МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами

Кафедра програмного забезпечення автоматизованих систем

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»

Завідувач кафедри

_  __ проф. Приходько С.Б.

«___» _____ 20___ р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «бакалавр»

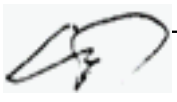
на тему: Розробка програмного забезпечення адаптивної веб-орієнтованої системи електронного документообігу

Виконав: студент групи 4157зст3

 Терещенко В.С.

(підпис, ПІБ)

Керівник роботи:

 проф. Приходько С.Б.

(посада, науковий ступень, вчене звання)

(підпис, ПІБ)

Миколаїв – 2026 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проєктами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____ доц. Макарова Л.М.
(підпис)
«___» _____ 20___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «бакалавр»

Студенту Терещенко Віолетта Станіславівна
(Прізвище, ім'я, по батькові)

1. Тема роботи: Розробка програмного забезпечення адаптивної веб-орієнтованої системи електронного документообігу
Керівник роботи проф. Приходько С.Б.
Затверджені наказом ректора №275-уч від 07.04.2026 р.
2. Термін подання роботи: 01.06.2026 р.
3. Вихідні дані по роботі: _____
_____

4. Перелік питань, що належать до розробки (найменування розділів) Титульний аркуш; Завдання на кваліфікаційну роботу; Анотація (українською та англійською мовами); Зміст; Вступ; Аналіз практичної задачі інженерії програмного забезпечення; Проєкт програмного забезпечення; Результати розробки програмного забезпечення; Розділ з охорони праці; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма та методика випробувань програмного забезпечення).
5. Перелік презентаційних матеріалів 1. Слайд «Актуальність теми та мета роботи». 2. Слайд «Аналіз існуючих систем електронного документообігу (M-Files, DocuWare, LogicalDOC)». 3. Слайд «Функціональні можливості розробленої системи». 4. Слайд «Технології та програмні засоби реалізації». 5. Слайд «Діаграма варіантів використання системи». 6. Слайд «Загальна

архітектура системи». 7. Слайд «Фізична модель бази даних». 8. Слайд «Діаграма послідовності процесу авторизації користувача». 9. Слайд «Форми реєстрації та авторизації користувачів». 10. Слайд «Сторінки для роботи з документами». 11. Слайд «Сторінки завантаження нової версії документа та керування доступом». 12. Слайд «Інформаційна панель системи». 13. Слайд «Сторінки звітності». 14. Слайд «Основні результати роботи та висновки».

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1			
2			
3			
4			

7. Дата видачі завдання 07.04.2026 р.

КАЛЕНДАРНИЙ ПЛАН

Номер	Назва етапів роботи	Терміни виконання	Примітка
1	Підготовка розділу «ВСТУП»	30.03.2026	
2	Аналіз практичної задачі інженерії ПЗ	06.04.2026	
3	Аналіз вимог до ПЗ	13.04.2026	
4	Розробка архітектури ПЗ	20.04.2026	
5	Розробка проєкту ПЗ	04.05.2026	
6	Реалізація та тестування ПЗ	11.05.2026	
7	Підготовка розділу «Результати розробки ПЗ»	15.05.2026	
8	Підготовка розділу з охорони праці	18.05.2026	
9	Підготовка розділу «ВИСНОВКИ»	22.05.2026	
10	Оформлення списку використаних джерел та додатків	25.05.2026	
11	Подання на кафедру ПЗАС тексту остаточного варіанта роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версій роботи (дод. 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів / ідентичності / схожості з використанням програмно-технічних засобів», який введений у дію наказом ректора НУК за № 20 від 20.01.2020 р.)	01.06.2026	не пізніше ніж за 14 діб до захисту (згідно з п.4.1 зазначеного Порядку)
12	Підготовка презентації та доповіді	05.06.2026	
13	Попередній захист роботи на засіданні кафедри ПЗАС	08.06.2026	
14	Подання на кафедру ПЗАС електронних версій наступних документів у форматі pdf: кваліфікаційної роботи; файла-опису кваліфікаційної роботи (згідно з Додатком до наказу ректора НУК за № 287-уч від	15.06.2026	

	19.05.2020 р.); презентації доповіді, а також Авторського договору з додатком		

* За результатами переддипломної практики (ПП), яка була з 02.02.2026 по 22.03.2026 р.

Студент _____ Терещенко В.С.
(підпис) (ПІБ)

Керівник роботи _____ проф. Приходько С.Б.
(підпис) (ПІБ)

АНОТАЦІЯ

Кваліфікаційна робота присвячена розробленню адаптивної веб-орієнтованої системи електронного документообігу, призначеної для централізованого зберігання, оброблення, пошуку, зміни версії та контролю використання електронних документів. Метою роботи є розроблення адаптивної веб-орієнтованої системи електронного документообігу.

Для досягнення поставленої мети було виконано аналіз предметної області електронного документообігу та існуючих програмних рішень, сформовано функціональні й нефункціональні вимоги до системи, розроблено архітектурну модель програмного забезпечення, спроєктовано структуру бази даних і реалізовано основні модулі системи. Також проведено тестування розробленого програмного забезпечення з метою перевірки його відповідності поставленим вимогам.

Під час виконання роботи використано методи аналізу предметної області, порівняльного аналізу програмних рішень, об’єктно-орієнтованого проєктування, моделювання структури даних, функціонального тестування та оцінювання результатів реалізації. Програмне забезпечення реалізовано з використанням мови програмування Python, фреймворку Django, бази даних SQLite3, Django ORM, HTML5, CSS3, Bootstrap 5 та JavaScript.

У результаті роботи розроблено веб-орієнтовану систему EDocFlow. Проведене тестування підтвердило коректність роботи основних функціональних сценаріїв і відповідність системи визначеним вимогам.

Кваліфікаційна робота містить 106 сторінок, 30 рисунків, 6 таблиць, 22 використаних джерел та 5 додатків.

Ключові слова: електронний документообіг, веб-орієнтована система, Django, документи, версії документів, права доступу, журнал дій, дашборд, звітність.

ANNOTATION

The qualification work is devoted to the development of an adaptive web-oriented electronic document management system intended for centralized storage, processing, search, version management, and control over the use of electronic documents. The aim of the work is to develop an adaptive web-oriented electronic document management system.

To achieve this aim, an analysis of the subject area of electronic document management and existing software solutions was carried out, functional and non-functional system requirements were formulated, the software architecture model was developed, the database structure was designed, and the main system modules were implemented. The developed software was also tested to verify its compliance with the specified requirements.

During the work, methods of subject area analysis, comparative analysis of software solutions, object-oriented design, data structure modeling, functional testing, and evaluation of implementation results were used. The software was implemented using the Python programming language, the Django framework, the SQLite3 database, Django ORM, HTML5, CSS3, Bootstrap 5, and JavaScript.

As a result of the work, the web-oriented EDocFlow system was developed. The conducted testing confirmed the correctness of the main functional scenarios and the compliance of the system with the defined requirements.

The qualification work consists of 106 pages, 30 figures, 6 tables, 22 references, and 5 appendices.

Keywords: electronic document management, web-oriented system, Django, documents, document versions, access rights, action log, dashboard, reporting.

ЗМІСТ

ВСТУП

Сучасний етап розвитку інформаційного суспільства характеризується інтенсивною цифровою трансформацією управлінських процесів, у межах якої особливого значення набуває переведення документообігу в електронну форму. Згідно з аналітичними дослідженнями у сфері цифровізації, впровадження електронних інформаційних систем дозволяє суттєво підвищити якість управлінських рішень, зменшити витрати часу на оброблення інформації та оптимізувати внутрішні бізнес-процеси організацій [1]. Водночас зростання обсягів цифрових даних і кількості інформаційних потоків формує нові вимоги до систем їх оброблення, структуризації та контролю.

У наукових працях, присвячених цифровій трансформації економіки, зазначається, що використання інформаційно-аналітичних систем стає ключовим чинником підвищення конкурентоспроможності організацій. Зокрема, сучасні дослідження підтверджують, що автоматизація управлінських процесів, включаючи документообіг, сприяє скороченню операційних витрат і підвищенню продуктивності діяльності підприємств у середньому на 20–30% залежно від рівня цифрової зрілості [2]. Це обумовлює необхідність розроблення ефективних програмних рішень, здатних забезпечити централізоване керування документами та їх життєвим циклом.

Актуальність досліджуваної проблематики також підтверджується державними та міжнародними ініціативами у сфері цифровізації. В Україні, зокрема, активно розвиваються електронні реєстри та інформаційні системи, які забезпечують доступ до даних у режимі електронної взаємодії між суб’єктами [3]. Це супроводжується зростанням обсягів електронної документації та необхідністю її систематизованого зберігання, контролю доступу та забезпечення цілісності. У контексті післявоєнного відновлення держави цифрові системи управління документами відіграють важливу роль у забезпеченні прозорості процесів, доступності інформації та ефективного управління ресурсами. Окрім того, результати сучасних досліджень у галузі інформаційних технологій свідчать про

стрімке зростання інтересу до цифрової трансформації як наукового та практичного напряму. Аналіз публікацій за останні роки демонструє суттєве збільшення кількості досліджень, присвячених впливу цифрових технологій на управління організаціями, що вказує на актуальність і перспективність цього напряму [4]. Особливе місце в цих процесах займають системи електронного документообігу як базовий інструмент організації інформаційних потоків.

Таким чином, необхідність розробки адаптивних веб-орієнтованих систем електронного документообігу обумовлена сукупністю факторів: зростанням обсягів інформації, потребою у підвищенні швидкості її оброблення, вимогами до безпеки та контролю доступу, а також загальними тенденціями цифровізації економіки та суспільства. Це визначає практичну значущість і наукову актуальність дослідження у напрямі розроблення сучасних програмних засобів управління електронними документами.

Метою роботи є розроблення адаптивної веб-орієнтованої системи електронного документообігу.

Для досягнення мети необхідно виконати наступні задачі:

− виконати аналіз предметної області електронного документообігу та існуючих програмних рішень із метою формалізації вимог до системи;

− сформувати функціональні та нефункціональні вимоги і розробити архітектурну модель програмного забезпечення;

− здійснити проєктування структури системи, бази даних та реалізувати програмне забезпечення із використанням обраного технологічного стеку;

− провести тестування системи та оцінити її характеристики з метою перевірки відповідності поставленим вимогам.

1 АНАЛІЗ ЗАДАЧІ РОЗРОБКИ АДАПТИВНОЇ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ ЕЛЕКТРОННОГО ДОКУМЕНТООБІГУ

1.1 Аналіз предметної області електронного документообігу та формалізація задачі

У сучасних організаціях документи вже не є лише носіями інформації або формальними результатами діловодства. Вони фіксують управлінські рішення, координують бізнес-процеси, підтверджують виконані дії та формують основу для подальшого аналізу. Тому системи електронного документообігу доцільно розглядати не як прості архівні сервіси, а як важливу частину інформаційної інфраструктури організації, що забезпечує створення, погодження, підписання, передавання, зберігання, пошук і контроль виконання документів.

Правове ядро цієї предметної області визначається тим, що електронний документообіг охоплює створення, передавання, одержання, зберігання, оброблення, використання та знищення електронних документів. Електронний документ не можна зводити лише до файлу, оскільки він має юридичний статус, реквізити, життєвий цикл, вимоги до цілісності, автентичності, доступності, перевірки підпису та збереження історії змін. Саме тому електронний документообіг перебуває на перетині документознавства, інформаційних систем, управління бізнес-процесами, кібербезпеки та правового регулювання [5].

У діяльності організації документ виконує оперативну функцію, коли використовується в поточній роботі, і доказову функцію, коли підтверджує певний факт, дію або рішення. Дослідження показують, що результативність впровадження ERMS залежить не лише від технічних можливостей, а й від організаційної культури, підтримки керівництва, ресурсів, навчання персоналу та державної політики [6]. Отже, така система має підтримувати як щоденну роботу з документами, так і їх довгострокову надійність, контрольованість та відповідність установленим вимогам.

Наукові праці щодо EDMS визначають базове функціональне ядро таких систем: автоматизоване захоплення інформації, пошук документів, засоби

співпраці, індексування, класифікацію, зберігання, архівування, робочі потоки, механізми безпеки, аудит і підтримку комплаєнсу [7]. Це свідчить, що електронний документообіг не можна обмежувати лише завантаженням файлів або веденням електронної картотеки.

Формалізація взаємозв'язків між компонентами системи потребує розгляду життєвого циклу документа як послідовності станів і переходів. Такий підхід дозволяє поєднати окремі функції в єдиний процес, де кожна операція з документом має визначене місце в організаційній роботі.

Узагальнену схему життєвого циклу документа подано як структуровану модель, у якій етап формування або реєстрації супроводжується визначенням основних атрибутів і подальшою класифікацією для впорядкованого доступу та оброблення документа (рис. 1.1).

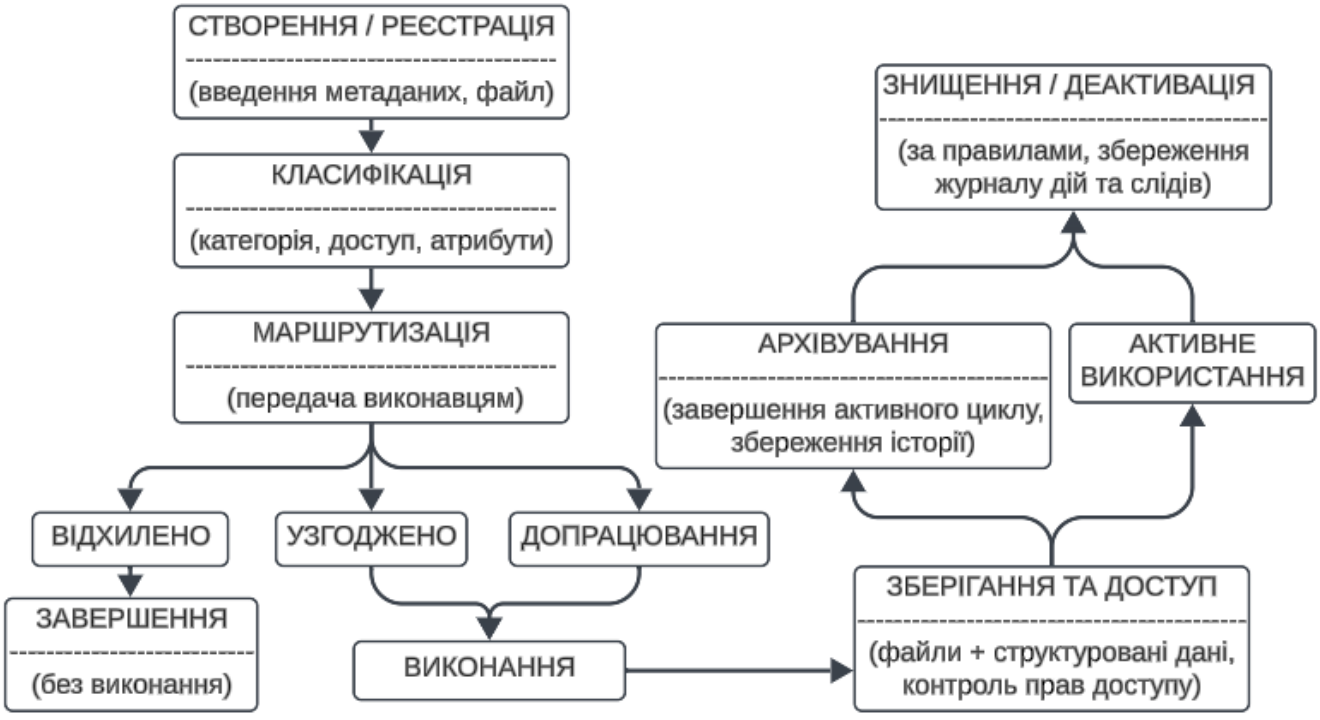


Рисунок 1.1 – Узагальнена схема життєвого циклу документа

Подальший етап життєвого циклу документа пов'язаний із маршрутизацією, яка забезпечує його передавання між виконавцями, погодження, повернення на доопрацювання або відхилення. Після досягнення узгодженого стану документ переходить до виконання, де здійснюються його обробка, оновлення версій, фіксація змін і збереження в системі з урахуванням файлової та атрибутивної

складових. Наступні стадії охоплюють використання документа в операційній та аналітичній діяльності через пошук, фільтрацію й повторне використання інформації. Завершення життєвого циклу відбувається шляхом архівування або деактивації із збереженням журналу подій для подальшого аналізу й аудиту.

У сучасних дослідженнях адаптивність веб-систем розглядається ширше, ніж звичайне пристосування інтерфейсу до різних екранів. Хоча адаптивний дизайн залишається базовою вимогою, емпіричні дослідження підтверджують його позитивний вплив на зручність і результативність використання вебсайтів, особливо з огляду на активне використання смартфонів [8]. Для систем електронного документообігу це означає, що погодження, перегляд статусів, контроль строків і робота з документами мають бути зручними не лише на настільних пристроях, а й у мобільному середовищі.

Водночас адаптивність такої системи має охоплювати доступність, контекстну чутливість і поведінкову гнучкість. Інтерфейс повинен враховувати контрастність, фокус-навігацію, керування з клавіатури, зрозумілість форм і повідомлень про помилки. Крім того, різні ролі користувачів потребують різного інформаційного навантаження, сценаріїв переходів і пріоритетів елементів. Саме такий підхід відображено в дослідженнях, де адаптація інтерфейсу виконується на основі контексту та взаємодій користувача із застосуванням рекомендаційних механізмів і server-driven UI [9].

Зв’язок адаптивності з персоналізацією та доступністю посилюється завдяки використанню машинного навчання. Систематичний огляд 2025 року, що охопив 57 досліджень, виявив 24 роботи з ML-рішеннями для адаптивних інтерфейсів [10]. Це підтверджує, що персоналізована й інклюзивна взаємодія стала окремим напрямом сучасних досліджень. Для електронного документообігу це важливо, оскільки адаптивність може бути не декоративною властивістю інтерфейсу, а архітектурною характеристикою програмного продукту.

Для системи електронного документообігу адаптивність доцільно розглядати у трьох площинах. Технічна адаптивність забезпечує коректну роботу на різних пристроях і в різних браузерах. Функціональна адаптивність передбачає зміну

набору інструментів залежно від ролі, повноважень і поточного стану документа. Когнітивна адаптивність спрямована на зменшення інформаційного перевантаження, спрощення навігації та підтримку швидкого прийняття рішень. У такому розумінні система електронного документообігу є не просто цифровим архівом, а організованим робочим середовищем.

Оскільки системи електронного документообігу працюють із договорами, кадровими документами, фінансовими матеріалами, службовим листуванням і звітами, безпека є обов'язковою складовою їх проєктування. За даними CERT-UA, у 2024 році було опрацьовано 4315 кіберінцидентів, що майже на 70% більше, ніж роком раніше; водночас сектор публічної адміністрації залишається одним із найбільш привабливих для атак через критичність сервісів і концентрацію даних [11]. Отже, така система має створюватися з урахуванням постійної наявності ризику.

Це узгоджується з підходами NIST CSF 2.0, де кібербезпека розглядається як сукупність результатів, придатних для організацій різного масштабу, а зрілість практик змінюється від фрагментарних до адаптивних [12]. Для веб-орієнтованої системи документообігу це означає необхідність вбудування безпеки в усі етапи: проєктування, розроблення, розгортання, експлуатацію, виявлення подій, реагування на інциденти та відновлення. Практичним продовженням є підхід zero trust, за якого доступ має залишатися контрольованим незалежно від пристрою, мережі чи середовища.

Для такої системи недостатньо лише класичного role-based доступу. У реальних сценаріях права мають залежати від типу документа, статусу, підрозділу, рівня конфіденційності, часу, мережевого контексту, електронного підпису та участі користувача в маршруті погодження. Сучасні дослідження показують, що для relational і NoSQL-середовищ потрібні складніші моделі авторизації, ніж традиційний розподіл за ролями [13]. Тому доцільним є поєднання ролей, атрибутів користувача та динамічного оцінювання довіри.

Окреме значення мають механізми електронного підпису. Сучасні сервіси дають змогу підписувати документ без передавання його змісту провайдеру,

оскільки оброблення може виконуватися на боці браузера. Такий підхід відповідає принципам privacy-by-design і зменшує ризики витоку інформації під час підписання [14]. У поєднанні з правовим визнанням кваліфікованих електронних підписів це забезпечує юридичну значущість електронного документа.

Отже, безпековий контур системи електронного документообігу має охоплювати автентифікацію, розмежування доступу, захищене API, контроль завантаження файлів, аудит, виявлення аномалій, резервування, відновлення та інтеграцію з довірчими сервісами. Лише за таких умов можна поєднати зручність роботи з документами й належний рівень захисту.

Проведений аналіз дає підстави розглядати електронний документообіг як складну соціотехнічну систему, що поєднує правові вимоги, організаційні процедури, веб-технології, моделі доступу, аудит і аналітичне узагальнення даних. Документ у такій системі є не просто файлом, а керованим інформаційним об’єктом із життєвим циклом, станами, атрибутами, історією дій і правилами доступу.

1.2 Аналіз існуючих веб-орієнтованих систем електронного документообігу

Сучасні веб-орієнтовані системи електронного документообігу є ключовим інструментом організації інформаційних потоків, забезпечуючи централізоване зберігання документів, контроль доступу, підтримку робочих процесів та аналітичне опрацювання даних. Вони реалізують інтегрований підхід до керування документами, поєднуючи функції реєстрації, класифікації, маршрутизації, версійності та фіксації дій користувачів у межах єдиного програмного середовища. Разом із тим існуючі рішення відрізняються за рівнем реалізації функціональних можливостей, складністю архітектури та ступенем адаптації до конкретних потреб організації.

M-Files є програмним рішенням класу ECM/EDMS, призначеним для організації керування документами та інформаційними об’єктами на основі метаданих (рис. 1.2). Основне призначення системи полягає у забезпеченні централізованого доступу до документів, автоматизації процесів їх оброблення,

контролю версій, розмежування прав доступу та підтримки робочих процесів із урахуванням контексту використання інформації [15].

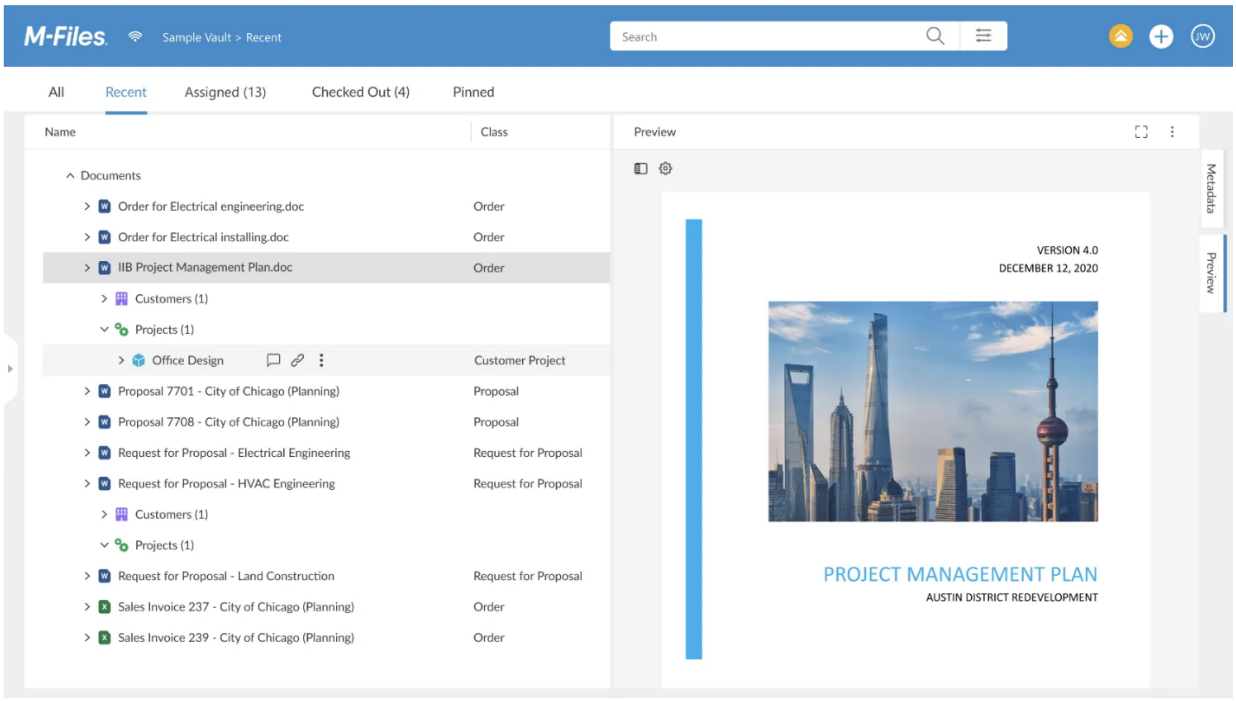


Рисунок 1.2 – Приклад інтерфейсу системи «M-Files» [16]

Переваги M-Files:

– підтримка автоматизованих workflow дозволяє формалізувати процеси узгодження та оброблення документів;

– можливість інтеграції з різними джерелами даних і зовнішніми системами підвищує універсальність використання;

– реалізація розмежування доступу та контролю версій забезпечує належний рівень безпеки та керованості даними.

