційної веб-системи керування документами.

В третьому розділі здійснено проєктування архітектури інформаційної технології, зокрема розроблено UML-діаграми для інформаційної веб-системи керування документами. Здійснено програмну реалізацію інформаційної веб-системи керування документами. Розроблено інтерфейс користувача інформаційної веб-системи керування документами.

4. Позитивні сторони роботи: висока практична цінність роботи.

5. Негативні сторони роботи: розробка ПЗ була проведена з недостатньою увагою до процесу тестування.

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

7. Відгук про роботу в цілому: Робота виконана на належному науково-технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: задовільно, 3.25 D

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

доцент кафедри ІПЗ, к.т.н., доц. Гурман І.В.

_____

"30" травня 2023 р. _____ (підпис)

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

Своріня Кирила Геннадійовича
ПІБ здобувача вищої освіти

ФІТ, 3 курсу, групи ІСТс-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.

22 квітня 2023 року

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 веб-система керування документами на підприємстві

Автор: Сворінь Кирил Геннадійович

Спеціальність: 126– Інформаційні системи та технології

Освітня програма: освітньо-професійна

Науковий керівник: Лисенко Сергій Миколайович, д.т.н, професор

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, які були виявлені в роботі, є легальними і не можуть бути вважається плагіатом, оскільки:

1) Запозичення, які були включені в розділи про аналіз існуючих аналогів та прототипів, не стосуються прямо авторського дослідження і не відносяться до результатів проведених досліджень;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
3) окремі знайдені співпадіння є загальновживаними фразами або виразами, що підтверджується посиланням системи на збіг з 10-40 джерелами для одного фрагмента речення;
4) система виявила в деяких місцях запозичення послідовностей чотирьохрозрядних двійкових кодів, які є вхідними даними для багатьох різних задач. Ці послідовності не можуть бути розглянуті як об'єкт авторських прав і, відповідно, їх використання не порушує авторські права;
5) усі виявлені системою ознаки модифікації тексту відносяться до поєднання латинських символів з україномовними скороченнями індексів у формулах. Це не є зміною тексту.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 2.43% з інтернету та 2.52% з бібліотеки і адресується до 80 першоджерела, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ С. М. Лисенко

Гарант ОП _____ Є.Г. Гнатчук

Завідувач кафедри КІІС _____ Т. О. Говорущенко