Недоліки:

– складність початкового налаштування метаданих і структури системи потребує значних зусиль;

– комерційна модель ліцензування може бути фінансово витратною;

– потреба у навчанні користувачів через відмінність від традиційних файлових підходів.

Отже, система M-Files є функціонально розвиненим рішенням для управління документами, яке забезпечує високий рівень структурованості,

контролю та інтеграції даних, однак її ефективне використання потребує належної організації впровадження та адаптації користувачів до метаданого підходу.

DocuWare є веб-орієнтованою системою електронного документообігу, призначеною для централізованого керування документами, автоматизації бізнес-процесів та забезпечення контрольованого доступу до інформаційних ресурсів організації (рис. 1.3). Основне призначення системи полягає у цифровізації документообігу, зменшенні паперового навантаження, прискоренні оброблення документів і підвищенні прозорості управлінських процедур [17].

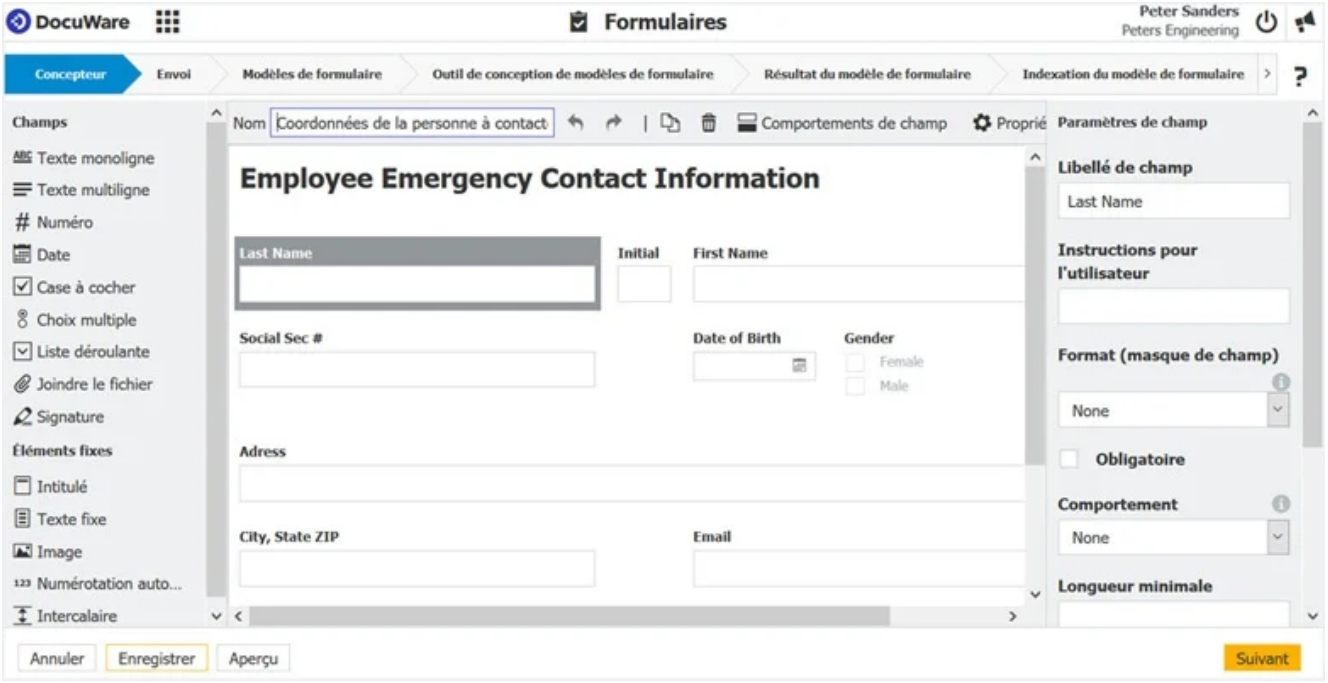


Рисунок 1.3 – Приклад інтерфейсу системи «DocuWare» [18]

Переваги DocuWare:

− орієнтація на веб та хмарну інфраструктуру забезпечує доступ до системи з будь-якого пристрою без складного розгортання;

− вбудовані засоби індексування та пошуку забезпечують швидкий доступ до документів за різними атрибутами;

− високий рівень інтеграції з корпоративними системами (ERP, CRM) розширює функціональні можливості використання.

Недоліки:

− комерційна модель використання передбачає значні фінансові витрати, особливо для малих організацій;

– обмежена гнучкість у глибокій кастомізації порівняно з відкритими рішеннями;

– потреба у навчанні персоналу для ефективного використання функціоналу системи.

Отже, DocuWare є сучасною веб-орієнтованою системою електронного документообігу, яка забезпечує високий рівень автоматизації та доступності, проте її використання доцільне переважно в умовах наявності відповідних ресурсів і готовності до впровадження хмарних технологій.

LogicalDOC є веб-орієнтованою системою електронного документообігу, призначеною для централізованого керування документами, організації доступу до них та автоматизації процесів оброблення інформації в межах підприємства (рис. 1.4). Основне призначення системи полягає у забезпеченні структурованого зберігання документів, їх класифікації, пошуку, контролю версій і підтримки спільної роботи користувачів із урахуванням ролей та прав доступу [19].

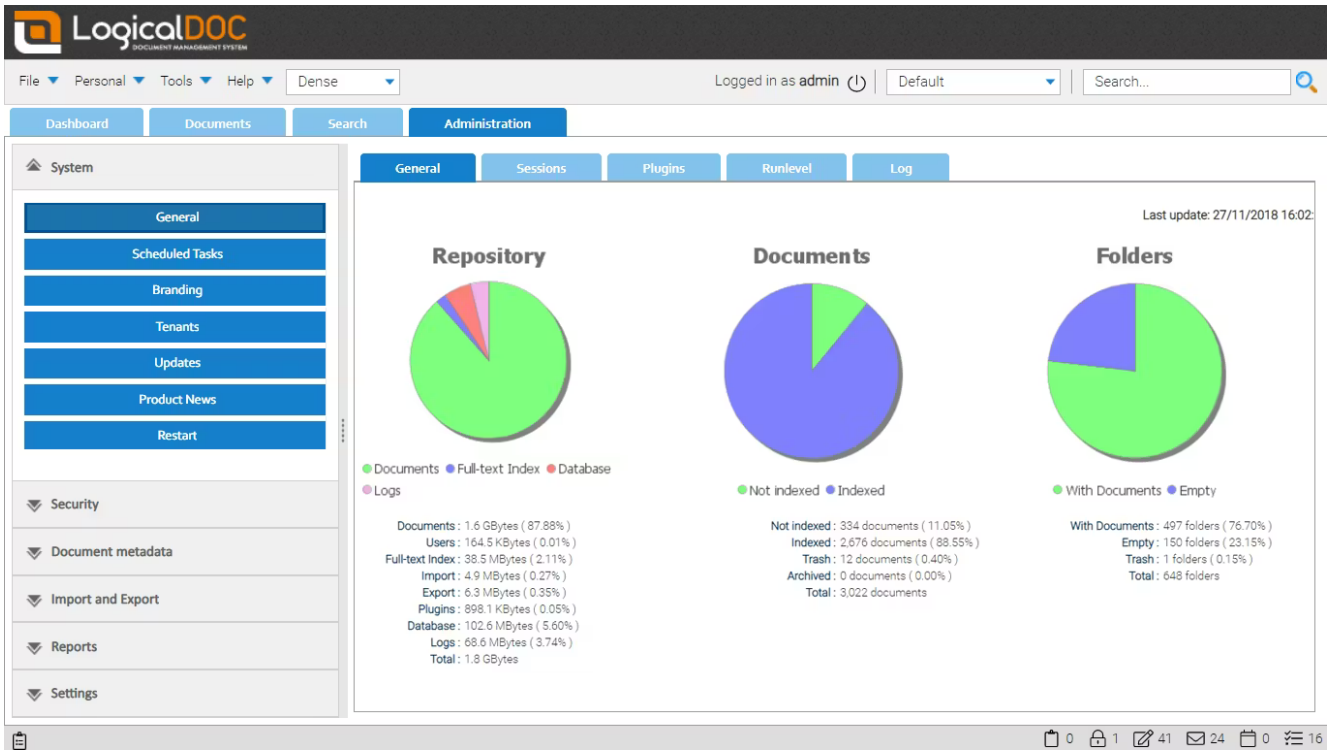


Рисунок 1.4 – Приклад інтерфейсу системи «LogicalDOC» [20]

Переваги LogicalDOC:

– наявність веб-інтерфейсу забезпечує доступ до системи без встановлення клієнтського ПЗ;

− розвинуті механізми індексування та повнотекстового пошуку підвищують швидкість роботи з документами;

− підтримка версійності та журналювання дій забезпечує контроль змін і відстеження активності;

− можливість інтеграції через API дозволяє взаємодіяти з іншими інформаційними системами.

Недоліки:

− обмежена функціональність у безкоштовній версії порівняно з комерційною редакцією;

− менш розвинуті механізми workflow у порівнянні з корпоративними ECM-рішеннями;

− потреба у налаштуванні серверного середовища та адмініструванні системи;

− інтерфейс може потребувати адаптації для підвищення зручності використання.

Отже, LogicalDOC є збалансованим рішенням для організації електронного документообігу, яке забезпечує базові та розширені можливості керування документами, однак найбільш ефективне використання досягається за умови розгортання повнофункціональної версії та належного адміністрування системи.

Проведений аналіз існуючих систем електронного документообігу показує, що більшість із них орієнтована на корпоративний сегмент і характеризується складною архітектурою, надлишковим функціоналом та високими вимогами до впровадження й супроводу. Такі рішення передбачають значні фінансові витрати, тривалий етап налаштування та необхідність підготовки користувачів, що ускладнює їх використання в умовах невеликих організацій або навчальних проєктів. Водночас базові потреби документообігу можуть бути забезпечені за рахунок значно простішої функціональної моделі, яка охоплює ключові операції без перевантаження системи другорядними можливостями. У зв’язку з цим доцільною є розробка нового програмного рішення, що орієнтується на мінімально

необхідний набір функцій, забезпечує швидке розгортання та не потребує ліцензійних витрат.

1.3 Постановка задачі на розробку адаптивної системи електронного документообігу

Виходячи з поставленої задачі розроблення програмного забезпечення у сфері електронного документообігу, доцільно сформулювати ключові вимоги, що визначають функціональні можливості системи, її технічну основу та сумісність із зовнішнім середовищем. Таке структурування дозволяє чітко окреслити межі розробки та забезпечити узгодженість між складовими програмного рішення:

Вимоги до функціональних характеристик:

– забезпечення реєстрації та автентифікації користувачів із підтримкою ролей;

– реалізація механізмів керування обліковими записами (зміна ролей, блокування та розблокування);

– забезпечення створення, редагування, перегляду та видалення документів;

– реалізація завантаження файлів документів і зберігання їх у файловій системі;

– підтримка версійності документів із фіксацією змін;

– забезпечення пошуку та фільтрації документів за основними атрибутами;

– реалізація розмежування доступу до документів і операцій над ними;

– підтримка архівації документів із можливістю подальшого перегляду;

– ведення журналу дій користувачів із фіксацією основних подій;

– формування аналітичної інформації у вигляді дашборду та звітів.

Вимоги до складу та параметрів технічних засобів:

– використання персонального комп’ютера або мобільного пристрою з доступом до мережі Інтернет;

– наявність сучасного веб-браузера для роботи із системою;

– використання серверного середовища, що підтримує виконання веб-застосунків;

– забезпечення роботи програмного забезпечення на базі Python і веб-фреймворку;

– використання реляційної системи керування базами даних для зберігання структурованих даних;

– використання файлового сховища для зберігання документів;

– забезпечення достатнього рівня продуктивності для оброблення типового обсягу даних.

Вимоги до інформаційної та програмної сумісності:

– використання стандартів веб-розробки (HTML, CSS, JavaScript) для забезпечення кросплатформеності;

– забезпечення коректної роботи системи в різних веб-браузерах;

– підтримка клієнт-серверної архітектури програмного забезпечення;

– використання ORM для взаємодії з базою даних;

– забезпечення узгодженого зберігання метаданих і файлових об’єктів документів;

– можливість інтеграції з іншими інформаційними системами через стандартні протоколи;

– забезпечення можливості подальшого розширення та модифікації системи без порушення її роботи.

Сформульована постановка задачі визначає вимоги до створення програмного забезпечення, яке забезпечує базовий функціонал електронного документообігу, адаптивність інтерфейсу та можливість подальшого розвитку системи відповідно до потреб предметної області.

2 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ АДАПТИВНОЇ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ ЕЛЕКТРОННОГО ДОКУМЕНТООБІГУ

2.1 Аналіз вимог та побудова діаграм варіантів використання

Аналіз вимог до програмного забезпечення є базовим етапом проєктування, оскільки на цьому рівні визначаються функціональні межі системи, ролі користувачів, основні сценарії роботи та взаємозв’язки між операціями [21]. Для адаптивної веб-орієнтованої системи електронного документообігу особливо важливим є розмежування повноважень адміністратора і звичайного користувача, оскільки вони мають різний доступ до документів, облікових записів, журналу подій, аналітики та звітності.

Для формалізації функціональних можливостей доцільно використати діаграми варіантів використання, які відображають взаємодію користувачів із системою через основні сценарії. На рис. 3.1 наведено діаграму варіантів використання для ролі адміністратора.

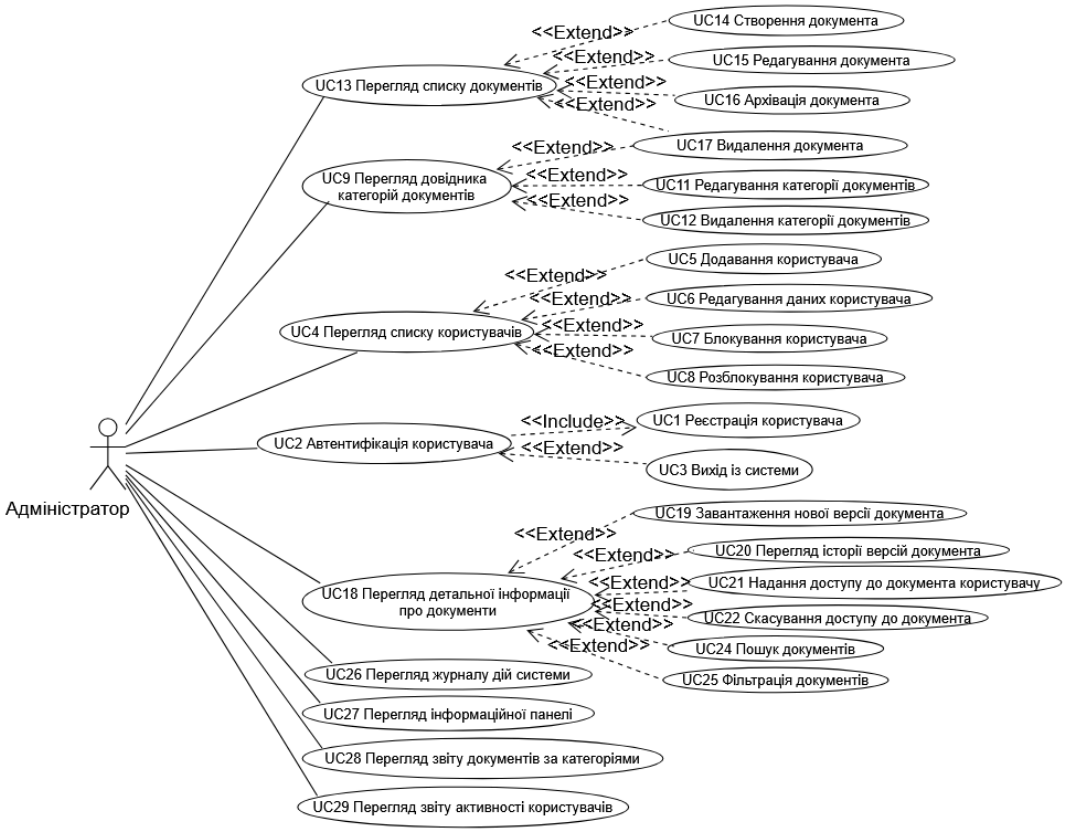


Рисунок 2.1 – Діаграма варіантів використання для ролі «адміністратора»

Адміністратор має найвищий рівень повноважень і виконує керівні та контрольні функції. До основних сценаріїв належать автентифікація, керування користувачами, категоріями й документами, перегляд деталей документа, журналу дій, інформаційної панелі та звітів. Додатково адміністратор може додавати, редагувати, блокувати й розблоковувати користувачів, створювати, змінювати, архівувати або видаляти документи й категорії. Окремо передбачено керування життєвим циклом документа: завантаження нових версій, перегляд історії, надання або скасування доступу, пошук і фільтрація. Отже, роль адміністратора охоплює не лише операційне керування даними, а й контроль активності системи та аналіз стану документообігу.

Для повного представлення функціональної моделі системи необхідно окремо розглянути сценарії звичайного користувача, оскільки ця роль орієнтована не на адміністрування, а на безпосередню роботу з документами. Користувач взаємодіє лише з тими об’єктами й операціями, які відповідають його рівню доступу, що забезпечує контрольовану та безпечну модель документообігу. Рис. 2.2 відображає діаграму варіантів використання для ролі «користувач».

Рисунок 2.2 – Діаграма варіантів використання для ролі «користувач»

Роль користувача охоплює вхід до системи, реєстрацію, завершення сеансу та перегляд списку документів. Центральним сценарієм є робота з документами, що включає пошук, фільтрацію, перегляд деталей, створення нового документа, редагування власного документа, архівацію та завантаження нової версії. Така структура демонструє, що документ виступає основним інформаційним об’єктом системи.

Окреме значення мають сценарії контролю доступу та версійності. Користувач може переглядати доступні йому документи, перевіряти власні права щодо конкретного документа та переглядати історію версій. Це підвищує прозорість роботи з файлами, зменшує ризик несанкціонованих змін і підтримує контроль актуального стану документа.

Також передбачено перегляд профілю користувача, що забезпечує персоналізовану взаємодію із системою. Отже, роль користувача має обмежений, але достатній набір функцій для повсякденної роботи з документами без доступу до адміністративних механізмів керування системою.

2.2 Розробка архітектури системи

Для веб-орієнтованої системи електронного документообігу архітектура має забезпечувати виконання користувацьких сценаріїв, підтримку ролей, контроль доступу, зберігання документів, ведення журналу дій, формування аналітики та адаптивне відображення сторінок на різних пристроях. З огляду на обрану технологічну основу доцільним є використання клієнт-серверної архітектури, у якій користувач працює через веб-інтерфейс, а обробка запитів, перевірка прав, робота з документами та доступ до даних виконуються на серверному рівні. Такий підхід централізує бізнес-логіку, зменшує залежність клієнтської частини від структури даних і забезпечує контрольовану взаємодію з документами. Загальну архітектуру системи наведено на рис. 2.3.

Користувачі взаємодіють із системою через веб-інтерфейс, реалізований засобами HTML5, CSS3, Bootstrap 5 і JavaScript. Цей рівень відповідає за відображення сторінок входу, реєстрації, роботи з документами, інформаційної

панелі та звітів. Його основне призначення полягає у забезпеченні зручної взаємодії із серверною частиною через HTTP-запити та відповіді.

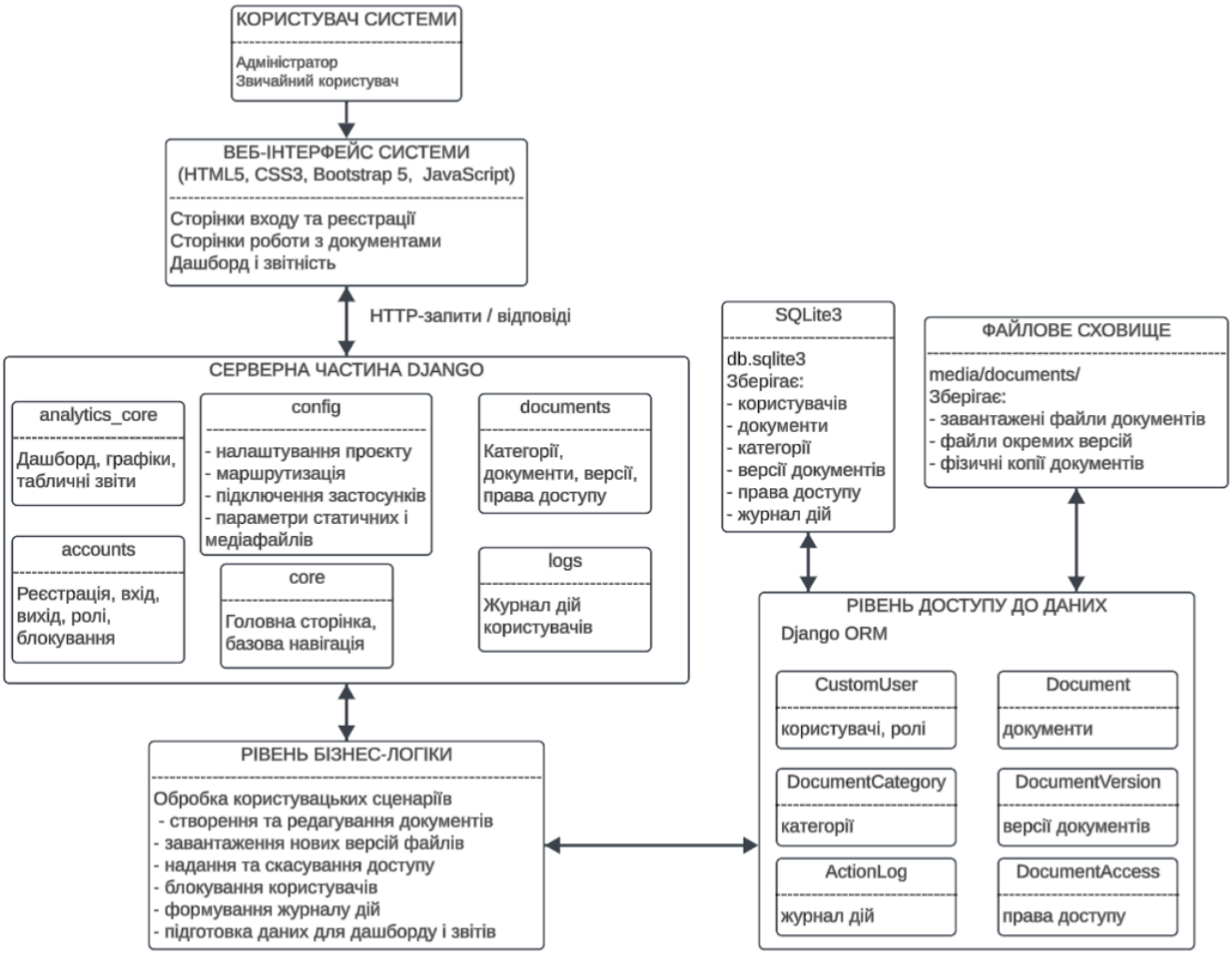


Рисунок 2.3 – Загальна архітектура системи

Центральним елементом архітектури є серверна частина Django, що виконує роль ядра системи. Вона складається з логічних модулів: accounts відповідає за користувачів, ролі та блокування; documents – за категорії, документи, версії та права доступу; logs – за фіксацію дій користувачів; analytics_core – за дашборд, графіки й звіти; core – за головну сторінку та навігацію; config – за налаштування проєкту, маршрутизацію, підключення застосунків і роботу зі статичними та медіафайлами.

Рівень бізнес-логіки узгоджує користувацькі сценарії з операціями над даними. У ньому виконуються створення й редагування документів, завантаження нових версій, керування доступом, блокування користувачів, формування журналу дій і підготовка даних для аналітики. Завдяки цьому кожна дія користувача

проходить через перевірки, правила доступу та процедури оброблення, а не передається безпосередньо до бази даних.

Доступ до даних реалізовано через Django ORM, що пов’язує програмні моделі з фізичним зберіганням інформації. Основними сутностями є CustomUser, Document, DocumentCategory, DocumentVersion, ActionLog і DocumentAccess. Вони забезпечують зберігання користувачів і ролей, облік документів, класифікацію, версійність, журналювання дій та розмежування доступу без дублювання даних.

Фізичне зберігання поділено на базу даних і файлове сховище. У SQLite3-файлі db.sqlite3 зберігаються структуровані дані про користувачів, документи, категорії, версії, права доступу та журнал дій. Самі файли документів розміщуються у каталозі media/documents/, що дозволяє зберігати в базі метадані та зв’язки, а великі бінарні об’єкти – у файловій системі.

Загалом подана архітектура відображає розподіл системи на інтерфейсний, серверний, бізнесовий, модельний і фізичний рівні зберігання. Це спрощує підтримку модулів, підвищує керованість розробки та створює основу для подальшого розширення, зокрема додавання нових звітів, складніших механізмів погодження документів або переходу від SQLite3 до серверної СУБД.

2.3 Проєктування структури даних програмного забезпечення

У межах розроблюваного програмного забезпечення структура даних має відображати логіку предметної області, де документ є не лише файлом, а повноцінним інформаційним об’єктом із реквізитами, статусом, автором, категорією, історією змін і правилами доступу. З урахуванням обраного технологічного стеку доцільним є використання реляційної моделі даних на основі SQLite3 та Django ORM. Це дозволяє описувати сутності системи через моделі, автоматично формувати таблиці бази даних, підтримувати зв’язки між ними та виконувати операції з даними без прямого написання SQL-запитів. Функції автентифікації й базового керування користувачами доцільно реалізовувати засобами стандартних механізмів Django, що спрощує програмну структуру.

У межах розроблюваної системи структура бази даних побудована так, щоб відобразити основні об’єкти електронного документообігу: користувачів, документи, категорії, версії файлів, права доступу, журнали дій і службові механізми Django. Замість детального опису кожного атрибута доцільно подати перелік таблиць із коротким поясненням їхнього функціонального призначення:

- accounts_customuser – зберігає облікові записи користувачів, дані автентифікації, службові ознаки активності, адміністративні права та роль користувача в системі;
- documents_document – містить основні відомості про документи, зокрема їх назву, опис, статус, автора, категорію та часові параметри створення й оновлення;
- documents_documentversion – забезпечує зберігання версій документів, інформації про завантажені файли, їх розмір, формат, дату завантаження та користувача, який виконав дію;
- documents_documentcategory – використовується як довідник категорій документів і забезпечує їх класифікацію для подальшого пошуку, фільтрації та формування звітності;
- documents_documentaccess – фіксує індивідуальні права доступу користувачів до документів, визначаючи можливість перегляду або редагування;
- logs_actionlog – зберігає журнал дій користувачів, пов’язаних із входом, виходом, роботою з документами, зміною ролей, блокуванням користувачів і керуванням доступом;
- django_content_type – містить службову інформацію про моделі Django, необхідну для роботи механізму дозволів;
- auth_permission – зберігає перелік дозволів, які визначають можливі операції над об’єктами системи;
- auth_group – використовується для зберігання груп користувачів, через які можуть призначатися колективні права доступу;

– auth_group_permissions – реалізує зв’язок між групами користувачів і дозволами;

– accounts_customuser_groups – встановлює належність користувачів до певних груп;

– accounts_customuser_user_permissions – забезпечує індивідуальне призначення дозволів окремим користувачам;

– django_admin_log – фіксує службові дії, виконані через адміністративну панель Django.

На основі зазначених таблиць формується фізична модель бази даних, яка відображає взаємозв’язки між сутностями програмного забезпечення (рис. 2.4)

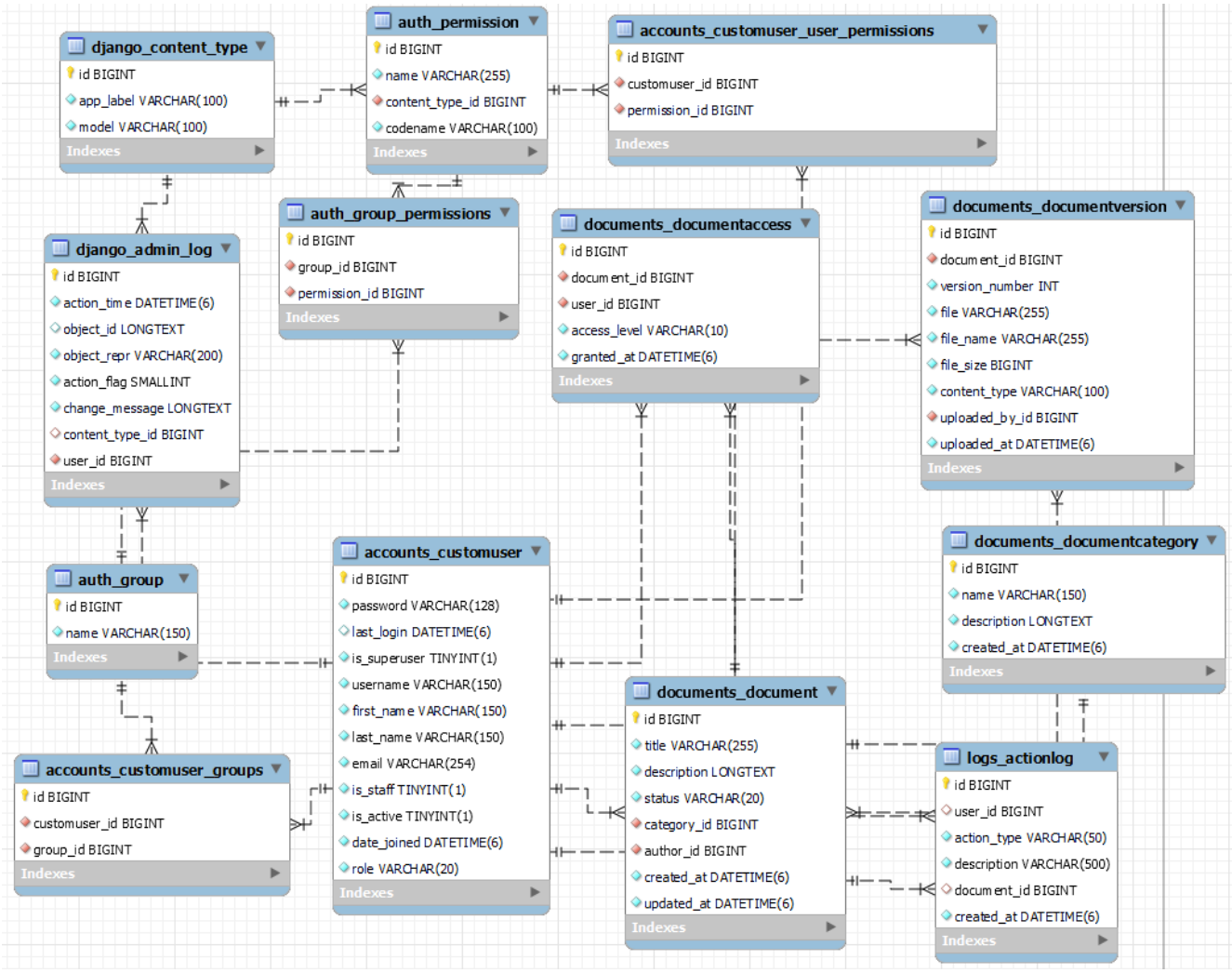


Рисунок 2.4 – Фізична модель бази даних

Побудована модель дозволяє наочно простежити залежності між основними об’єктами системи та оцінити коректність організації даних ще до етапу

програмної реалізації. Вона також спрощує подальше створення моделей Django, налаштування зв’язків між сутностями та розширення бази даних у разі додавання нових функціональних можливостей.

2.4 Моделювання основних процесів

Моделювання основних процесів програмного забезпечення виконується для уточнення логіки взаємодії між користувачем, веб-представленнями, формами, механізмами автентифікації та службовими компонентами системи. Якщо діаграми варіантів використання показують функціональні можливості на рівні ролей, то діаграми послідовності деталізують порядок виконання окремих сценаріїв і склад програмних компонентів, що беруть участь у реалізації певної дії [22]. Для системи електронного документообігу одним із базових процесів є авторизація користувача, оскільки після входу визначається доступ до документів, адміністративних функцій, журналу дій та аналітики. Діаграму послідовності процесу авторизації користувача наведено на рис. 2.5.

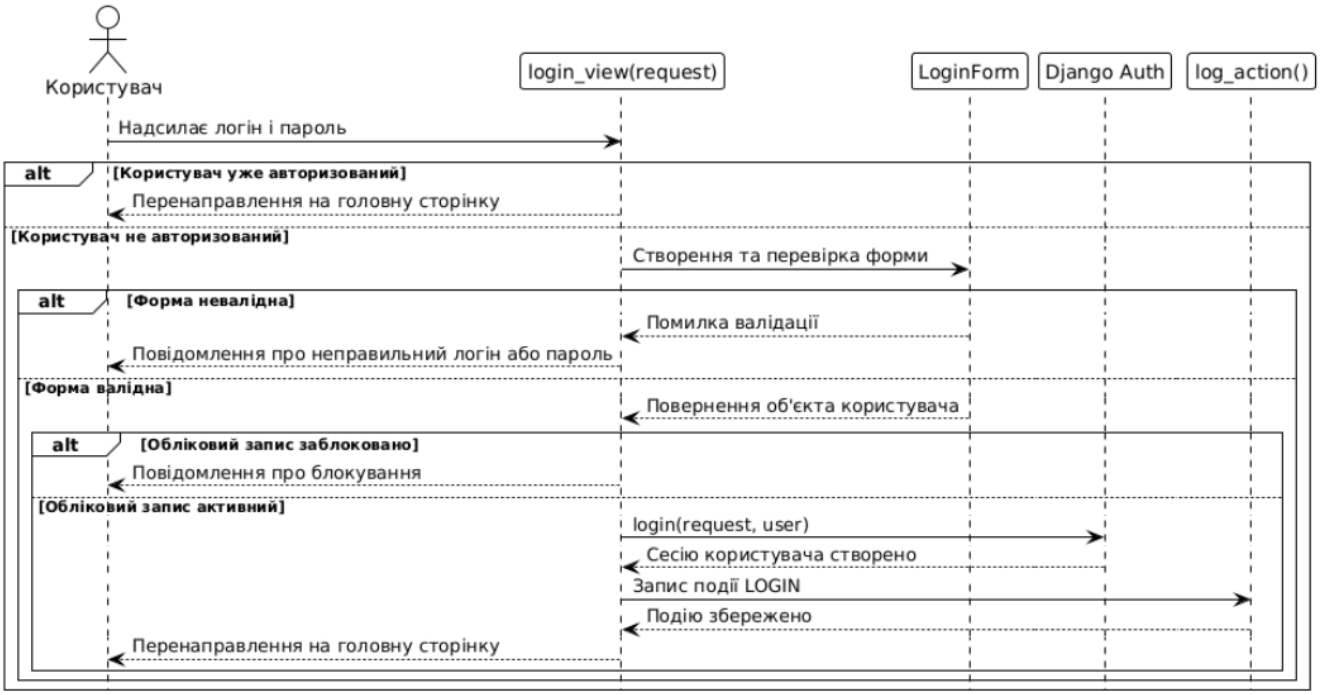


Рисунок 2.5 – Діаграма послідовності процесу авторизації користувача

Процес авторизації починається з надсилання користувачем логіна й пароля до системи. Запит обробляється представленням login_view(request), яке координує подальший сценарій. Якщо користувач уже авторизований, система не виконує

повторну перевірку даних, а перенаправляє його на головну сторінку, що усуває дублювання операцій.

Якщо користувач не авторизований, створюється та перевіряється форма LoginForm, яка виконує первинну валідацію введених даних. У разі помилки система повертає повідомлення про неправильний логін або пароль і не створює сесію. За умови коректності форми виконується перевірка стану облікового запису: якщо він заблокований, авторизація припиняється, навіть за правильних облікових даних.

Якщо обліковий запис активний, механізм Django Auth виконує login(request, user) і створює користувацьку сесію. Після цього функція log_action() фіксує подію входу з типом LOGIN у журналі дій, а користувач перенаправляється на головну сторінку.

Для деталізації роботи з документами доцільно розглянути також процес формування файлових версій, оскільки саме версійність забезпечує контроль змін і збереження історії оновлення документа. У системі ці дії реалізуються через взаємодію користувача з представленнями Django, формами введення, моделями даних і журналом подій.

Для деталізації роботи з документами доцільно розглянути процес створення документа та формування його файлової версії, оскільки версійність забезпечує контроль змін і збереження історії оновлень. У системі ці дії реалізуються через взаємодію користувача з представленнями Django, формами введення, моделями даних і журналом подій. На рис. 2.6 наведено діаграму послідовності процесу створення документа та його версії.

Процес починається з введення користувачем даних документа, після чого запит надходить до представлення Django. Представлення перевіряє форму, і за умови коректності даних створює документ із прив’язкою до поточного користувача через request.user. Після цього в журналі дій фіксується подія CREATE_DOCUMENT, а користувач переходить на сторінку створеного документа. Якщо дані введено некоректно, система повертає форму з повідомленнями про помилки, не створюючи запис у базі даних.

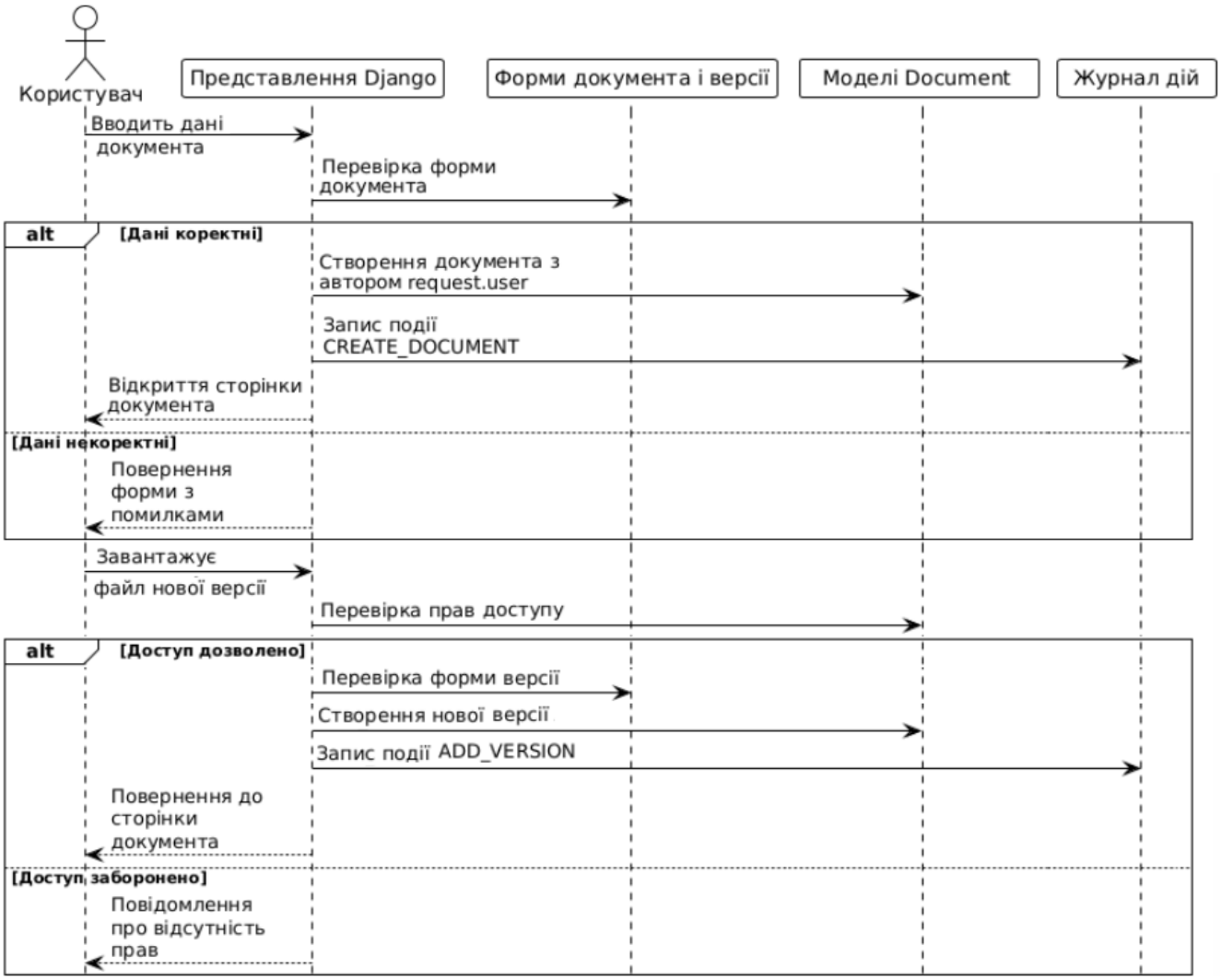


Рисунок 2.6 – Діаграма послідовності процесу створення документа та його версії

Друга частина процесу стосується завантаження нової версії документа. Користувач передає файл, після чого система перевіряє його права доступу. Якщо доступ дозволено, виконується перевірка форми версії, створюється новий запис у базі даних, а в журналі фіксується подія ADD_VERSION. У разі відсутності прав система блокує завантаження та повертає відповідне повідомлення, що підтверджує серверний характер контролю доступу.

Окремого розгляду потребує процес керування доступом до документа, оскільки він визначає захищеність інформації та коректність розмежування прав між користувачами. Надання доступу передбачає перевірку повноважень, валідацію введених даних і контроль відсутності дублювання вже створених прав. На рис. 2.7 наведено діаграму послідовності процесу надання доступу до документа.

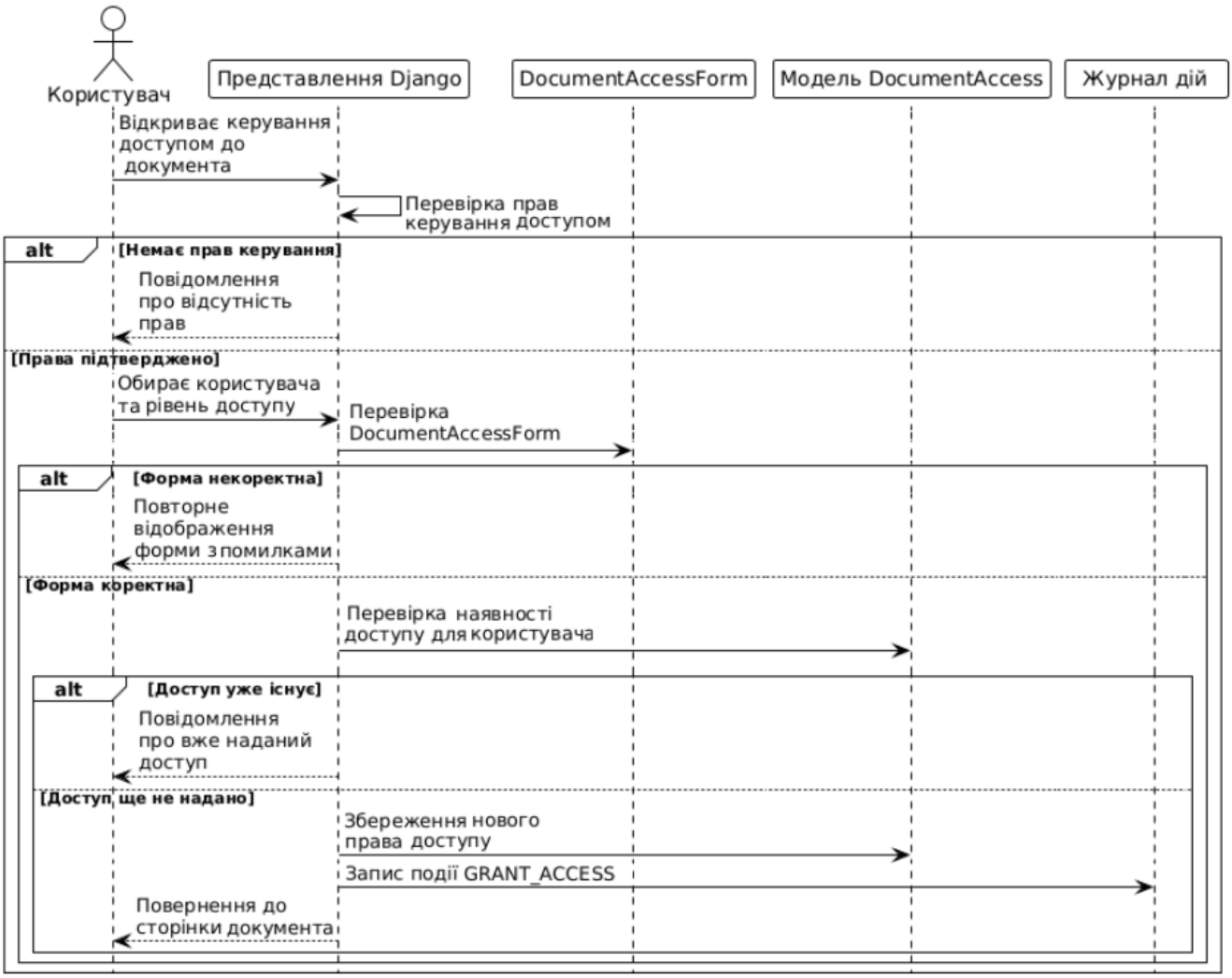


Рисунок 2.7 – Діаграма послідовності процесу надання доступу до документа

Процес починається з відкриття сторінки керування доступом. Представлення Django перевіряє, чи має користувач відповідні повноваження. Якщо прав немає, система припиняє сценарій і повертає повідомлення про відсутність доступу, що підтверджує виконання контролю на серверному рівні.

Якщо запис доступу вже існує, система повідомляє про дублювання. Якщо доступ ще не надавався, створюється новий запис у DocumentAccess, у журналі дій фіксується подія GRANT_ACCESS, після чого користувач повертається до сторінки документа.

2.5 Реалізація програмного забезпечення

Реалізація адаптивної веб-орієнтованої системи електронного документообігу виконувалася на основі фреймворку Django, який забезпечує

механізми маршрутизації, роботи з моделями, формами, шаблонами, автентифікацією та правами користувачів. Основну увагу було зосереджено на модулях реєстрації, автентифікації, керування документами, зберігання версій, розмежування доступу, журналювання дій та формування аналітичної інформації.

Базовим елементом системи є модель користувача, оскільки вона визначає можливість входу, роль користувача та рівень доступу до функцій системи. Для цього стандартну модель Django було розширено полем ролі, що дозволило розмежувати адміністратора і звичайного користувача без ускладнення структури бази даних. Реалізацію моделі користувача наведено на рис. 2.8.

```
class CustomUser(AbstractUser):
    class Roles(models.TextChoices):
        ADMIN = "administrator", "Адміністратор"
        USER = "user", "Користувач"

    role = models.CharField(
        max_length=20,
        choices=Roles.choices,
        default=Roles.USER,
        verbose_name="Роль"
    )

    def __str__(self):
        return f"{self.username} ({self.get_role_display()})"
```

Рисунок 2.8 – Реалізація моделі користувача

Клас CustomUser успадковує AbstractUser, зберігаючи стандартні можливості Django для роботи з обліковими записами, паролями, сесіями та автентифікацією. Вкладений перелік Roles визначає два типи користувачів: адміністратора та звичайного користувача. Поле role реалізовано як символьне поле з обмеженим набором значень, причому за замовчуванням новому обліковому запису призначається роль звичайного користувача. Це відповідає логіці безпечної реєстрації, оскільки адміністративні повноваження не надаються автоматично. Метод __str__() формує зручне текстове представлення користувача із зазначенням його імені та ролі.

Функція, наведена на рис. 2.9, реалізує реєстрацію нового користувача в системі електронного документообігу. Спочатку перевіряється, чи користувач уже

авторизований; якщо так, він перенаправляється на головну сторінку, що запобігає повторній реєстрації.

```
def register_view(request):
    if request.user.is_authenticated:
        return redirect("home")
    if request.method == "POST":
        form = RegisterForm(request.POST)
        if form.is_valid():
            user = form.save(commit=False)
            user.role = "user"
            user.save()
        log_action(
            user=user,
            action_type=ActionLog.ActionTypes.REGISTER,
            description=f"Зареєстровано нового користувача:
            {user.username}."
        )
        messages.success(request, "Реєстрацію успішно завершено.
        Тепер увійдіть до системи.")
        return redirect("login")
    else:
        form = RegisterForm()
    return render(request, "accounts/register.html", {"form": form})
```

Рисунок 2.9 – Реалізація процесу реєстрації користувача

Функція, наведена на рис. 2.10, перевизначає стандартний метод збереження моделі для автоматичного формування метаданих завантаженого файлу. Зокрема, назва файлу визначається як остання частина шляху, що дозволяє зберігати в базі даних його фактичне ім’я без службових каталогів.

```
def save(self, *args, **kwargs):
    if self.file:
        self.file_name = self.file.name.split("/")[-1]
        self.file_size = self.file.size or 0
        if hasattr(self.file.file, "content_type"):
            self.content_type = self.file.file.content_type or ""
    super().save(*args, **kwargs)
```

Рисунок 2.10 – Автоматичне формування метаданих при збереженні документа

На рис. 2.11 наведено фрагмент програмного коду, який реалізує створення нового документа авторизованим користувачем. Декоратор login_required обмежує доступ до функції лише для користувачів, які виконали вхід до системи, що відповідає вимогам контрольованої роботи з документами. Якщо запит має тип

POST, система отримує дані з форми, перевіряє їх коректність і лише після успішної валідації переходить до створення об’єкта документа.

```
@login_required
def document_create_view(request):
    if request.method == "POST":
        form = DocumentForm(request.POST)
        if form.is_valid():
            document = form.save(commit=False)
            document.author = request.user
            document.save()
            log_action(
                user=request.user,
                action_type=ActionLog.ActionTypes.CREATE_DOCUMENT,
                description=f"Створено документ '{document.title}'.",
                document=document
            )
            messages.success(request, "Документ успішно створено.")
            return redirect("document_detail", document_id=document.id)
    else:
        form = DocumentForm()
    return render(request, "documents/document_form.html",
    {"form": form, "mode": "create"})
```

Рисунок 2.11 – Створення документа з прив’язкою до автора та фіксації події

Фрагмент коду на рис. 2.12 реалізує початковий етап додавання нової версії документа. Доступ до функції мають лише авторизовані користувачі, після чого система за ідентифікатором шукає відповідний документ у базі даних. Якщо документ не знайдено, повертається стандартна помилка, що забезпечує коректну обробку запиту до початку роботи з файлом.

```
@login_required
def document_add_version_view(request, document_id):
    document = get_object_or_404(Document, id=document_id)

    can_edit = (
        request.user.is_superuser
        or request.user.role == "administrator"
        or document.author_id == request.user.id
        or document.accesses.filter(user=request.user,
        access_level="edit").exists()
    )
```

Рисунок 2.12 – Перевірка прав перед додаванням нової версії документа

На рис. 2.13 наведено фрагмент програмного коду, який реалізує початкову перевірку прав доступу під час переходу до керування доступом до конкретного документа. Функція доступна лише авторизованим користувачам, після чого

система отримує документ за його ідентифікатором або повертає помилку, якщо відповідного запису не існує. Такий механізм забезпечує коректну прив'язку подальших дій до конкретного документа та запобігає виконанню операцій над неіснуючими об'єктами.

```
@login_required
def document_manage_access_view(request, document_id):
    document = get_object_or_404(Document, id=document_id)

    can_manage = (
        request.user.is_superuser
        or request.user.role == "administrator"
        or document.author_id == request.user.id
    )

    if not can_manage:
        messages.error(request,
        "У вас немає прав на керування доступом до цього документа.")
        return redirect("document_detail", document_id=document.id)
```

Рисунок 2.13 – Перевірка повноважень для керування доступом до документа

Функція, яку наведено на рис. 2.14, реалізує універсальний механізм додавання запису до журналу дій системи. Її призначення полягає у централізованій фіксації важливих подій, які виникають під час роботи користувачів із програмним забезпеченням, зокрема реєстрації, входу, створення документа, додавання версії, зміни доступу або інших операцій. Завдяки використанню окремої функції журналювання однакова логіка запису подій може застосовуватися в різних частинах системи без дублювання коду.

```
def log_action(*, user=None, action_type, description, document=None):
    ActionLog.objects.create(
        user=user,
        action_type=action_type,
        description=description,
        document=document
    )
```

Рисунок 2.14 – Реалізація універсального механізму фіксації дій користувачів

У процесі виконання функція створює новий об'єкт моделі ActionLog, до якого передаються користувач, тип дії, текстовий опис події та, за потреби, пов'язаний документ. Параметри user і document можуть бути необов'язковими, що

дозволяє фіксувати як дії конкретних користувачів над документами, так і загальні системні події.

На рис. 2.15 наведено фрагмент програмного коду, який реалізує формування адміністративного дашборду системи електронного документообігу. Доступ до функції обмежено декоратором admin_required, тому перегляд зведеної статистики дозволяється лише користувачам з адміністративними повноваженнями. У функції виконується підрахунок основних кількісних показників системи: загальної кількості користувачів, активних облікових записів, документів, активних і архівних документів, категорій, версій документів та записів журналу дій.

```
@admin_required
def dashboard_view(request):
    total_users = User.objects.count()
    active_users = User.objects.filter(is_active=True).count()
    total_documents = Document.objects.count()
    active_documents = Document.objects.filter(status=Document.Statuses.ACTIVE).count()
    archived_documents = Document.objects.filter(status=Document.Statuses.ARCHIVED).count()
    total_categories = DocumentCategory.objects.count()
    total_versions = DocumentVersion.objects.count()
    total_logs = ActionLog.objects.count()
    categories_data = (
        DocumentCategory.objects
        .annotate(doc_count=Count("documents"))
        .order_by("name")
    )
```

Рисунок 2.15 – Формування статистичних показників адміністративного дашборду

Окрема частина коду відповідає за підготовку агрегованих даних щодо категорій документів. Для цього використовується механізм ORM-агрегації annotate(), який додає до кожної категорії кількість пов’язаних із нею документів, після чого результати впорядковуються за назвою. Така реалізація дозволяє формувати інформаційну панель не на основі вручну підготовлених значень, а безпосередньо з актуального стану бази даних, що забезпечує коректне відображення статистики та створює основу для побудови графіків і звітів у веб-інтерфейсі.

3 РЕЗУЛЬТАТИ РОЗРОБКИ АДАПТИВНОЇ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ ЕЛЕКТРОННОГО ДОКУМЕНТООБІГУ

3.1 Опис реалізованих можливостей системи та рівень виконання поставлених вимог

У межах практичної реалізації було розроблено адаптивну веб-орієнтовану систему електронного документообігу, призначену для організації роботи з електронними документами, користувачами, версіями файлів, правами доступу, журналом дій та аналітичною звітністю. Програмне забезпечення побудовано на основі фреймворку Django, що дозволило використати вбудовані механізми автентифікації, маршрутизації, роботи з формами, ORM-моделей і файлового зберігання. Реалізований функціонал охоплює основні вимоги, визначені на етапі постановки задачі, зокрема реєстрацію користувачів, рольове розмежування доступу, керування документами, зберігання версій, фіксації подій та формування аналітичних представлень.

Для узагальнення результатів реалізації доцільно зіставити поставлені функціональні вимоги з фактично реалізованими можливостями програмного забезпечення. Такий підхід дозволяє оцінити повноту виконання вимог і визначити, які компоненти системи були реалізовані повністю, а які можуть бути розширені в подальших версіях. Результати такого зіставлення наведено в табл. 3.1.

Таблиця 3.1 – Оцінка реалізованих можливостей системи відповідно до поставлених вимог

№	Поставлена вимога	Реалізована можливість системи	Рівень виконання
1	2	3	4
1	Реєстрація користувачів у системі	Реалізовано веб-форму реєстрації користувача з введенням імені користувача, електронної адреси та пароля	Виконано повністю
2	Автентифікація користувачів	Реалізовано вхід до системи за логіном і паролем із використанням стандартного механізму Django Auth	Виконано повністю

Продовження таблиці 3.1.

1	2	3	4
3	Завершення сеансу роботи	Реалізовано вихід користувача із системи через окремий маршрут і кнопку в навігаційному меню	Виконано повністю
4	Підтримка ролей адміністратора і користувача	Створено розширену модель користувача з полем ролі administrator або user	Виконано повністю
5	Блокування та розблокування користувачів	Адміністратор може змінювати стан активності користувача, а заблокований користувач не може увійти до системи	Виконано повністю
6	Перегляд списку користувачів	Реалізовано сторінку адміністрування користувачів із пошуком і фільтрацією	Виконано повністю
7	Зміна ролі користувача	Реалізовано форму редагування користувача, у якій адміністратор може змінити роль облікового запису	Виконано повністю
8	Ведення довідника категорій документів	Реалізовано створення та перегляд категорій документів через веб-інтерфейс	Виконано повністю
9	Створення запису про документ	Реалізовано форму створення документа із зазначенням назви, опису, категорії та статусу	Виконано повністю
10	Перегляд списку документів	Реалізовано сторінку списку документів із відображенням назви, категорії, статусу, автора та дати створення	Виконано повністю
11	Пошук документів	Реалізовано пошук документів за назвою та описом	Виконано повністю
12	Фільтрація документів	Реалізовано фільтрацію документів за категорією та статусом	Виконано повністю
13	Перегляд детальної інформації про документ	Реалізовано сторінку документа з основними атрибутами, версіями та правами доступу	Виконано повністю
14	Редагування документа	Реалізовано зміну основних атрибутів документа для адміністратора, автора або користувача з правом редагування	Виконано повністю
15	Архівація документа	Реалізовано переведення документа в архівний статус без фізичного видалення з бази даних	Виконано повністю

Продовження таблиці 3.1.

16	Видалення документа	Реалізовано видалення документа лише для адміністратора з попереднім підтвердженням	Виконано повністю
17	Завантаження файлів документа	Реалізовано завантаження файлів через форму додавання нової версії документа	Виконано повністю
18	Зберігання кількох версій документа	Реалізовано модель версій документа з автоматичним присвоєнням номера версії	Виконано повністю
19	Збереження атрибутів файлу	Автоматично зберігаються назва файла, розмір, тип, дата завантаження та користувач, який виконав завантаження	Виконано повністю
20	Надання доступу до документа конкретному користувачу	Реалізовано механізм надання прав перегляду або редагування документа окремим користувачам	Виконано повністю
21	Скасування доступу до документа	Реалізовано видалення раніше наданого права доступу через сторінку керування доступом	Виконано повністю
22	Ведення журналу дій	Реалізовано модель журналу, у якій фіксуються основні події системи	Виконано повністю
23	Перегляд журналу дій адміністратором	Реалізовано окрему сторінку журналу дій із таблицею подій	Виконано повністю
24	Фільтрація журналу дій	Реалізовано фільтрацію журналу за типом події та користувачем	Виконано повністю
25	Інформаційна панель системи	Реалізовано дашборд із кількісними показниками користувачів, документів, категорій, версій і подій	Виконано повністю
26	Графічна візуалізація даних	Реалізовано графіки розподілу документів за категоріями, створення документів за датами та подій журналу	Виконано повністю
27	Звіт документів за категоріями	Реалізовано табличний звіт із кількістю документів, активних і архівних документів за кожною категорією	Виконано повністю
28	Звіт активності користувачів	Реалізовано звіт із кількістю дій, завантажених версій і датою останньої активності користувачів	Виконано повністю

Отже, у межах підрозділу було узагальнено результати практичної реалізації програмного забезпечення та встановлено відповідність між поставленими вимогами і фактично розробленими можливостями системи. Проведене зіставлення показало, що реалізована система охоплює основні функціональні напрями електронного документообігу: роботу з користувачами, документами, категоріями, версіями файлів, правами доступу, журналом дій, дашбордом і звітністю. Особливе значення має те, що більшість вимог виконано повністю, а ключові операції супроводжуються серверною перевіркою прав і фіксацією дій користувачів у журналі. Це підтверджує, що розроблене програмне забезпечення відповідає визначеним функціональним вимогам і може використовуватися як завершений прототип адаптивної веб-орієнтованої системи електронного документообігу.

3.2 Оцінка характеристик програмного забезпечення та результати його тестування

Оцінювання характеристик розробленого програмного забезпечення виконувалося з метою перевірки відповідності системи функціональним, технічним та експлуатаційним вимогам. Основну увагу було приділено працездатності веб-інтерфейсу, коректності виконання операцій із користувачами та документами, правильності розмежування доступу, стабільності роботи механізму завантаження файлів, веденню журналу дій і формуванню аналітичної інформації. Тестування проводилося в локальному середовищі з використанням веб-браузера, Django-сервера розробки, бази даних SQLite3 та підготовленого набору демонстраційних даних.

У процесі перевірки було застосовано функціональне тестування основних сценаріїв використання системи. Кожен сценарій передбачав виконання конкретної дії користувачем або адміністратором та перевірку очікуваного результату у веб-інтерфейсі, базі даних або журналі подій. Окремо перевірялися ситуації, пов’язані з обмеженням доступу, оскільки для системи електронного

документообігу важливо не лише забезпечити створення і зберігання документів, а й гарантувати, що користувачі отримують доступ тільки до дозволених їм ресурсів.

Для фіксації результатів тестування інтерфейсу спочатку перевірено доступність головної сторінки, оскільки вона є початковою точкою взаємодії користувача із системою. За цим тестовим сценарієм перевірялося коректне завантаження сторінки, відображення навігаційного меню, наявність основного інформаційного блоку та доступність переходів до сторінок входу і реєстрації. Результат перевірки головної сторінки системи електронного документообігу наведено на рис. 3.1.

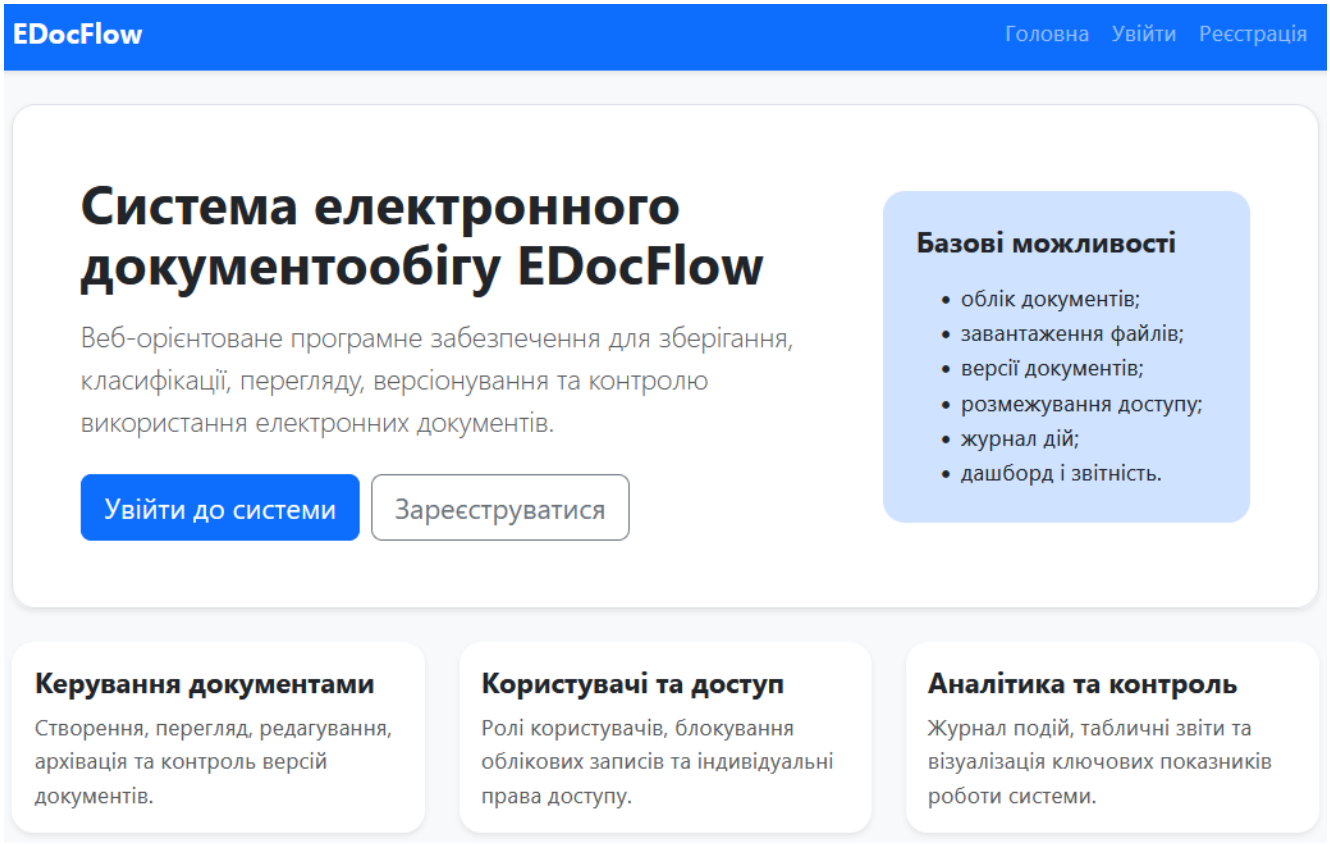


Рисунок 3.1 – Головна сторінка системи електронного документообігу

На зображенні видно, що головна сторінка системи EDocFlow відображається коректно та містить основні елементи початкової взаємодії з користувачем. У верхній частині сторінки розміщено навігаційну панель із назвою системи та посиланнями на головну сторінку, вхід і реєстрацію, що підтверджує доступність базових маршрутів вебзастосунку. Центральний блок містить короткий опис призначення системи, кнопки переходу до авторизації та реєстрації,

а також перелік базових можливостей, пов’язаних з обліком документів, завантаженням файлів, версіями, доступом, журналом дій, дашбордом і звітністю.

На наступному етапі тестування було перевірено сценарій створення нового облікового запису, оскільки реєстрація є початковою умовою подальшої персоналізованої роботи користувача із системою. У межах перевірки оцінювалася наявність необхідних полів, коректність структури форми та доступність переходу до сторінки входу для вже зареєстрованих користувачів. Форму реєстрації нового користувача наведено на рис. 3.2.

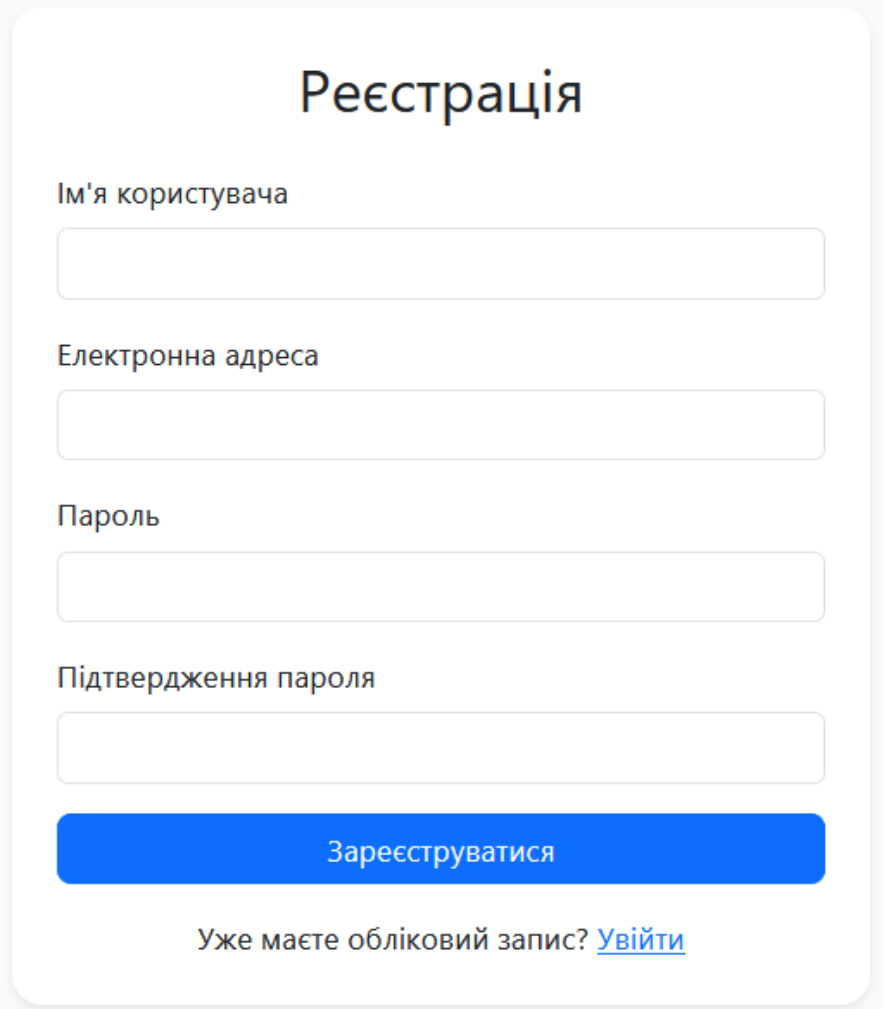


Рисунок 3.2 – Форма реєстрації нового користувача

Форма реєстрації містить усі основні поля, необхідні для створення облікового запису: ім’я користувача, електронну адресу, пароль і підтвердження пароля. Така структура відповідає базовим вимогам до реєстрації, оскільки дозволяє ідентифікувати користувача, зберегти контактні дані та перевірити правильність введення пароля перед створенням облікового запису.

Для перевірки сценарію входу до системи було проаналізовано сторінку автентифікації, яка забезпечує доступ користувача до персоналізованих функцій електронного документообігу. Основними критеріями перевірки виступали коректність відображення полів введення, доступність кнопки входу та наявність переходу до реєстрації для нових користувачів. Форма автентифікації користувача наведена на рис. 3.3.

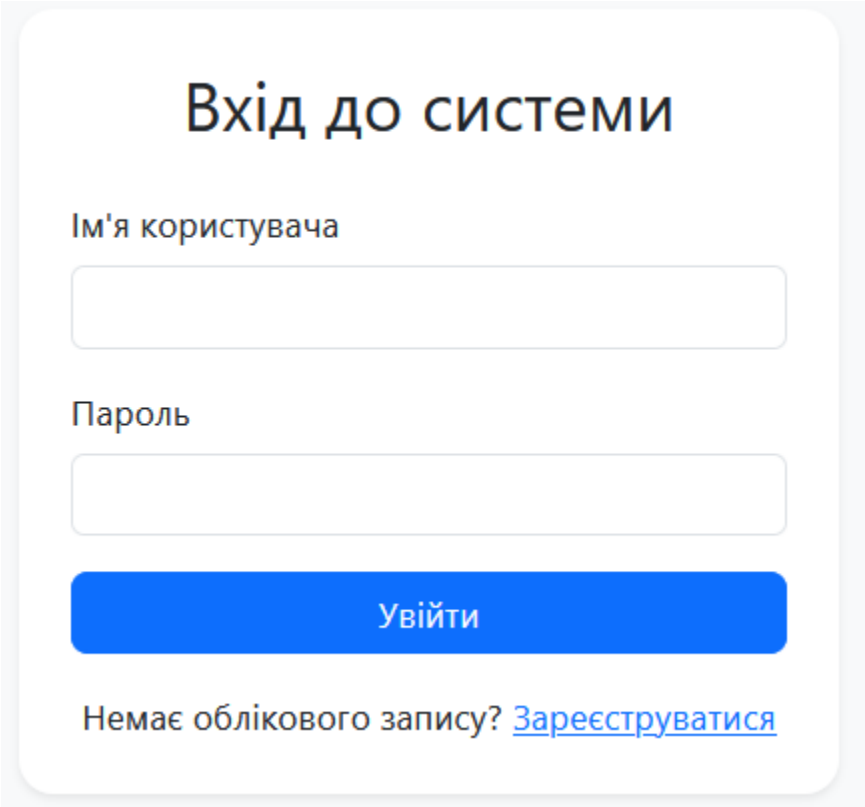


Рисунок 3.3 – Форма автентифікації користувача

Форма входу містить два основні поля: ім’я користувача та пароль. Така структура відповідає стандартному сценарію автентифікації, за якого система перевіряє введені облікові дані та визначає можливість подальшого доступу до функцій вебзастосунку.

Інтерфейс форми є лаконічним і не містить зайвих елементів, що спрощує виконання дії входу. Кнопка «Увійти» розміщена після полів введення та чітко визначає основну операцію сторінки, а посилання «Зареєструватися» забезпечує перехід до створення нового облікового запису. Проведена перевірка підтверджує коректне відображення сторінки автентифікації та її відповідність базовому сценарію входу користувача до системи.

Для перевірки адміністративних можливостей було розглянуто сторінку керування обліковими записами, яка використовується для контролю користувачів системи. Основними критеріями оцінювання виступали коректність відображення списку користувачів, наявність фільтрації за роллю та станом, а також доступність переходу до перегляду детальної інформації про обліковий запис. Сторінку адміністрування користувачів наведено на рис. 3.4.

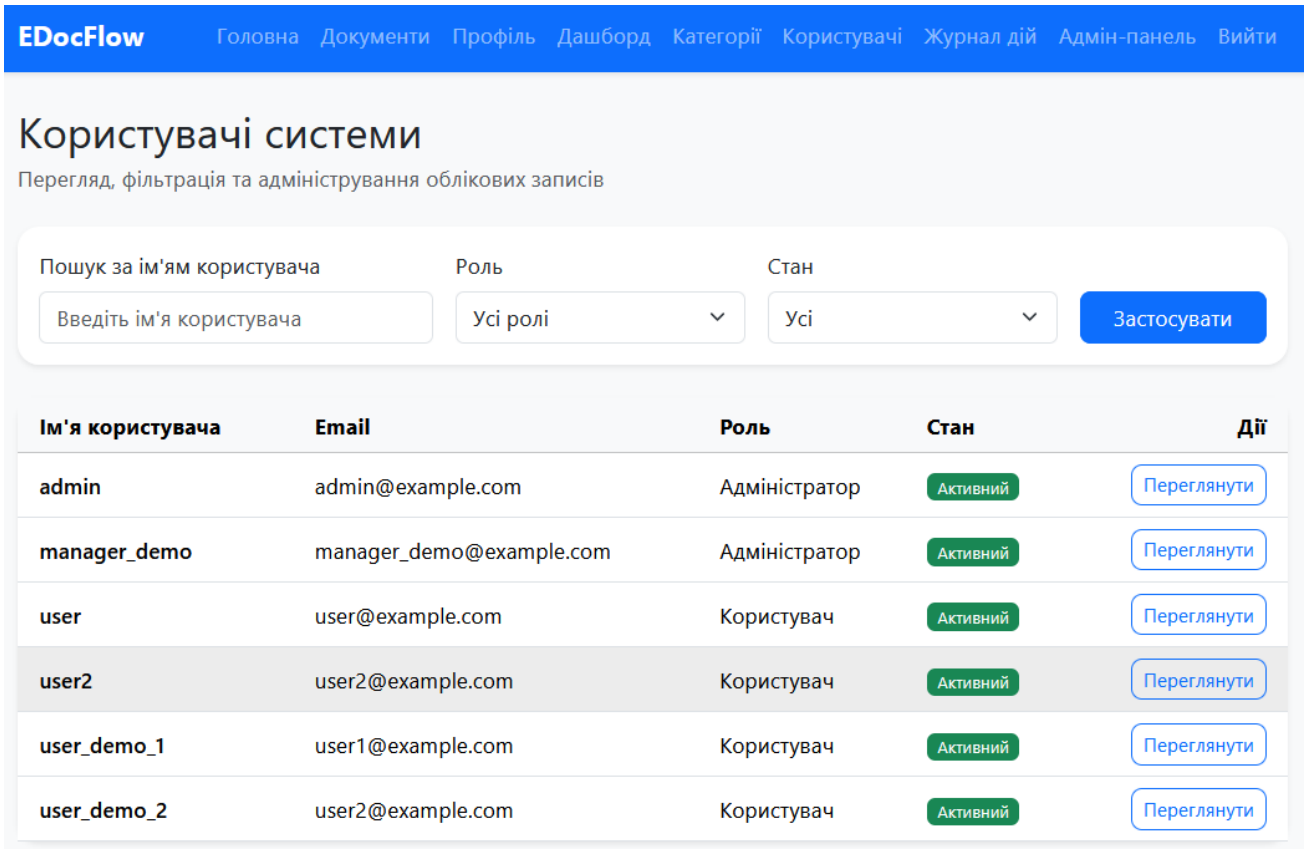


Рисунок 3.4 – Сторінка адміністрування користувачів

Сторінка містить таблицю користувачів із відображенням імені користувача, електронної адреси, ролі, стану облікового запису та доступних дій. Така структура дозволяє адміністратору швидко оцінити поточний склад користувачів системи, визначити їхні повноваження та перевірити активність облікових записів.

У верхній частині сторінки реалізовано блок фільтрації, який дає змогу виконувати пошук за іменем користувача, відбирати записи за роллю та станом. Наявність кнопки «Застосувати» підтверджує можливість керованого формування результатів перегляду. Проведена перевірка свідчить, що сторінка адміністрування

користувачів коректно відображає облікові записи та забезпечує адміністратору базові засоби контролю, пошуку і переходу до подальших дій з користувачами.

Модуль керування документами забезпечує перегляд, пошук і фільтрацію документів системи. Основними критеріями оцінювання були коректність відображення переліку документів, наявність основних атрибутів записів, працездатність фільтрів і доступність переходу до детального перегляду. Сторінку перегляду списку документів наведено на рис. 3.5.

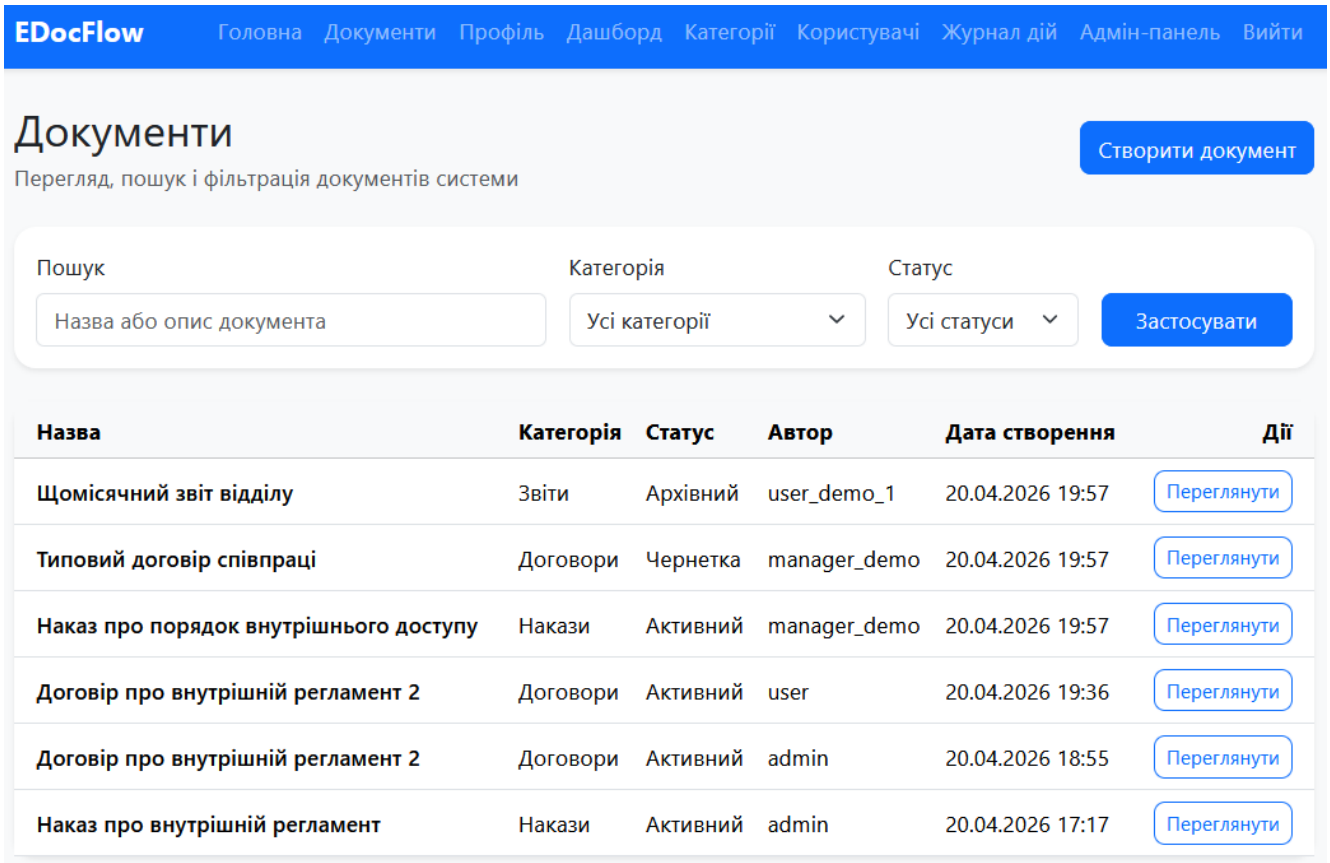


Рисунок 3.5 – Сторінка перегляду списку документів

Сторінка містить заголовок «Документи», коротке пояснення її призначення, кнопку створення нового документа та блок фільтрації. Користувач може виконувати пошук за назвою або описом документа, а також обмежувати результати за категорією та статусом. Це підтверджує реалізацію базових засобів навігації в документному сховищі та спрощує роботу з великою кількістю записів.

Основна частина сторінки представлена таблицею, у якій відображаються назва документа, категорія, статус, автор, дата створення та доступна дія перегляду. Наявність документів із різними категоріями й статусами свідчить про

коректне відображення структурованих даних із бази. Проведена перевірка підтверджує, що сторінка списку документів забезпечує зручний доступ до основних записів системи, підтримує пошук і фільтрацію та дозволяє перейти до подальшої роботи з конкретним документом.

Для сценарію додавання нового документа було проаналізовано форму введення основних атрибутів, яка забезпечує створення запису в системі електронного документообігу. Під час оцінювання враховувалися наявність обов'язкових полів, коректність структури форми, можливість вибору категорії та статусу, а також доступність повернення до попередньої сторінки. Форму створення нового документа наведено на рис. 3.6.

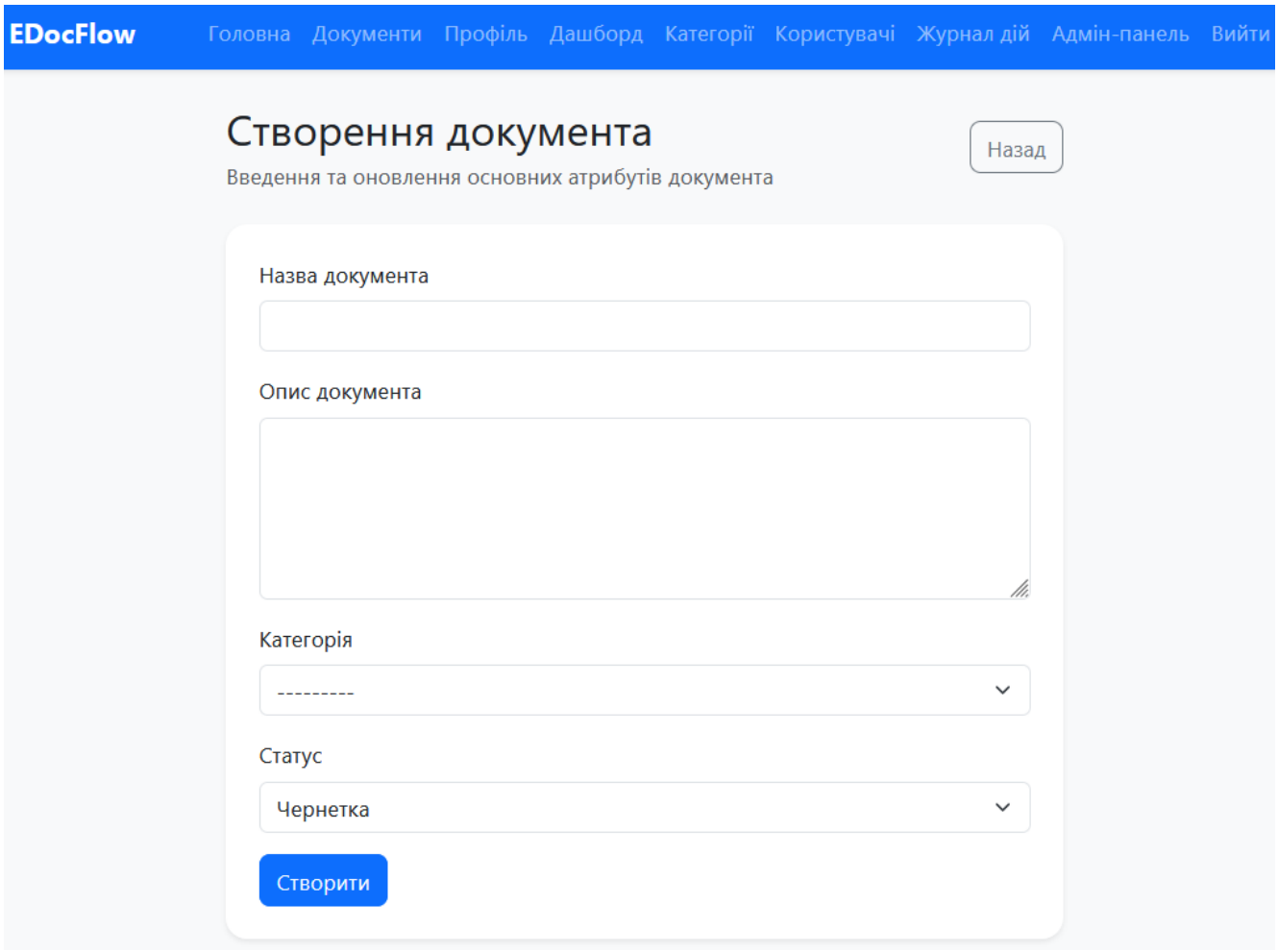


Рисунок 3.6 – Форма створення нового документа

Форма містить основні поля, необхідні для реєстрації документа в системі: назву документа, опис, категорію та статус. Така структура дозволяє не лише створити запис про документ, а й одразу задати його класифікаційну належність і

поточний стан у життєвому циклі. Наявність випадаючих списків для категорії та статусу зменшує ймовірність помилкового введення даних і забезпечує уніфіковане зберігання значень у базі.

Робота з версіями є важливою складовою електронного документообігу, оскільки дозволяє зберігати історію оновлення файлів без втрати попередніх станів документа. Такий механізм забезпечує контроль змін, можливість простеження послідовності завантажень і підтримку актуальності документних матеріалів. Форму завантаження нової версії документа наведено на рис. 3.7.

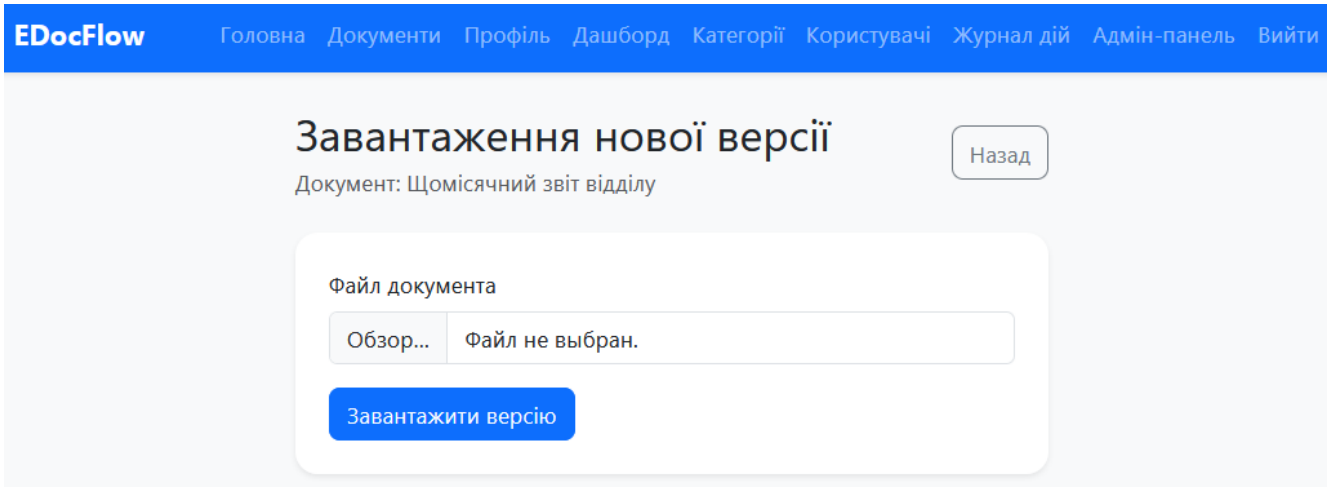


Рисунок 3.7 – Завантаження нової версії документа

Сторінка призначена для додавання нового файлу до вже створеного документа, про що свідчить відображення його назви під заголовком форми. Основним елементом сторінки є поле вибору файлу, через яке користувач може обрати документ із локального носія для подальшого завантаження в систему.

Інтерфейс форми є мінімалістичним і орієнтованим на виконання однієї конкретної дії – завантаження нової версії. Кнопка «Завантажити версію» ініціює передавання файлу на сервер, а кнопка «Назад» дозволяє повернутися до попереднього екрана без виконання операції. Перевірка цієї сторінки підтверджує коректність реалізації сценарію версійності документа та наявність необхідних елементів для додавання нового файлового представлення.

Керування доступом є одним із ключових механізмів захисту документів, оскільки воно визначає, які користувачі можуть переглядати або редагувати конкретний документ. Наявність окремої сторінки для цієї операції дозволяє

адміністратору або власнику документа централізовано контролювати індивідуальні права користувачів. Сторінку керування доступом до документа наведено на рис. 3.8.

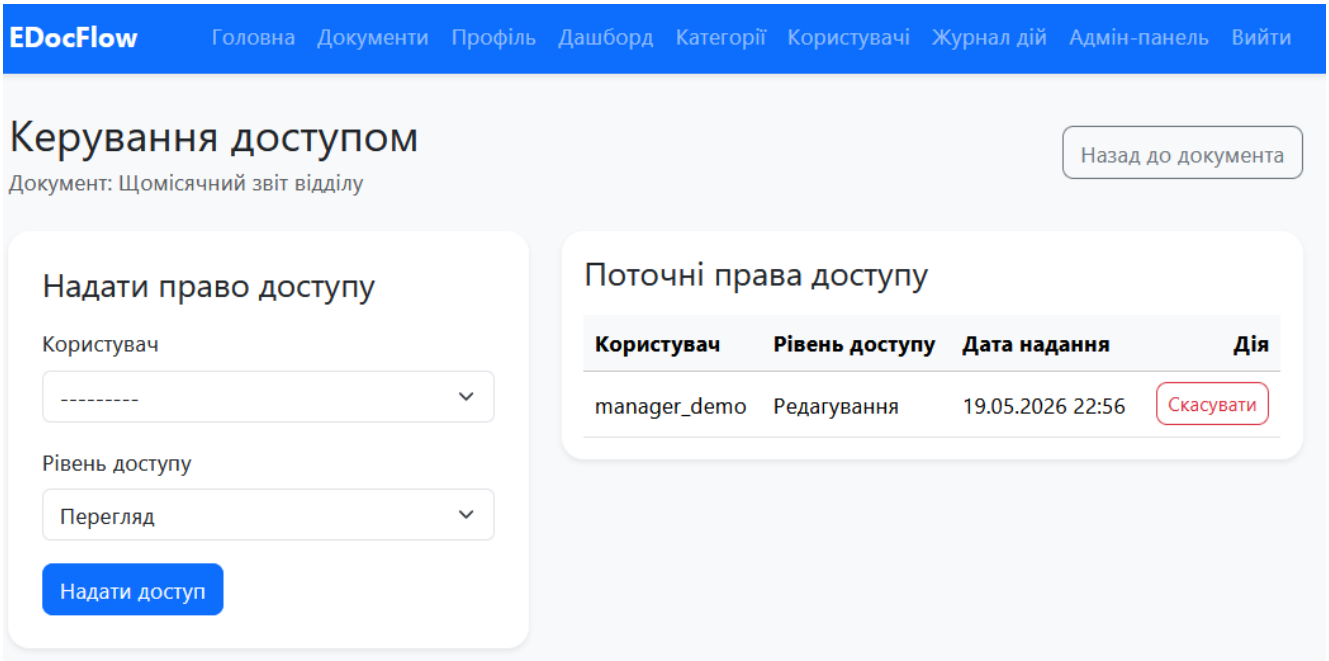


Рисунок 3.8 – Сторінка керування доступом до документа

Сторінка складається з двох основних блоків: форми надання нового права доступу та таблиці поточних прав. У лівій частині інтерфейсу користувач обирає обліковий запис і рівень доступу, зокрема перегляд або редагування, після чого може підтвердити дію кнопкою «Надати доступ». Така структура забезпечує зрозумілий сценарій призначення прав без перевантаження сторінки зайвими елементами.

У правій частині відображаються вже надані права доступу із зазначенням користувача, рівня доступу, дати надання та доступної дії скасування. Наявність кнопки «Скасувати» підтверджує можливість не лише призначати, а й відкликати доступ до документа. Проведена перевірка свідчить, що сторінка коректно відображає поточний стан прав доступу та забезпечує кероване розмежування дій користувачів щодо конкретного документа.

Аналітична складова системи оцінювалася через сторінку інформаційної панелі, яка узагальнює поточний стан електронного документообігу та подає його у вигляді числових показників і графічних представлень. Такий інтерфейс дозволяє

адміністратору швидко отримати зведені дані про користувачів, документи, категорії, версії та події журналу без переходу до окремих розділів. Інформаційну панель системи наведено на рис. 3.9.

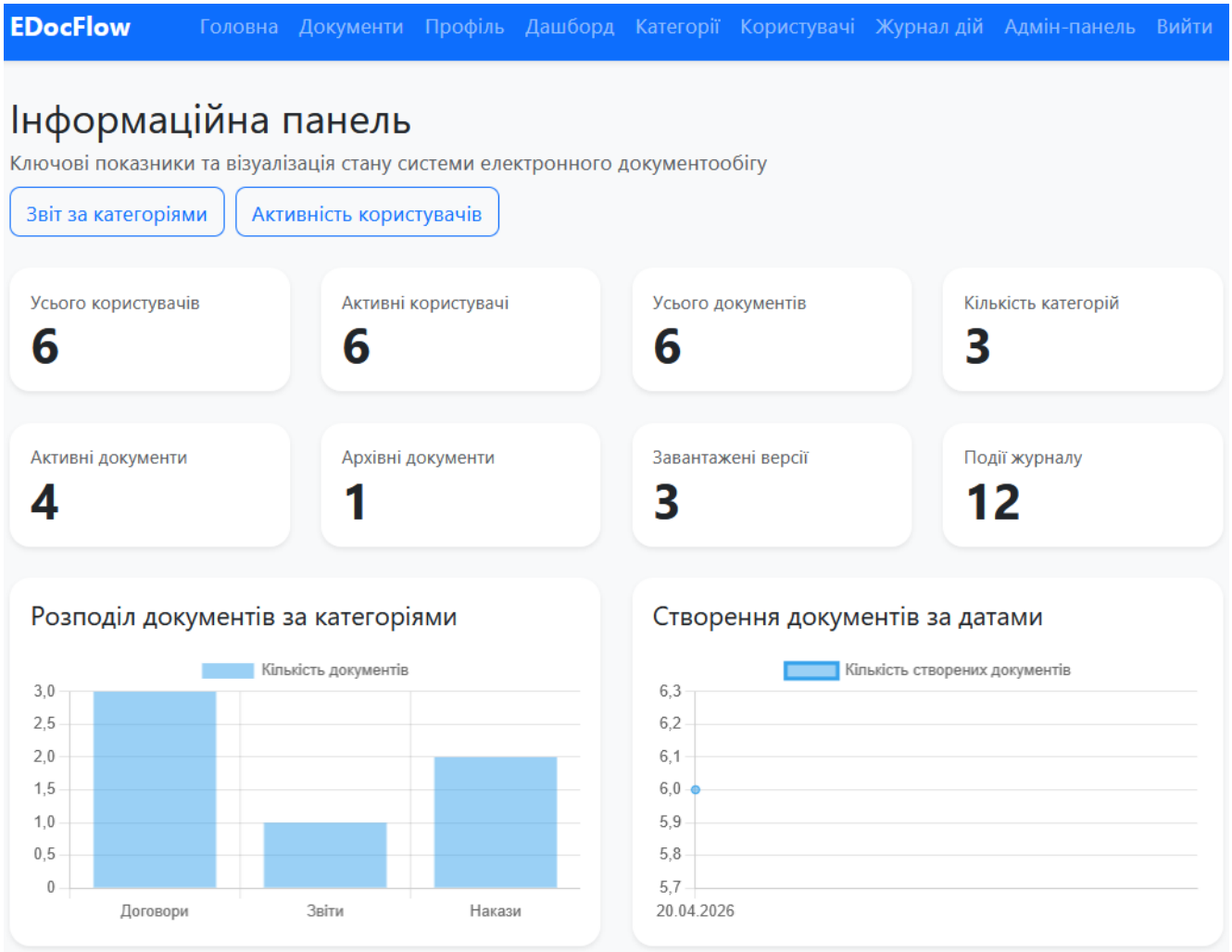


Рисунок 3.9 – Інформаційна панель системи

Сторінка містить набір інформаційних карток із кількісними показниками: загальною кількістю користувачів, активними користувачами, загальною кількістю документів, кількістю категорій, активних і архівних документів, завантажених версій та подій журналу. Таке подання дає змогу оперативно оцінити стан системи та перевірити, чи коректно агрегуються дані з основних таблиць бази даних.

У нижній частині сторінки відображено графіки, які візуалізують розподіл документів за категоріями та створення документів за датами. Наявність графічного подання підтверджує працездатність модуля аналітики та коректну передачу статистичних даних із серверної частини до веб-інтерфейсу. Проведена перевірка свідчить, що інформаційна панель забезпечує цілісне представлення

стану системи та може використовуватися адміністратором для контролю активності електронного документообігу.

Звітна частина системи оцінювалася через сторінку табличного узагальнення документів, яка дає змогу проаналізувати їх розподіл за категоріями та статусами. Звіт по зведенню документів за категоріями наведено на рис. 3.10.

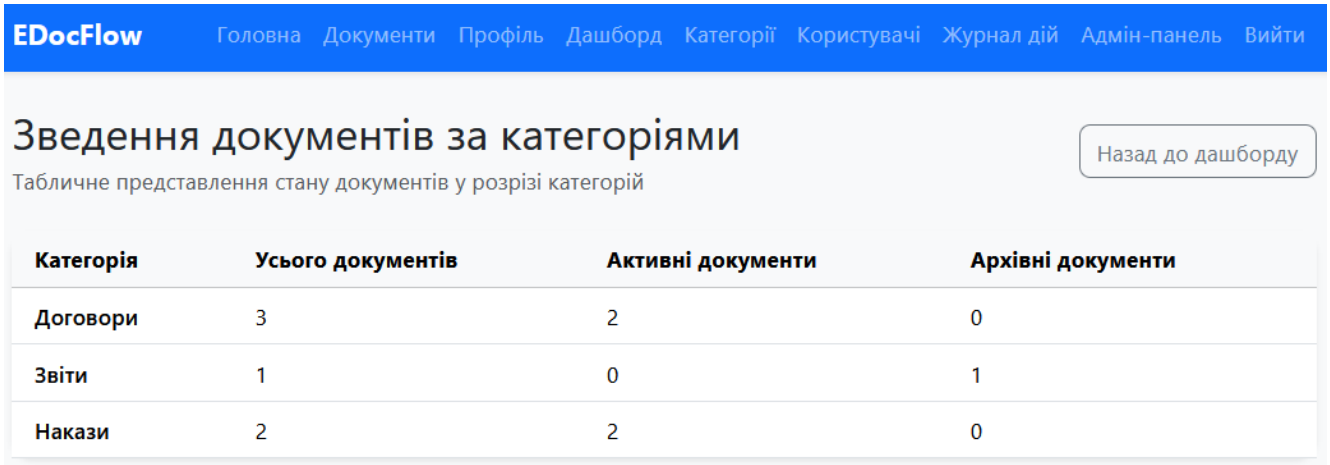


Рисунок 3.10 – Звіт по зведенню документів за категоріями

Звіт подається у вигляді таблиці з такими основними колонками: категорія, загальна кількість документів, кількість активних документів і кількість архівних документів. У таблиці відображено категорії «Договори», «Звіти» та «Накази», для кожної з яких система автоматично підраховує відповідні показники.

Наявність окремого стовпця для архівних документів дозволяє відокремити актуальні матеріали від тих, що вже не перебувають в активному використанні. Кнопка повернення до дашборду забезпечує зручну навігацію між аналітичними сторінками. Отриманий результат підтверджує коректність формування звіту та правильність агрегування даних за категоріями документів.

Звіт активності користувачів (рис. 3.11) використовується для оцінювання інтенсивності роботи облікових записів і контролю виконаних дій у системі. Він формується у вигляді таблиці, де для кожного користувача відображаються ім’я, роль, кількість виконаних дій, кількість завантажених версій і остання дата активності. Ці показники дозволяють адміністратору оцінити, які користувачі активно працювали із системою, а які облікові записи не виконували операцій або не мають зафіксованої активності.

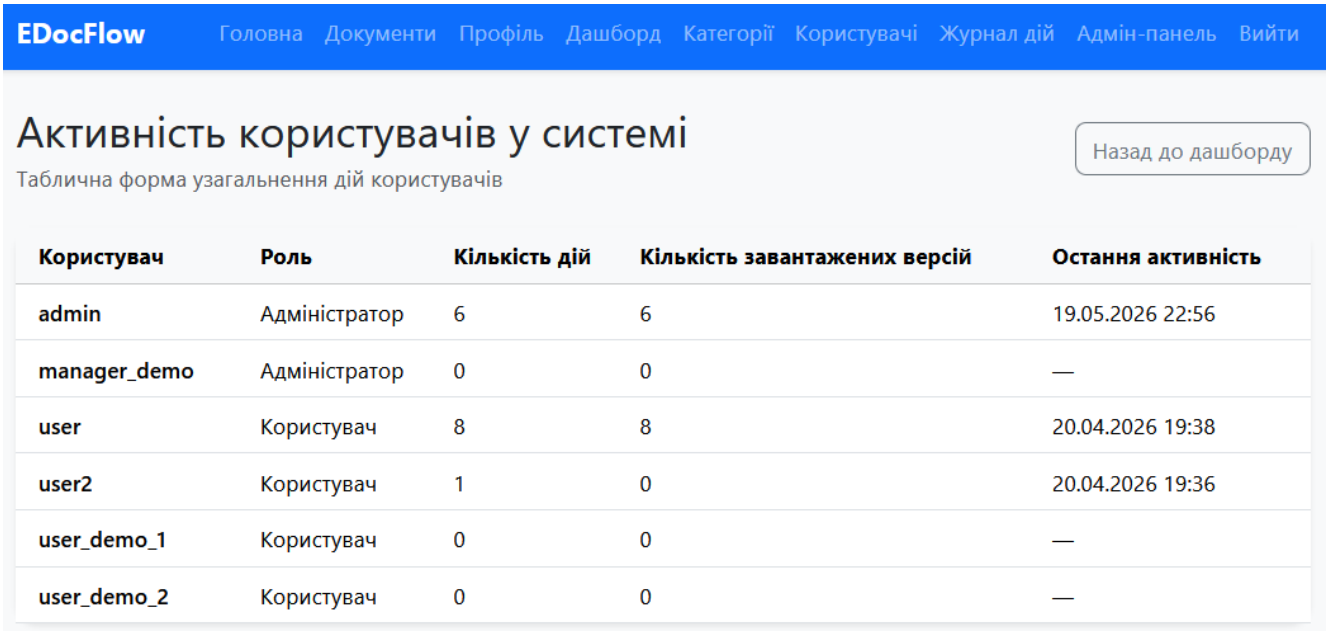


Рисунок 3.11 – Звіт активності користувачів

Таблиця коректно відображає рольову структуру, а значення кількості дій і завантажених версій підтверджують роботу журналу подій. Отже, звіт активності формує достовірну інформацію та придатний для контролю користувачів у середовищі електронного документообігу.

Загалом результати тестування підтвердили працездатність розробленого програмного забезпечення. Система коректно виконує основні операції електронного документообігу, забезпечує розмежування доступу, підтримує фіксації дій і надає адміністративні засоби аналізу даних. Виявлені під час розробки дрібні недоліки, пов’язані переважно з навігацією та відображенням форм, були усунуті на етапі налагодження інтерфейсу. Отримані результати дають підстави вважати розроблену систему придатною для демонстраційної експлуатації в межах поставленого завдання.

4 ОХОРОНА ПРАЦІ

У сучасних умовах охорона праці є важливою складовою розроблення програмного забезпечення, оскільки робота над адаптивною веб-орієнтованою системою електронного документообігу передбачає тривале використання комп’ютера, монітора, клавіатури, маніпулятора, мережевого обладнання та локального серверного середовища. Основна мета охорони праці полягає у створенні безпечних і комфортних умов, які зменшують вплив небезпечних та шкідливих чинників на здоров’я і працездатність фахівця. Для розробника вебзастосунку основними ризиками є зорове навантаження, статична поза, напруження опорно-рухового апарату, психоемоційне навантаження, невідповідні параметри мікроклімату й освітлення, а також потенційна небезпека під час використання електрообладнання.

4.1 Характеристика умов праці програміста

Під час розроблення веб-орієнтованої системи електронного документообігу робота виконавця пов’язана з тривалим використанням комп’ютера, аналізом коду, проєктуванням бази даних, тестуванням інтерфейсу та перевіркою серверної частини. Така діяльність належить до інтелектуальної праці з підвищеним зоровим і нервово-емоційним навантаженням, оскільки потребує концентрації, точності та постійного аналізу технічних рішень.

Основними шкідливими чинниками є напруження зору, статичне навантаження на м’язи шиї, спини й рук, тривале сидяче положення, монотонність роботи, а також можливий вплив шуму, недостатнього освітлення, несприятливого мікроклімату та електрообладнання. Особливе значення має ергономічна організація робочого місця, оскільки неправильне розташування монітора, клавіатури або крісла може спричиняти втому й зниження працездатності.

Для зменшення негативного впливу цих чинників необхідно дотримуватися режиму праці та відпочинку, робити короткі перерви, змінювати робочу позу й виконувати вправи для очей і м’язів. Робоче місце має забезпечувати зручне

розташування монітора, відсутність відблисків, достатнє освітлення та природне положення тіла під час роботи.

4.2 Вимоги до виробничих приміщень

Колірне оформлення приміщення та меблів має забезпечувати комфортне сприйняття інформації з монітора й не створювати додаткового психоемоційного навантаження. Робоче місце слід розташовувати так, щоб природне або штучне світло не потрапляло безпосередньо на екран, а вікно знаходилося збоку від користувача.

Освітлення робочого місця програміста повинно забезпечувати зручну роботу з екраном, кодом і документацією. Недостатня освітленість спричиняє напруження зору та втому, а надмірна яскравість може викликати подразнення очей і відблиски. Для робіт середньої зорової точності рекомендована загальна освітленість становить близько 200 лк, комбінована – 300 лк, а для робіт високої точності – відповідно 300 лк і 750 лк.

Параметри мікроклімату також впливають на працездатність розробника, оскільки комп’ютерна техніка та мережеве обладнання виділяють тепло й можуть знижувати вологість повітря. Тому в приміщенні необхідно підтримувати оптимальні значення температури, вологості та швидкості руху повітря. Рекомендовані параметри мікроклімату для приміщень із комп’ютерною технікою наведено в табл. 4.1.

Таблиця 4.1 – Параметри мікроклімату для приміщень із комп’ютерною технікою

Період року	Параметр мікроклімату	Рекомендоване значення
Холодний	Температура повітря	22–24 °C
Холодний	Відносна вологість	40–60 %
Холодний	Швидкість руху повітря	до 0,1 м/с
Теплий	Температура повітря	23–25 °C
Теплий	Відносна вологість	40–60 %
Теплий	Швидкість руху повітря	0,1–0,2 м/с

Для підтримання зазначених параметрів доцільно використовувати природну або механічну вентиляцію, системи кондиціонування та опалення. Мінімальний об’єм приміщення для одного працівника, який працює з комп’ютерною технікою, має становити не менше 19,5 м³. Обсяг подавання свіжого повітря залежить від об’єму приміщення, що припадає на одну особу, і наведений у табл. 4.2.

Таблиця 4.2 – Норми подавання свіжого повітря для приміщень із комп’ютерною технікою

Характеристика приміщення	Об’ємна витрата свіжого повітря, м³/год на 1 особу
Об’єм до 20 м³ на одну особу	не менше 30
Об’єм від 20 до 40 м³ на одну особу	не менше 20
Об’єм понад 40 м³ на одну особу	допускається природна вентиляція

Під час розроблення програмного забезпечення рівень шуму зазвичай формується роботою системного блока, вентиляторів, принтера, мережевого обладнання або зовнішніх джерел. Надмірний шум знижує концентрацію уваги, підвищує втому та може негативно впливати на нервову систему. Для інтелектуальної праці, пов’язаної з програмуванням, тестуванням і аналізом даних, бажано підтримувати знижений рівень шуму, оскільки така робота належить до напруженої або дуже напруженої за характером зорового й розумового навантаження. Граничні рівні шуму залежно від категорії напруженості та важкості праці наведено в табл. 4.3.

Таблиця 4.3 – Граничні рівні шуму залежно від категорії праці, дБА

Категорія напруженості праці	Легка праця	Середня праця	Важка праця	Дуже важка праця
I. Мало напружена	80	80	75	75
II. Помірно напружена	70	70	65	65
III. Напружена	60	60	–	–
IV. Дуже напружена	50	50	–	–

Розрахунок мінімального об’єму приміщення для одного працівника виконується за формулою:

$$V = S \ h, \tag{4.1}$$

де V – об’єм приміщення, м³; S – площа приміщення, м²; h – висота приміщення, м.

Якщо робоче приміщення має площу S=8 м², а висота становить h=2,7 м, тоді:

$$V = 8 \ \ 2{,}7 = 21{,}6 \text{ м}^3.$$

Отримане значення 21,6 м3 перевищує мінімально рекомендований об’єм 19,5 м³ на одну особу, тому приміщення є допустимим для організації робочого місця програміста. Оскільки об’єм приміщення на одну особу перебуває в межах від 20 до 40 м³, згідно з табл. 4.2 необхідна подача свіжого повітря має становити не менше 20 м³/год на одного працівника.

Ергономічна організація робочого місця програміста необхідна для зниження фізичного навантаження під час тривалої роботи з комп’ютером. Робочий стіл має забезпечувати достатній простір для монітора, клавіатури, миші та документації, мати матову поверхню й відповідати рекомендованій висоті 680–760 мм, а крісло повинно бути регульованим, зі зручною спинкою та висотою сидіння 420–550 мм.

Монітор слід розміщувати в центральній зоні на зручній відстані, клавіатуру – перед користувачем, а мишу – у зоні легкої досяжності. Документи доцільно розташовувати поруч із монітором, щоб зменшити зайві рухи та повороти корпусу.

Оптимальна робоча поза передбачає незначний нахил голови, розслаблені плечі, кут у ліктях близько 80–100° і горизонтальне положення передпліч та кистей. Дотримання цих вимог зменшує ризик м’язового напруження, болю в спині, шиї та руках під час розроблення системи електронного документообігу.

4.3 Розрахунок освітленості і рівня шуму

Для забезпечення безпечних умов праці під час розроблення системи електронного документообігу необхідно оцінити параметри штучного освітлення робочого місця. Оскільки робота програміста пов’язана з тривалим переглядом коду, веб-сторінок, таблиць бази даних і технічної документації, недостатня освітленість може спричиняти зорову втому та зниження концентрації.

Для такого типу робіт доцільно використовувати LED- або люмінесцентні світильники, які забезпечують рівномірний світловий потік і комфортне сприйняття інформації. Нормована освітленість робочої поверхні приймається на рівні E = 300 лк.

Для розрахунку прийнято такі вихідні дані: площа приміщення S = 15 м², довжина A = 5 м, ширина B = 3 м, висота H = 3 м, розрахункова висота підвісу світильника h = 2,2 м. Також враховано коефіцієнт запасу K = 1,4, коефіцієнт нерівномірності Z = 1,1 і коефіцієнт використання світлового потоку η = 0,6.

Індекс приміщення визначається за формулою:

$$i = \frac{S}{h\,(A + B)}. \tag{4.2}$$

Підставимо значення:

$$i = \frac{15}{2.2(5 + 3)} + \frac{15}{17.6} = 0.85\,.$$

Отримане значення індексу приміщення свідчить, що для забезпечення рівномірного освітлення доцільно використовувати кілька світильників, розміщених по площі стелі, а не одне потужне джерело світла. Загальний необхідний світловий потік визначається за формулою:

$$F = \frac{E\ S\ K\ Z}{\eta}. \tag{4.3}$$

Підставимо вихідні дані:

$$F = \frac{300\ \ 15\ \ 1{,}4\ \ 1{,}1}{0{,}6} = \frac{6930}{0{,}6} = 11550\ лм.$$

Отже, для забезпечення нормативного рівня освітленості приміщення площею 15 м² необхідний загальний світловий потік становить приблизно 11550 лм. Для освітлення доцільно використати LED-світильники зі світловим потоком одного світильника F_{CB}:

$$N = \frac{F}{F_{CB}}. \tag{4.4}$$

$$N = \frac{11550}{3300} = 3{,}5.$$

Отримане значення округлюється до більшого цілого числа, тому для приміщення доцільно встановити 4 LED-світильники. Перевіримо фактичну середню освітленість за умови використання чотирьох світильників:

$$E_{Ф} = \frac{N \ F_{CB} \ \eta}{S \ K \ Z}. \tag{4.5}$$

$$E_{Ф} = \frac{4 \ 3300 \ 0.6}{15 \ 1.4 \ 1.1} = \frac{7920}{23.1} = 342.9 \ ЛК.$$

Фактична освітленість становить приблизно 343 лк, що перевищує мінімально необхідне значення 300 лк і є прийнятним для роботи програміста. Таким чином, встановлення 4 LED-світильників зі світловим потоком близько 3300 лм кожен забезпечує достатній рівень освітлення для розроблення та тестування веб-орієнтованої системи електронного документообігу.

Таблиця 4.4 – Вихідні дані для розрахунку освітлення

Показник	Позначення	Значення
Площа приміщення	S	15 м²
Довжина приміщення	A	5 м
Ширина приміщення	B	3 м
Розрахункова висота підвісу	h	2,2 м
Нормована освітленість	E	300 лк
Коефіцієнт запасу	K	1,4
Коефіцієнт нерівномірності	Z	1,1
Коефіцієнт використання світлового потоку	η	0,6
Світловий потік одного LED-світильника	F_{CB}	3300 лм

Окрім освітлення, важливим чинником умов праці є рівень шуму на робочому місці (табл. 4.5). Під час розроблення програмного забезпечення основними джерелами шуму можуть бути системний блок або ноутбук, вентилятори охолодження, клавіатура, миша, принтер та інше периферійне обладнання. Для інтелектуальної праці бажано підтримувати помірний рівень шуму, оскільки надмірний акустичний вплив знижує концентрацію уваги, підвищує втому та ускладнює виконання завдань, пов’язаних з аналізом коду й тестуванням системи.

Таблиця 4.5 – Рівні звукового тиску джерел шуму на робочому місці

Джерело шуму	Рівень шуму, дБ
Системний блок / вентилятор охолодження	35
Монітор	15
Клавіатура	20
Комп’ютерна миша	10
Принтер під час роботи	45

Сумарний рівень шуму від кількох джерел не визначається простим арифметичним додаванням, оскільки децибели є логарифмічною величиною. Тому загальний рівень звукового тиску визначається за формулою:

$$L_{\Sigma} = 10 \lg\left(\sum_{i=1}^{n} 10^{0.1 L_i}\right). \quad (4.6)$$

де L_{Σ} – сумарний рівень шуму, дБ; L_i – рівень шуму окремого джерела, дБ; n – кількість джерел шуму.

Для наявних джерел шуму розрахунок матиме вигляд:

$$L_{\Sigma} = 10 \lg\left(10^{3,5} + 10^{1,5} + 10^{2} + 10^{1} + 10^{4,5}\right) = 45{,}4 \text{ дБ}.$$

Отриманий сумарний рівень шуму становить приблизно 45,4 дБ, що не перевищує допустимих значень для робочого місця з комп’ютерною технікою. При цьому слід враховувати, що принтер не працює постійно, тому фактичний рівень шуму під час програмування, проєктування бази даних і тестування вебзастосунку зазвичай буде нижчим.

4.4 Заходи та засоби протипожежного захисту

Пожежна безпека робочого місця розробника передбачає мінімізацію ризику займання, поширення пожежі та негативного впливу на працівників, обладнання й інформаційні ресурси. Основними джерелами небезпеки є комп’ютерна техніка, блоки живлення, мережеве обладнання, подовжувачі, розетки, зарядні та периферійні пристрої.

Запобігання пожежам передбачає справність електромережі, недопущення перевантаження розеток, використання сертифікованих мережевих фільтрів,

своєчасне вимкнення обладнання та контроль його перегрівання. Вентиляційні отвори техніки не повинні перекриватися, оскільки це може порушити тепловідведення й спричинити займання.

Організаційні заходи включають інструктаж із безпечної роботи з електрообладнанням, ознайомлення з планом евакуації, призначення відповідальних осіб і контроль протипожежного режиму. Забороняється використовувати несправні електроприлади, залишати техніку без нагляду, перевантажувати розетки та зберігати легкозаймисті матеріали біля джерел тепла.

Технічні заходи передбачають справну електропроводку, автоматичні вимикачі, заземлення, перевірку розеток, кабелів і блоків безперебійного живлення. У разі появи ознак перегрівання, запаху гару або нестабільної роботи пристрій необхідно негайно вимкнути з мережі. Для локалізації загоряння в приміщенні з комп’ютерною технікою доцільно використовувати вуглекислотні або порошкові вогнегасники, придатні для електрообладнання під напругою до 1000 В. Використання води є небезпечним через ризик ураження струмом і пошкодження техніки.

У разі пожежі працівник повинен припинити роботу, за можливості вимкнути електроживлення, повідомити відповідальних осіб або пожежну службу та діяти за планом евакуації. Отже, пожежна безпека під час розроблення системи електронного документообігу забезпечується поєднанням організаційних, технічних і експлуатаційних заходів.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розв’язано задачу розроблення доступного веб-орієнтованого засобу для електронного документообігу. Аналіз предметної області показав, що наявні системи часто є складними, функціонально перевантаженими та ресурсомісткими, що ускладнює їх використання в невеликих організаціях і навчальних середовищах.

На основі цього обґрунтовано створення спрощеної адаптивної системи, яка підтримує реєстрацію, автентифікацію, рольове розмежування доступу, керування документами, версійність файлів, надання прав, фіксацію дій, формування дашборду та звітів. У процесі роботи було спроєктовано архітектуру, визначено структуру бази даних, змодельовано основні процеси та реалізовано програмне забезпечення засобами Python, Django, SQLite3, Bootstrap 5 і JavaScript.

Експериментальна перевірка підтвердила коректність основних користувацьких і адміністративних сценаріїв, зокрема реєстрації, входу, роботи з документами, завантаження файлових версій, керування користувачами та доступу до службових сторінок. Тестування показало, що веб-інтерфейс коректно реагує на дії користувачів, форми забезпечують введення потрібних даних, а результати операцій відображаються у відповідних розділах системи.

Також перевірено контроль доступу, фіксації подій та аналітичне подання даних. Система обмежує доступ до документів відповідно до ролі, авторства або індивідуальних прав, фіксує основні події в журналі та відображає узагальнені показники на інформаційній панелі й у звітах, що підтверджує відповідність програмного забезпечення функціональним вимогам.

Практичне значення результатів полягає у можливості використання системи для централізованого зберігання документів, швидкого пошуку, обмеження несанкціонованого доступу та підвищення прозорості дій користувачів. Подальший розвиток системи доцільно пов’язати з реалізацією погодження документів, експорту звітів, електронного підпису, розширенням ролей користувачів і переходом на серверну СУБД.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ustymenko O. Analysis of strategies of IT outsourcing companies based on the concept of dynamic capabilities using the example of «Accenture». *Theoretical and Practical Aspects of the Formation of Scientific Area* : proceedings of the International Scientific Conference. Riga : Publishing House «Baltija Publishing», 2025. P. 34–37. URL: https://doi.org/10.30525/978-9934-26-552-5-23.
2. Трансформація бізнесу для сталого майбутнього: дослідження, цифровізація та інновації : монографія / за ред. О. А. Сороківської. Тернопіль : ФОП Паляниця В. А., 2024. 593 с. URL: https://elartu.tntu.edu.ua/bitstream/lib/46517/1/%D0%9E%D1%81%D1%82%D1%80%20%D0%A6%D1%96%D1%85%20%D0%A8%D0%B5%D1%80%D1%81%D1%82%20%D0%9C%D0%BE%D0%BD%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D1%96%D1%8F_ICBuTS_24.pdf (дата звернення: 28.04.2026).
3. Цифрові трансформації України 2023: виклики та реалії : *зб. наук. пр. за матеріалами IV Круглого столу* (м. Харків, 29 вересня 2023 року) / за ред. С. В. Глібка та ін. Харків : НДІ ПЗІР НАПрН України, 2023. 421 с. URL: https://ndipzir.org.ua/wp-content/uploads/2023/09/conf_29.09.23.pdf (дата звернення: 28.04.2026).
4. Zhang P., Wang Y. Digital transformation: *A systematic review and bibliometric analysis from the corporate finance perspective*. arXiv Preprint. 2024. URL: https://doi.org/10.48550/arXiv.2412.19817.
5. Про електронні документи та електронний документообіг : Закон України від 22.05.2003 № 851-IV. URL: https://zakon.rada.gov.ua/laws/show/851-15#Text (дата звернення: 28.04.2026).
6. Bizi M. K., Ghazali A. M., Yunus M. N. The Adoption of Electronic Records Management System in an Organization: A Systematic Literature Review Approach. *Journal of Information and Knowledge Management*. 2025. Vol. 15, No. 12. P. 280–292. URL: https://doi.org/10.24191/jikm.v15iSI2.8299.

7. Awang Gani D. H., Abd Kadir I. K., Masrek M. N., Ab Rahman A. An evaluation of electronic document management system functionalities and effectiveness in Malaysia. *European Proceedings of Social and Behavioural Sciences*. 2024. URL: https://doi.org/10.15405/epsbs.2024.05.50.
8. Parlakkiliç A. Evaluating the effects of responsive design on the usability of academic websites in the pandemic. *Education and Information Technologies*. 2022. Vol. 27, No. 1. P. 307–322. URL: https://doi.org/10.1007/s10639-021-10650-9.
9. Web Content Accessibility Guidelines (WCAG) 2.2. URL: https://www.w3.org/TR/WCAG22/ (дата звернення: 28.04.2026).
10. Kristić M., Zakarija I., Škopljanac-Mačina F., Car Ž. Machine Learning for Adaptive Accessible User Interfaces: *Overview and Applications.* Applied Sciences. 2025. Vol. 15, No. 23.
11. CERT-UA recorded 4,315 cyber incidents in 2024. URL: https://cip.gov.ua/en/news/cert-ua-minulogo-roku-opracyuvala-4315-kiberincidentiv (дата звернення: 28.04.2026).
12. The NIST Cybersecurity Framework (CSF) 2.0. URL: https://doi.org/10.6028/NIST.CSWP.29 (дата звернення: 28.04.2026).
13. Mohamed A. K., Auer D., Hofer D., Küng J. A systematic literature review of authorization and access control requirements and current state of the art for different database models. *International Journal of Web Information Systems.* 2024. Vol. 20, No. 1. P. 1–23. URL: https://doi.org/10.1108/IJWIS-04-2023-0072.
14. Semantha F. H., Azam S., Shanmugam B., Yeo K. C. Pbdinehr: A novel privacy by design developed framework using distributed data storage and sharing for secure and scalable electronic health records management. *Journal of Sensor and Actuator Networks.* 2023. Vol. 12, No. 2. URL: https://doi.org/10.3390/jsan12020036.
15. What is M-Files? URL: https://www.gartner.com/reviews/product/m-files (дата звернення: 28.04.2026).

16. M-Files Heightens User Experience with Enhanced Desktop User Interface. URL: https://www.m-files.com/press-releases/m-files-heightens-user-experience-with-enhanced-desktop-user-interface/ (дата звернення: 28.04.2026).
17. DocuWare Reviews & Product Details. URL: https://www.g2.com/products/docuware/reviews (дата звернення: 28.04.2026).
18. Solution Forms de DocuWare. URL: https://start.docuware.com/fr/docuware-forms (дата звернення: 28.04.2026).
19. Exploring the Architecture of LogicalDOC Document Management System. URL: https://www.logicaldoc.com/product/architecture (дата звернення: 28.04.2026).
20. About LogicalDOC. URL: https://www.softwareadvice.com.au/software/45365/logicaldoc (дата звернення: 28.04.2026).
21. Томашевський О. М., Цегелик Г. Г., Вітер М. Б., Дубук В. І. Інформаційні технології та моделювання бізнес-процесів : навч. посіб. Київ : Центр учбової літератури, 2012. 296 с. URL: http://moodle.nati.org.ua/pluginfile.php/14680/mod_resource/content/1/tomashevskii_o_m_ta_in_informaciini_tehnologii_ta_modelyuvan.pdf (дата звернення: 28.04.2026).
22. Ganesh R., Prabu G. Determination of Internet Banking Usage and Purpose with Explanation of Data Flow Diagram and Use Case Diagram. *International Journal of Management and Humanities.* 2020. Vol. 4, No. 7. P. 52–58. URL: https://www.researchgate.net/publication/342887376_Determination_of_Internet_Banking_Usage_and_Purpose_with_Explanation_of_Data_Flow_Diagram_and_Use_Case_Diagram (дата звернення: 28.04.2026).

ДОДАТКИ

Додаток А. Технічне завдання

1. Вступ

1.1 Назва проєкту

Адаптивна веб-орієнтована система електронного документообігу.

1.2 Цілі проєкту

Метою проєкту є розроблення адаптивної веб-орієнтованої системи для автоматизації процесів реєстрації, зберігання, оброблення, пошуку, версіонування та контролю використання електронних документів у межах організації. Система повинна забезпечити централізоване керування документами, розмежування прав доступу між користувачами, ведення журналу дій, формування аналітичної інформації та зручну роботу через веб-інтерфейс на різних типах пристроїв.

1.3 Задачі проєкту

Розробка структури бази даних для зберігання інформації про користувачів, документи, категорії, версії файлів, права доступу та журнал дій.

Реалізація механізмів реєстрації, автентифікації користувачів, завершення сеансу роботи та рольового розмежування доступу.

Створення функціоналу адміністрування користувачів із можливістю перегляду облікових записів, зміни ролей, блокування та розблокування користувачів.

Реалізація засобів створення, редагування, перегляду, пошуку, фільтрації, архівації та видалення електронних документів.

Забезпечення можливості завантаження файлів документів, збереження кількох версій одного документа та автоматичної фіксації метаданих файлу.

Розробка механізму надання та скасування індивідуальних прав доступу до документів із підтримкою рівнів перегляду та редагування.

Реалізація журналу системних подій для фіксації дій користувачів, пов’язаних із входом, створенням документів, зміною доступу, додаванням версій та адміністративними операціями.

Побудова інформаційної панелі та звітів для відображення статистики щодо користувачів, документів, категорій, версій і активності в системі.

Забезпечення адаптивного веб-інтерфейсу з використанням сучасних засобів HTML, CSS, Bootstrap 5 та JavaScript.

Проведення тестування основних функціональних сценаріїв і перевірка відповідності програмного забезпечення поставленим вимогам.

2. Опис поточного стану

2.1 Опис існуючих процесів

На даний момент у багатьох організаціях робота з електронними документами часто здійснюється за допомогою окремих файлів, локальних папок, електронної пошти або нескладних хмарних сховищ. Документи можуть зберігатися без єдиної структури, а їх пошук, передавання іншим користувачам, оновлення та контроль актуальності виконуються вручну. У таких умовах ускладнюється визначення останньої версії документа, перевірка історії змін, контроль доступу до файлів та аналіз дій користувачів.

Відсутність єдиної веб-орієнтованої системи призводить до того, що кожен користувач може організовувати роботу з документами по-різному, що знижує узгодженість документообігу. Частина документів може дублюватися, зберігатися у різних місцях або передаватися без фіксації факту доступу. Формування звітів про кількість документів, активність користувачів, стан категорій або історію завантаження версій зазвичай потребує ручного збирання даних, що збільшує витрати часу та підвищує ризик помилок.

2.2 Основні проблеми та недоліки

Відсутність централізованого сховища електронних документів ускладнює їх систематизацію, пошук і подальше використання.

Зберігання документів у різних папках або окремих файлах може призводити до дублювання інформації та втрати актуальної версії документа.

Ручне передавання документів між користувачами не забезпечує належного контролю доступу та не дозволяє чітко визначити, хто має право перегляду або редагування.

Відсутність механізму версійності ускладнює відстеження змін, внесених до документа, і не дозволяє зберігати історію його оновлень.

Недостатній рівень захисту інформації виникає через відсутність авторизації, рольового розмежування доступу та серверної перевірки прав користувачів.

Відсутність журналу дій не дозволяє відстежувати операції користувачів, зокрема створення документів, редагування, завантаження нових версій або зміну доступу.

Ручне формування аналітичних відомостей і звітів потребує додаткового часу та не забезпечує оперативного отримання узагальненої інформації про стан документообігу.

Відсутність адаптивного веб-інтерфейсу обмежує зручність роботи з документами на різних пристроях, зокрема ноутбуках, планшетах і смартфонах.

3. Вимоги до функціональності

3.1 Основні функції

Реєстрація користувачів у системі з автоматичним призначенням базової ролі користувача.

Автентифікація користувачів через введення логіна та пароля.

Завершення сеансу роботи користувача шляхом виходу з облікового запису.

Підтримка двох основних ролей: адміністратора та користувача.

Адміністрування облікових записів із можливістю перегляду користувачів, зміни ролей, блокування та розблокування.

Створення, редагування, перегляд, архівація та видалення електронних документів.

Ведення довідника категорій документів із можливістю додавання, редагування та видалення категорій адміністратором.

Завантаження файлів документів на сервер із фіксацією назви, розміру, типу файлу, дати завантаження та користувача, який виконав операцію.

Збереження кількох версій одного документа з автоматичним присвоєнням номера версії.

Пошук і фільтрація документів за назвою, описом, категорією та статусом.

Перегляд детальної інформації про документ, включаючи його основні атрибути, історію версій і надані права доступу.

Надання та скасування індивідуальних прав доступу до документа для окремих користувачів.

Підтримка рівнів доступу до документа, зокрема перегляду та редагування.

Ведення журналу дій користувачів із фіксацією основних подій системи.

Формування інформаційної панелі з узагальненими показниками щодо користувачів, документів, категорій, версій і журналу подій.

Формування табличних звітів, зокрема звіту про зведення документів за категоріями та звіту активності користувачів.

3.2 Додаткові функції

Відображення службових повідомлень про успішне виконання операцій або помилки введення даних.

Автоматичне перенаправлення користувача після виконання основних дій, зокрема після реєстрації, створення документа або завантаження нової версії.

Відображення статусів документів у зручному для користувача вигляді: чернетка, активний, архівний.

Автоматичне збереження метаданих завантаженого файлу без ручного введення технічних параметрів.

Використання адаптивного інтерфейсу для коректного відображення сторінок на персональних комп’ютерах, ноутбуках, планшетах і смартфонах.

Візуалізація аналітичних даних за допомогою графіків на інформаційній панелі.

Серверна перевірка прав доступу до документів, редагування, додавання версій і керування доступом.

Можливість повернення до попередніх сторінок через навігаційні елементи інтерфейсу.

3.3 Обмеження та припущення

Система розробляється як веб-орієнтований застосунок із використанням мови програмування Python та фреймворку Django.

База даних реалізується за допомогою SQLite3, що є достатнім для навчального, демонстраційного або локального використання системи.

Взаємодія з базою даних здійснюється через Django ORM без необхідності ручного написання SQL-запитів для більшості операцій.

Інтерфейс користувача реалізується з використанням HTML5, CSS3, Bootstrap 5 та JavaScript.

Для побудови графічних елементів інформаційної панелі може використовуватися клієнтська бібліотека Chart.js.

Завантажені файли документів зберігаються у файловій системі сервера, а в базі даних зберігаються їхні метадані та шлях до файлу.

Система передбачає роботу через сучасний веб-браузер і не потребує встановлення окремого клієнтського програмного забезпечення.

На поточному етапі система орієнтована на локальне або навчальне середовище експлуатації та не передбачає промислового навантаження великої організації.

Мінімальні вимоги до обладнання: процесор із тактовою частотою не нижче 2,0 ГГц, оперативна пам’ять не менше 4 ГБ, вільний простір на диску не менше 1 ГБ для проєкту, бази даних і завантажених файлів.

Для базового використання системи необхідне локальне серверне середовище Django та веб-браузер; постійне підключення до Інтернету не є обов’язковим, якщо система розгортається локально.

4. Вимоги до інтерфейсу користувача

4.1 Вимоги до зручності використання

Інтерфейс користувача адаптивної веб-орієнтованої системи електронного документообігу має бути логічно структурованим, зрозумілим і зручним для

швидкого виконання основних операцій із документами. Користувач повинен мати можливість без тривалого навчання виконувати реєстрацію, вхід до системи, перегляд доступних документів, створення нових записів, завантаження версій, пошук, фільтрацію та перегляд звітної інформації.

Основні вимоги до зручності використання включають:

- наявність головної сторінки системи з коротким описом її призначення та переходами до реєстрації й автентифікації;
- наявність навігаційного меню з доступом до основних розділів: документи, категорії, користувачі, журнал дій, інформаційна панель і звіти;
- розмежування видимості пунктів меню залежно від ролі користувача, щоб звичайний користувач не мав доступу до адміністративних функцій;
- використання єдиного стилю оформлення сторінок із застосуванням Bootstrap 5, спільної палітри кольорів, однакових типів кнопок, форм і таблиць;
- чітке підписування полів введення, кнопок і посилань, щоб користувач однозначно розумів призначення кожного елемента інтерфейсу;
- наявність форм для створення та редагування документів із полями для введення назви, опису, вибору категорії та статусу;
- використання випадаючих списків для вибору категорії, статусу документа, користувача та рівня доступу з метою зменшення кількості помилок введення;
- реалізація пошуку та фільтрації документів за назвою, описом, категорією та статусом;
- відображення списків документів, користувачів, журналу дій і звітів у вигляді структурованих таблиць;
- наявність зрозумілих повідомлень про успішне виконання операцій або помилки введення без надмірного технічного змісту;
- підтримка адаптивного відображення сторінок на персональних комп’ютерах, ноутбуках, планшетах і смартфонах;

− відсутність необхідності горизонтальної прокрутки на стандартних екранах ноутбуків і персональних комп’ютерів;

− візуальне виділення основних дій, зокрема створення, збереження, видалення, архівації, надання доступу та завантаження версії документа;

− наявність кнопок повернення або переходу до попередніх сторінок після виконання основних операцій;

− відображення аналітичних даних у вигляді інформаційних карток, графіків і табличних звітів.

Інтерфейс системи має забезпечувати не лише доступ до функцій, а й послідовну логіку роботи користувача з документами. Реалізація таких вимог дозволить зменшити кількість помилкових дій, підвищити зручність використання вебзастосунку та забезпечити комфортну роботу з електронними документами в різних умовах експлуатації.

5. Вимоги до продуктивності

Адаптивна веб-орієнтована система електронного документообігу повинна забезпечувати стабільну обробку користувацьких запитів у межах локального або навчального середовища експлуатації. Оскільки система працює з документами, користувачами, правами доступу, версіями файлів, журналом подій та аналітичними звітами, продуктивність має забезпечувати комфортну роботу без суттєвих затримок під час виконання типових операцій.

Основні вимоги до продуктивності включають:

− час відкриття основних сторінок вебзастосунку, зокрема головної сторінки, списку документів, сторінки входу, реєстрації та детального перегляду документа, не повинен перевищувати 2 секунд у стандартному локальному середовищі;

− час виконання пошуку та фільтрації документів за назвою, описом, категорією або статусом не повинен перевищувати 3 секунд при роботі з типовим обсягом демонстраційних даних;

− час створення нового документа або редагування його основних атрибутів не повинен перевищувати 2 секунд за умови коректного заповнення форми;

− час завантаження нової версії документа залежить від розміру файлу, однак для файлів невеликого та середнього обсягу операція повинна виконуватися без відчутних затримок для користувача;

− час відкриття сторінки керування доступом до документа не повинен перевищувати 2–3 секунд, включаючи завантаження переліку користувачів і вже наданих прав доступу;

− формування журналу дій із можливістю перегляду та фільтрації подій повинно виконуватися протягом 3 секунд при роботі з типовою кількістю записів навчального проєкту;

− формування інформаційної панелі з кількісними показниками, графіками та агрегованими даними повинно виконуватися протягом 5 секунд;

− формування табличних звітів, зокрема зведення документів за категоріями та звіту активності користувачів, повинно виконуватися протягом 5 секунд;

− система повинна коректно працювати при одночасній роботі кількох користувачів у локальному середовищі без помітного зниження швидкості виконання основних операцій;

− програмне забезпечення повинно раціонально використовувати ресурси комп’ютера, не створюючи надмірного навантаження на процесор, оперативну пам’ять і файлову систему під час стандартної роботи;

− розмір завантажуваних файлів документів повинен обмежуватися відповідно до налаштувань серверного середовища, щоб уникнути надмірного навантаження на файлове сховище;

− робота з базою даних повинна виконуватися через Django ORM із використанням оптимальних запитів, що дає змогу уникнути надмірного дублювання операцій вибірки та збереження даних.

Дотримання зазначених вимог до продуктивності дозволить забезпечити зручну роботу користувачів із документами, швидке виконання основних операцій і стабільне функціонування системи в умовах навчального або локального використання. Для подальшого розширення системи та роботи з більшими обсягами даних доцільно передбачити можливість переходу від SQLite3 до серверної СУБД, наприклад PostgreSQL або MySQL.

6. Вимоги до безпеки

Адаптивна веб-орієнтована система електронного документообігу повинна забезпечувати захист облікових записів користувачів, документів, файлових версій, журналу дій та аналітичної інформації. Оскільки система працює з електронними документами організації, особливу увагу необхідно приділити автентифікації, авторизації, контролю доступу до документів і серверній перевірці прав користувачів .

6.1 Автентифікація та авторизація користувачів

Доступ до функціональних можливостей системи повинен надаватися лише після успішного входу користувача за логіном і паролем.

Для роботи з обліковими записами необхідно використовувати стандартні механізми Django Auth, які забезпечують автентифікацію, створення сесій і збереження паролів у хешованому вигляді.

У системі повинні підтримуватися дві основні ролі: адміністратор і користувач.

Роль користувача повинна визначати доступний набір функцій, зокрема можливість адміністрування облікових записів, керування документами, перегляду журналу дій, дашборду та звітів.

Адміністративні сторінки повинні бути доступні лише користувачам із відповідними повноваженнями.

Заблокований користувач не повинен мати можливості виконати вхід до системи до моменту розблокування адміністратором.

6.2 Захист від несанкціонованого доступу

Доступ до документів повинен перевірятися на серверному рівні, а не лише шляхом приховування кнопок або посилань в інтерфейсі.

Користувач повинен мати доступ лише до тих документів, які він створив самостійно або до яких йому було надано право перегляду чи редагування.

Операції редагування документа, архівації, завантаження нової версії та керування доступом повинні виконуватися лише після перевірки ролі, авторства або індивідуально наданих прав.

Паролі користувачів не повинні зберігатися у відкритому вигляді; для їх збереження має використовуватися стандартне хешування Django.

У разі спроби доступу до забороненої функції система повинна перенаправляти користувача на дозволену сторінку або виводити зрозуміле повідомлення про відсутність прав.

Журнал дій повинен фіксувати основні операції користувачів, зокрема реєстрацію, вхід, вихід, створення документів, додавання версій, зміну ролей, блокування користувачів і керування доступом.

6.3 Зберігання та передача даних

Структуровані дані системи повинні зберігатися в базі даних SQLite3, а файли документів – у файловому сховищі сервера.

У базі даних повинні зберігатися метадані файлів, зокрема назва, розмір, тип, дата завантаження та користувач, який виконав завантаження.

Фізичний доступ до серверного середовища, бази даних і каталогу з медіафайлами повинен бути обмежений.

Під час локального використання системи передача даних здійснюється в межах середовища розгортання Django; у разі подальшого розгортання у відкритій мережі доцільно використовувати HTTPS.

Система повинна зберігати лише ті дані, які необхідні для виконання функцій електронного документообігу, без надмірного накопичення службової або персональної інформації.

Резервне копіювання бази даних і каталогу завантажених документів може розглядатися як додатковий засіб підвищення надійності та захисту інформації.

7. Вимоги до інтеграції

На поточному етапі адаптивна веб-орієнтована система електронного документообігу розглядається як самостійний програмний продукт, призначений для локального або навчального використання. Разом із тим структура системи повинна передбачати можливість подальшого розширення та інтеграції з іншими інформаційними сервісами організації.

Основні вимоги до інтеграції включають:

− система повинна мати модульну структуру Django-проєкту, що дозволяє окремо розвивати модулі користувачів, документів, журналу дій, аналітики та звітності;

− архітектура програмного забезпечення повинна допускати подальший перехід від SQLite3 до серверної СУБД, наприклад PostgreSQL або MySQL;

− структура бази даних повинна забезпечувати можливість додавання нових ролей користувачів, типів документів, статусів і рівнів доступу;

− механізм роботи з документами повинен допускати подальше розширення функціями погодження, електронного підпису, експорту звітів або інтеграції з поштовими сервісами;

− формування аналітичних даних повинно базуватися на запитах до бази даних, що дозволить у майбутньому реалізувати додаткові графіки, фільтри та звіти;

− система повинна використовувати стандартні веб-технології, що полегшує її подальше розгортання на сервері або інтеграцію з іншими вебзастосунками;

− у разі подальшого розвитку може бути передбачено API для обміну даними з іншими інформаційними системами організації.

8. Вимоги до тестування

Тестування адаптивної веб-орієнтованої системи електронного документообігу є обов’язковим етапом перевірки її працездатності, відповідності поставленим вимогам і готовності до використання в локальному або навчальному

середовищі. Перевірка повинна охоплювати основні сценарії роботи користувача й адміністратора, коректність взаємодії між модулями системи, обробку помилкових даних, захист доступу до документів і правильність формування аналітичної інформації.

8.1 Типи тестування

Функціональне тестування: перевірка реалізації основних функцій системи, зокрема реєстрації користувачів, автентифікації, виходу із системи, створення документів, редагування, архівації, видалення, завантаження версій, пошуку, фільтрації, керування доступом, перегляду журналу дій, дашборду та звітів.

Інтеграційне тестування: перевірка коректності взаємодії між веб-інтерфейсом, представленнями Django, формами, ORM-моделями, базою даних SQLite3 та файловим сховищем документів.

Тестування безпеки: перевірка роботи автентифікації та авторизації, недопущення доступу неавторизованих користувачів до захищених сторінок, перевірка рольового розмежування доступу, блокування користувачів і серверного контролю прав на перегляд, редагування та керування документами.

Тестування продуктивності: оцінювання часу завантаження основних сторінок, виконання пошуку й фільтрації документів, створення записів, завантаження файлових версій, відкриття інформаційної панелі та формування табличних звітів.

Тестування зручності використання: оцінка зрозумілості навігації, логічності розміщення елементів інтерфейсу, коректності відображення форм, таблиць, повідомлень, кнопок і адаптивності сторінок на різних розмірах екрана.

Тестування коректності даних: перевірка збереження документів, категорій, версій, прав доступу та записів журналу дій у базі даних відповідно до виконаних операцій користувача.

8.2 Критерії приймання робіт

Успішне виконання основних функціональних сценаріїв без критичних помилок.

Відповідність реалізованих модулів системи функціональним вимогам, визначеним у технічному завданні.

Коректна робота реєстрації, автентифікації, виходу із системи та блокування облікових записів.

Правильне розмежування доступу між адміністратором і звичайним користувачем.

Коректне створення, редагування, архівація, видалення та перегляд електронних документів.

Успішне завантаження нових версій документів із фіксацією їхніх метаданих.

Коректне надання та скасування прав доступу до документів для окремих користувачів.

Стабільне відображення журналу дій із фіксацією основних подій системи.

Правильне формування інформаційної панелі, графіків і табличних звітів на основі актуальних даних.

Коректна обробка помилкових або неповних даних без аварійного завершення роботи системи.

Стабільна робота веб-інтерфейсу в сучасному браузері без порушення структури сторінок.

Виконання основних операцій у межах прийнятного часу відгуку, визначеного у вимогах до продуктивності.

9. План впровадження

Впровадження адаптивної веб-орієнтованої системи електронного документообігу передбачає послідовне виконання дій, спрямованих на підготовку програмного середовища, перевірку працездатності системи, початкове наповнення даними та ознайомлення користувачів із базовими сценаріями роботи. Оскільки система орієнтована на локальне або навчальне використання, процес впровадження може бути виконаний без складної серверної інфраструктури.

9.1 Підготовка до запуску

Встановлення необхідного програмного середовища, зокрема Python, Django та залежностей проєкту.

Розгортання вебзастосунку в локальному середовищі або на вибраному сервері.

Створення та застосування міграцій бази даних для формування необхідних таблиць.

Налаштування параметрів роботи з базою даних SQLite3, статичними файлами та каталогом для завантажених документів.

Створення адміністративного облікового запису для початкового керування системою.

Проведення контрольного тестування основних сценаріїв перед передачею системи користувачам.

9.2 Навчання користувачів

Ознайомлення адміністратора з функціями керування користувачами, ролями, блокуванням облікових записів, категоріями документів, журналом дій, дашбордом і звітами.

Ознайомлення звичайних користувачів із процесами реєстрації, входу до системи, перегляду доступних документів, створення документа, завантаження нової версії та роботи з наданими правами доступу.

Пояснення правил безпечної роботи із системою, зокрема необхідності збереження конфіденційності пароля, виходу з облікового запису після завершення роботи та уважного надання прав доступу до документів.

Надання короткої інструкції користувача у вигляді текстового або PDF-документа з описом основних операцій у системі.

9.3 Підтримка після впровадження

Після запуску системи необхідно забезпечити спостереження за її роботою, виявлення можливих помилок і оперативне внесення незначних виправлень. Особливу увагу слід приділяти коректності збереження документів, роботі файлового сховища, стабільності авторизації, правильності розмежування доступу та формуванню звітної інформації.

У подальшому система може бути розширена відповідно до потреб користувачів. До можливих напрямів розвитку належать додавання механізму погодження документів, підтримка електронного підпису, експорт звітів у CSV або Excel, удосконалення пошуку, створення API для інтеграції з іншими сервісами та перехід від SQLite3 до серверної СУБД для роботи з більшими обсягами даних.

Додаток Б. Вихідні тексти програмних модулів

Лістинг 1. Програмний код файлу «documents/views.py»

```
from django.contrib import messages
from django.contrib.auth.decorators import login_required
from django.db.models import Q, Max
from django.shortcuts import get_object_or_404, redirect, render
from logs.services import log_action
from logs.models import ActionLog

from accounts.decorators import admin_required
from .forms import (
    DocumentCategoryForm,
    DocumentForm,
    DocumentVersionForm,
    DocumentAccessForm,
)
from .models import DocumentCategory, Document, DocumentVersion, DocumentAccess


@admin_required
def category_list_view(request):
    categories = DocumentCategory.objects.all().order_by("name")
    return render(request, "documents/category_list.html", {"categories": categories})


@admin_required
def category_create_view(request):
    if request.method == "POST":
        form = DocumentCategoryForm(request.POST)
        if form.is_valid():
            form.save()
            messages.success(request, "Категорію документа успішно створено.")
            return redirect("category_list")
    else:
        form = DocumentCategoryForm()

    return render(request, "documents/category_form.html", {"form": form, "mode": "create"})


@login_required
def document_list_view(request):
    search = request.GET.get("search", "").strip()
    category_id = request.GET.get("category", "").strip()
    status = request.GET.get("status", "").strip()

    documents = Document.objects.select_related("category", "author").all()

    if not (request.user.is_superuser or request.user.role == "administrator"):
        documents = documents.filter(
```

```
            Q(author=request.user) | Q(accesses__user=request.user)
        ).distinct()

    if search:
        documents = documents.filter(
            Q(title__icontains=search) | Q(description__icontains=search)
        )

    if category_id:
        documents = documents.filter(category_id=category_id)

    if status:
        documents = documents.filter(status=status)

    documents = documents.order_by("-created_at")

    context = {
        "documents": documents,
        "search": search,
        "selected_category": category_id,
        "selected_status": status,
        "categories": DocumentCategory.objects.all().order_by("name"),
        "status_choices": Document.Statuses.choices,
    }
    return render(request, "documents/document_list.html", context)


@login_required
def document_detail_view(request, document_id):
    document = get_object_or_404(
        Document.objects.select_related("category", "author"),
        id=document_id
    )

    if not (request.user.is_superuser or request.user.role == "administrator"):
        has_access = (
            document.author_id == request.user.id
            or document.accesses.filter(user=request.user).exists()
        )
        if not has_access:
            messages.error(request, "У вас немає доступу до цього документа.")
            return redirect("document_list")

    versions = document.versions.select_related("uploaded_by").all()
    accesses = document.accesses.select_related("user").all()

    can_edit = (
        request.user.is_superuser
        or request.user.role == "administrator"
        or document.author_id == request.user.id
        or document.accesses.filter(user=request.user, access_level="edit").exists()
    )
```

```
    can_manage_access = (
        request.user.is_superuser
        or request.user.role == "administrator"
        or document.author_id == request.user.id
    )

    return render(
        request,
        "documents/document_detail.html",
        {
            "document": document,
            "versions": versions,
            "accesses": accesses,
            "can_edit": can_edit,
            "can_manage_access": can_manage_access,
        }
    )


@login_required
def document_create_view(request):
    if request.method == "POST":
        form = DocumentForm(request.POST)
        if form.is_valid():
            document = form.save(commit=False)
            document.author = request.user
            document.save()

            log_action(
                user=request.user,
                action_type=ActionLog.ActionTypes.CREATE_DOCUMENT,
                description=f"Створено документ '{document.title}'.",
                document=document
            )

            messages.success(request, "Документ успішно створено.")
            return redirect("document_detail", document_id=document.id)
    else:
        form = DocumentForm()

    return render(request, "documents/document_form.html",
{"form": form, "mode": "create"})


@login_required
def document_update_view(request, document_id):
    document = get_object_or_404(Document, id=document_id)

    can_edit = (
        request.user.is_superuser
        or request.user.role == "administrator"
```

```
        or document.author_id == request.user.id
        or document.accesses.filter(user=request.user, access_level="edit").exists()
    )

    if not can_edit:
        messages.error(request, "У вас немає прав на редагування цього документа.")
        return redirect("document_detail", document_id=document.id)

    if request.method == "POST":
        form = DocumentForm(request.POST, instance=document)
        if form.is_valid():
            updated_document = form.save()

            log_action(
                user=request.user,
                action_type=ActionLog.ActionTypes.UPDATE_DOCUMENT,
                description=f"Оновлено документ '{updated_document.title}'.",
                document=updated_document
            )

            messages.success(request, "Дані документа успішно оновлено.")
            return redirect("document_detail", document_id=updated_document.id)
    else:
        form = DocumentForm(instance=document)

    return render(
        request,
        "documents/document_form.html",
        {
            "form": form,
            "mode": "edit",
            "document": document
        }
    )

@login_required
def document_add_version_view(request, document_id):
    document = get_object_or_404(Document, id=document_id)

    can_edit = (
        request.user.is_superuser
        or request.user.role == "administrator"
        or document.author_id == request.user.id
        or document.accesses.filter(user=request.user, access_level="edit").exists()
    )

    if not can_edit:
        messages.error(request, "У вас немає прав на додавання нової версії цього документа.")
        return redirect("document_detail", document_id=document.id)

    if request.method == "POST":
        form = DocumentVersionForm(request.POST, request.FILES)
```

```
        if form.is_valid():
            last_version =
document.versions.aggregate(max_version=Max("version_number"))["max_version"] or 0

            version = form.save(commit=False)
            version.document = document
            version.version_number = last_version + 1
            version.uploaded_by = request.user
            version.save()

            log_action(
                user=request.user,
                action_type=ActionLog.ActionTypes.ADD_VERSION,
                description=f"Для документа '{document.title}' завантажено версію
v{version.version_number}.",
                document=document
            )

            messages.success(request, "Нову версію документа успішно завантажено.")
            return redirect("document_detail", document_id=document.id)
    else:
        form = DocumentVersionForm()

    return render(
        request,
        "documents/document_version_form.html",
        {
            "form": form,
            "document": document,
        }
    )

@login_required
def document_manage_access_view(request, document_id):
    document = get_object_or_404(Document, id=document_id)

    can_manage = (
        request.user.is_superuser
        or request.user.role == "administrator"
        or document.author_id == request.user.id
    )

    if not can_manage:
        messages.error(request, "У вас немає прав на керування доступом до цього документа.")
        return redirect("document_detail", document_id=document.id)

    if request.method == "POST":
        form = DocumentAccessForm(request.POST, document=document)
        if form.is_valid():
            access = form.save(commit=False)
            access.document = document
```

```
            if DocumentAccess.objects.filter(document=document, user=access.user).exists():
                messages.error(request, "Для цього користувача право доступу вже надано.")
            else:
                access.save()

                log_action(
                    user=request.user,
                    action_type=ActionLog.ActionTypes.GRANT_ACCESS,
                    description=(
                        f"Користувачу {access.user.username} надано право "
                        f"'{access.get_access_level_display()}' до документа '{document.title}'."
                    ),
                    document=document
                )

                messages.success(request, "Право доступу успішно надано.")
                return redirect("document_detail", document_id=document.id)
    else:
        form = DocumentAccessForm(document=document)

    accesses = document.accesses.select_related("user").order_by("user__username")

    return render(
        request,
        "documents/document_access_manage.html",
        {
            "document": document,
            "form": form,
            "accesses": accesses,
        }
    )

@login_required
def document_remove_access_view(request, document_id, access_id):
    document = get_object_or_404(Document, id=document_id)

    can_manage = (
        request.user.is_superuser
        or request.user.role == "administrator"
        or document.author_id == request.user.id
    )

    if not can_manage:
        messages.error(request, "У вас немає прав на керування доступом до цього документа.")
        return redirect("document_detail", document_id=document.id)

    access = get_object_or_404(DocumentAccess, id=access_id, document=document)
    removed_username = access.user.username
    removed_level = access.get_access_level_display()
    access.delete()

    log_action(
```

```
        user=request.user,
        action_type=ActionLog.ActionTypes.REMOVE_ACCESS,
        description=(
            f"Для користувача {removed_username} скасовано право "
            f"'{removed_level}' до документа '{document.title}'."
        ),
        document=document
    )

    messages.success(request, "Право доступу успішно скасовано.")
    return redirect("document_manage_access", document_id=document.id)

@login_required
def document_archive_view(request, document_id):
    document = get_object_or_404(Document, id=document_id)

    can_archive = (
        request.user.is_superuser
        or request.user.role == "administrator"
        or document.author_id == request.user.id
        or document.accesses.filter(user=request.user, access_level="edit").exists()
    )

    if not can_archive:
        messages.error(request, "У вас немає прав на архівацію цього документа.")
        return redirect("document_detail", document_id=document.id)

    document.status = Document.Statuses.ARCHIVED
    document.save()

    log_action(
        user=request.user,
        action_type=ActionLog.ActionTypes.ARCHIVE_DOCUMENT,
        description=f"Документ '{document.title}' переведено в архівний стан.",
        document=document
    )

    messages.success(request, "Документ успішно архівовано.")
    return redirect("document_detail", document_id=document.id)


@login_required
def document_delete_view(request, document_id):
    document = get_object_or_404(Document, id=document_id)

    if not (request.user.is_superuser or request.user.role == "administrator"):
        messages.error(request, "Лише адміністратор може видаляти документи.")
        return redirect("document_detail", document_id=document.id)

    if request.method == "POST":
        document_title = document.title
```

```
        log_action(
            user=request.user,
            action_type=ActionLog.ActionTypes.DELETE_DOCUMENT,
            description=f"Документ '{document_title}' видалено адміністратором.",
            document=document
        )

        document.delete()
        messages.success(request, "Документ успішно видалено.")
        return redirect("document_list")

    return render(
        request,
        "documents/document_confirm_delete.html",
        {"document": document}
    )
```

Лістинг 2. Програмний код файлу «documents/models.py»

```
from django.conf import settings
from django.db import models


class DocumentCategory(models.Model):
    name = models.CharField(max_length=150, unique=True, verbose_name="Назва категорії")
    description = models.TextField(blank=True, verbose_name="Опис категорії")
    created_at = models.DateTimeField(auto_now_add=True, verbose_name="Дата створення")

    class Meta:
        verbose_name = "Категорія документа"
        verbose_name_plural = "Категорії документів"
        ordering = ["name"]

    def __str__(self):
        return self.name


class Document(models.Model):
    class Statuses(models.TextChoices):
        DRAFT = "draft", "Чернетка"
        ACTIVE = "active", "Активний"
        ARCHIVED = "archived", "Архівний"

        title = models.CharField(max_length=255, verbose_name="Назва документа")
    description = models.TextField(blank=True, verbose_name="Опис документа")
    category = models.ForeignKey(
        DocumentCategory,
        on_delete=models.PROTECT,
        related_name="documents",
        verbose_name="Категорія"
    )
    status = models.CharField(
```

```
        max_length=20,
        choices=Statuses.choices,
        default=Statuses.DRAFT,
        verbose_name="Статус"
    )
    author = models.ForeignKey(
        settings.AUTH_USER_MODEL,
        on_delete=models.CASCADE,
        related_name="authored_documents",
        verbose_name="Автор"
    )
    created_at = models.DateTimeField(auto_now_add=True, verbose_name="Дата створення")
    updated_at = models.DateTimeField(auto_now=True, verbose_name="Дата оновлення")

    class Meta:
        verbose_name = "Документ"
        verbose_name_plural = "Документи"
        ordering = ["-created_at"]

    def __str__(self):
        return self.title


class DocumentVersion(models.Model):
    document = models.ForeignKey(
        Document,
        on_delete=models.CASCADE,
        related_name="versions",
        verbose_name="Документ"
    )
    version_number = models.PositiveIntegerField(verbose_name="Номер версії")
    file = models.FileField(upload_to="documents/%Y/%m/%d/", verbose_name="Файл")
    file_name = models.CharField(max_length=255, verbose_name="Назва файлу")
    file_size = models.PositiveBigIntegerField(default=0, verbose_name="Розмір файлу")
    content_type = models.CharField(max_length=100, blank=True, verbose_name="Тип файлу")
    uploaded_by = models.ForeignKey(
        settings.AUTH_USER_MODEL,
        on_delete=models.CASCADE,
        related_name="uploaded_document_versions",
        verbose_name="Завантажив"
    )
    uploaded_at = models.DateTimeField(auto_now_add=True, verbose_name="Дата завантаження")

    class Meta:
        verbose_name = "Версія документа"
        verbose_name_plural = "Версії документів"
        ordering = ["-uploaded_at"]
        unique_together = ("document", "version_number")

    def save(self, *args, **kwargs):
        if self.file:
            self.file_name = self.file.name.split("/")[-1]
```

```
            self.file_size = self.file.size or 0
            if hasattr(self.file.file, "content_type"):
                self.content_type = self.file.file.content_type or ""
        super().save(*args, **kwargs)

    def __str__(self):
        return f"{self.document.title} - v{self.version_number}"


class DocumentAccess(models.Model):
    class AccessLevels(models.TextChoices):
        VIEW = "view", "Перегляд"
        EDIT = "edit", "Редагування"

    document = models.ForeignKey(
        Document,
        on_delete=models.CASCADE,
        related_name="accesses",
        verbose_name="Документ"
    )
    user = models.ForeignKey(
        settings.AUTH_USER_MODEL,
        on_delete=models.CASCADE,
        related_name="document_accesses",
        verbose_name="Користувач"
    )
    access_level = models.CharField(
        max_length=10,
        choices=AccessLevels.choices,
        default=AccessLevels.VIEW,
        verbose_name="Рівень доступу"
    )
    granted_at = models.DateTimeField(auto_now_add=True, verbose_name="Дата надання
доступу")

    class Meta:
        verbose_name = "Право доступу до документа"
        verbose_name_plural = "Права доступу до документів"
        unique_together = ("document", "user")

    def __str__(self):
        return f"{self.user.username} -> {self.document.title} ({self.get_access_level_display()})"
```

Додаток В. Опис програмного забезпечення

Розроблене програмне забезпечення являє собою адаптивну веб-орієнтовану систему електронного документообігу, призначену для централізованого зберігання, оброблення, пошуку, версіонування та контролю використання електронних документів. Система забезпечує організацію роботи з документами в межах єдиного веб-середовища, де користувачі можуть створювати документи, переглядати доступні матеріали, завантажувати нові версії файлів, а адміністратор – керувати користувачами, категоріями, доступом, журналом дій та аналітичною інформацією. Основна ідея програмного забезпечення полягає у заміні розрізненого файлового зберігання структурованою системою, у якій кожен документ має назву, опис, категорію, статус, автора, історію версій і визначені права доступу.

Програмне забезпечення реалізовано як вебзастосунок на основі мови програмування Python і фреймворку Django. Використання Django дало змогу застосувати готові механізми автентифікації, маршрутизації, обробки форм, роботи з базою даних через ORM, створення моделей і шаблонів. Для зберігання структурованої інформації використовується база даних SQLite3, у якій містяться відомості про користувачів, документи, категорії, версії документів, права доступу та журнал подій. Файли документів зберігаються у файловій системі сервера, а в базі даних фіксуються їхні метадані: назва, шлях, розмір, тип файлу, дата завантаження та користувач, який виконав операцію.

Інтерфейс системи реалізовано з використанням HTML5, CSS3, Bootstrap 5 та JavaScript. Завдяки цьому вебсторінки мають адаптивну структуру та можуть коректно відображатися на персональних комп’ютерах, ноутбуках, планшетах і смартфонах. Інтерфейс побудовано за модульним принципом: користувач має доступ до головної сторінки, сторінок реєстрації та входу, списку документів, форми створення документа, сторінки детального перегляду, завантаження версії та керування доступом. Адміністратор додатково отримує доступ до сторінок керування користувачами, категоріями, журналом дій, інформаційною панеллю та звітами.

У системі передбачено дві основні ролі: адміністратор і користувач. Звичайний користувач може реєструватися в системі, входити до облікового запису, переглядати доступні йому документи, створювати власні документи, завантажувати нові версії та виконувати дії відповідно до наданих прав. Адміністратор має розширені повноваження: перегляд користувачів, зміну ролей, блокування та розблокування облікових записів, керування документами, категоріями, доступом, журналом подій, дашбордом і звітністю. Такий поділ ролей забезпечує контрольовану модель доступу та дозволяє обмежити виконання критичних операцій лише користувачами з відповідними повноваженнями.

Одним із центральних компонентів програмного забезпечення є модуль керування документами. Він забезпечує створення запису про документ із зазначенням назви, опису, категорії та статусу. Документ може перебувати в одному з визначених станів: чернетка, активний або архівний. Це дозволяє відображати життєвий цикл документа та відокремлювати поточні документи від тих, що вже не використовуються активно. Для зручної роботи зі списком документів реалізовано пошук за назвою та описом, а також фільтрацію за категорією і статусом. Кожен документ має сторінку детального перегляду, де відображаються його основні атрибути, доступні версії файлів і відомості про права доступу.

Механізм версійності дозволяє зберігати кілька файлових представлень одного документа. Під час завантаження нової версії система автоматично визначає номер версії, зберігає файл у відповідному каталозі та фіксує його технічні характеристики. Такий підхід дозволяє не замінювати попередній файл безслідно, а накопичувати історію оновлень документа. Це є важливою функцією для електронного документообігу, оскільки користувачі можуть простежити, коли і ким було додано нову версію, а також контролювати актуальність документа.

Окремий функціональний блок становить механізм керування доступом. Він дозволяє надавати конкретним користувачам право перегляду або редагування документа. Право доступу може бути призначене адміністратором або автором документа, якщо він має відповідні повноваження. Перед виконанням дій із

документом система перевіряє роль користувача, авторство документа або наявність індивідуального права доступу. Це означає, що обмеження реалізовано не лише на рівні інтерфейсу, а й на серверному рівні, що підвищує надійність контролю доступу.

Для забезпечення прозорості роботи системи реалізовано журнал дій. У ньому фіксуються основні події, пов’язані з роботою користувачів: реєстрація, вхід, вихід, створення документа, редагування, додавання версії, архівація, видалення, зміна ролі, блокування користувача, надання або скасування доступу. Кожен запис журналу містить користувача, тип дії, короткий опис, дату і час виконання, а за потреби – пов’язаний документ. Наявність журналу дозволяє адміністратору контролювати активність користувачів, аналізувати виконані операції та виявляти спірні або помилкові дії.

Аналітична частина програмного забезпечення представлена інформаційною панеллю та табличними звітами. На дашборді відображаються основні кількісні показники: загальна кількість користувачів, активних користувачів, документів, категорій, активних і архівних документів, завантажених версій та записів журналу. Також передбачено графічне подання даних, зокрема розподіл документів за категоріями, динаміку створення документів і події журналу. Звітність реалізовано у вигляді таблиць, серед яких звіт по зведенню документів за категоріями та звіт активності користувачів. Це дає можливість адміністратору швидко оцінити стан документообігу та активність роботи із системою.

Структурно програмне забезпечення побудовано за модульним принципом. Модуль користувачів відповідає за реєстрацію, автентифікацію, ролі та блокування облікових записів. Модуль документів забезпечує роботу з документами, категоріями, версіями та правами доступу. Модуль журналу дій фіксує основні події системи. Модуль аналітики формує дашборд, графіки та звіти. Такий поділ спрощує супровід програмного забезпечення, дозволяє вносити зміни до окремих компонентів без порушення роботи всієї системи та створює основу для подальшого розширення.

Таким чином, розроблене програмне забезпечення забезпечує повний набір базових функцій для організації електронного документообігу: роботу з користувачами, документами, файлами, версіями, доступом, журналом дій і звітністю. Система є придатною для локального або навчального використання, не потребує складної серверної інфраструктури та може бути розгорнута у середовищі Django з базою SQLite3. У подальшому програмне забезпечення може бути розширене шляхом додавання механізму погодження документів, електронного підпису, експорту звітів, розширених ролей користувачів або переходу на серверну СУБД для роботи з більшими обсягами даних.

Додаток Г. Інструкції програмісту, оператору й користувачеві

Розроблене програмне забезпечення є веб-орієнтованою системою електронного документообігу, тому його експлуатація передбачає участь кількох категорій осіб: програміста, який відповідає за розгортання, супровід і доопрацювання системи; оператора або адміністратора, який здійснює поточне керування системою; користувача, який працює з електронними документами відповідно до наданих прав. Для забезпечення коректної роботи програмного продукту кожна з цих категорій повинна виконувати визначений набір дій та дотримуватися відповідних правил роботи із системою.

Інструкція програмісту

Програміст відповідає за встановлення, налаштування, запуск, супровід і можливе розширення програмного забезпечення. Перед початком роботи необхідно підготувати робоче середовище, яке підтримує виконання вебзастосунку на основі Python і Django. На комп’ютері або сервері повинні бути встановлені Python, менеджер пакетів pip, середовище для роботи з проєктом, а також веб-браузер для перевірки функціонування інтерфейсу. База даних у базовій версії реалізується засобами SQLite3, тому окреме встановлення серверної СУБД не є обов’язковим.

Після отримання файлів проєкту програміст повинен розмістити їх у робочому каталозі, відкрити командний рядок або термінал у папці проєкту та створити віртуальне середовище. Це дозволяє ізолювати залежності системи від інших Python-проєктів і зменшити ризик конфліктів між бібліотеками. Після активації віртуального середовища необхідно встановити залежності проєкту з файлу requirements.txt, якщо він передбачений у структурі розробки. У разі його відсутності необхідно вручну встановити Django та інші бібліотеки, які використовуються в системі.

Типова послідовність підготовки середовища може мати такий вигляд:

```
python -m venv venv
venv\Scripts\activate
pip install -r requirements.txt
```

Якщо файл залежностей не сформовано, потрібно встановити Django окремо:

```
pip install django
```

Після встановлення залежностей програміст повинен перевірити файл налаштувань проєкту settings.py. У цьому файлі необхідно звернути увагу на параметри підключення до бази даних, список встановлених застосунків, налаштування статичних файлів, медіафайлів, мови інтерфейсу, часової зони та користувацької моделі. Особливо важливо переконатися, що каталог для завантажених документів налаштовано через параметри MEDIA_ROOT і MEDIA_URL, оскільки саме там зберігатимуться файли версій документів.

Після налаштування проєкту необхідно виконати міграції бази даних. Міграції створюють фізичну структуру таблиць, необхідних для роботи системи: користувачів, документів, категорій, версій, прав доступу, журналу дій та службових таблиць Django. Для цього виконуються команди:

```
python manage.py makemigrations
python manage.py migrate
```

Після створення структури бази даних програміст повинен створити адміністративний обліковий запис, який використовуватиметься для початкового керування системою. Це виконується командою:

```
python manage.py createsuperuser
```

Після створення суперкористувача можна запустити локальний сервер розробки:

```
python manage.py runserver
```

Після запуску серверу необхідно відкрити веб-браузер і перейти за адресою, яку виведе Django, зазвичай http://127.0.0.1:8000/. На цьому етапі програміст повинен перевірити доступність головної сторінки, сторінок входу й реєстрації, адміністративної панелі, списку документів, сторінок дашборду та звітності. Якщо

виникають помилки, необхідно перевірити налаштування маршрутів, шаблонів, підключення застосунків, міграцій і прав доступу до каталогу завантажених файлів.

Під час супроводу системи програміст має контролювати цілісність бази даних, справність механізму завантаження файлів, коректність роботи форм і серверних перевірок доступу. У разі зміни структури моделей потрібно створювати та застосовувати нові міграції. Перед внесенням суттєвих змін до коду бажано створювати резервну копію бази даних db.sqlite3 і каталогу з медіафайлами, оскільки саме ці компоненти містять основні дані системи.

Програміст також відповідає за можливе розширення системи. До таких напрямів можуть належати додавання нових ролей користувачів, реалізація маршруту погодження документів, підключення електронного підпису, експорт звітів у CSV або Excel, удосконалення пошуку, інтеграція з поштовими сервісами або перехід від SQLite3 до серверної СУБД, наприклад PostgreSQL чи MySQL. Усі зміни повинні виконуватися з урахуванням модульної структури Django-проєкту, щоб не порушити роботу наявних функцій.

Інструкція оператору

Оператором системи може виступати адміністратор або відповідальна особа, яка виконує поточне керування електронним документообігом. Його основне завдання полягає у підтриманні коректної роботи системи на рівні користувачів, документів, категорій, прав доступу, журналу подій і звітної інформації. На відміну від програміста, оператор не вносить зміни до програмного коду, а працює з готовим веб-інтерфейсом.

Перед початком роботи оператор повинен відкрити веб-браузер і перейти на адресу системи. Далі необхідно виконати вхід за допомогою логіна та пароля адміністративного облікового запису. Після успішної автентифікації оператор отримує доступ до розширеного набору функцій, зокрема керування користувачами, документами, категоріями, журналом дій, інформаційною панеллю та звітами.

Першим етапом поточної роботи оператора є керування обліковими записами користувачів. Оператор повинен переглядати список зареєстрованих користувачів,

перевіряти їх ролі та стан активності. У разі потреби він може змінити роль користувача, заблокувати обліковий запис або розблокувати його. Блокування доцільно застосовувати у випадках, коли користувач більше не повинен мати доступ до системи або тимчасово втратив право працювати з документами.

Оператор також відповідає за ведення довідника категорій документів. Категорії використовуються для класифікації документів і полегшують пошук, фільтрацію та формування звітності. Під час створення категорії необхідно вказати її назву та, за потреби, опис. Не рекомендується створювати дублікати категорій із близькими назвами, оскільки це ускладнює подальше впорядкування документів і може знижувати якість аналітичних звітів.

Під час роботи з документами оператор може створювати нові документи, редагувати їх основні атрибути, архівувати документи, видаляти їх у разі потреби, переглядати детальну інформацію та завантажувати нові версії файлів. При створенні документа необхідно вказати назву, опис, категорію та статус. Назва документа повинна бути зрозумілою й інформативною, щоб інші користувачі могли швидко визначити його зміст. Статус документа потрібно встановлювати відповідно до його фактичного стану: чернетка, активний або архівний.

Окрему увагу оператор повинен приділяти керуванню доступом до документів. Якщо документ повинен бути доступний іншому користувачу, оператор має відкрити сторінку керування доступом, обрати потрібного користувача та визначити рівень доступу: перегляд або редагування. Право перегляду дозволяє користувачу ознайомитися з документом, а право редагування дає змогу виконувати розширені дії, зокрема змінювати відомості або додавати нові версії. Надання прав повинно здійснюватися лише тим користувачам, які дійсно мають потребу працювати з відповідним документом.

Оператор повинен регулярно переглядати журнал дій. У журналі фіксуються основні події системи: реєстрація, вхід, вихід, створення документів, додавання версій, архівація, видалення, зміна ролей, блокування користувачів, надання та скасування доступу. Перегляд журналу дозволяє контролювати активність

користувачів, виявляти помилкові дії, перевіряти виконання адміністративних операцій і відстежувати історію роботи з документами.

Для аналізу стану системи оператор може використовувати інформаційну панель. На ній відображаються кількісні показники щодо користувачів, документів, категорій, версій і подій журналу. Графіки дозволяють оцінити розподіл документів за категоріями, динаміку створення документів і активність користувачів. Табличні звіти використовуються для отримання узагальнених даних, наприклад кількості активних і архівних документів за категоріями або активності користувачів у системі.

Після завершення роботи оператор повинен вийти з облікового запису через відповідну кнопку в інтерфейсі. Це особливо важливо у випадку роботи на спільному комп’ютері або в навчальній аудиторії. Недотримання цієї вимоги може призвести до несанкціонованого виконання дій від імені адміністратора.

Інструкція користувачеві

Користувач системи працює з електронними документами відповідно до наданих йому прав. Його основні можливості охоплюють реєстрацію, вхід до системи, перегляд доступних документів, створення власних документів, завантаження нових версій, перегляд детальної інформації та виконання операцій, дозволених адміністратором або автором документа.

Для початку роботи користувач повинен відкрити веб-браузер і перейти на головну сторінку системи. Якщо облікового запису ще немає, необхідно обрати сторінку реєстрації. У формі реєстрації потрібно ввести ім’я користувача, електронну адресу, пароль і підтвердження пароля. Після успішного створення облікового запису система перенаправляє користувача на сторінку входу. За замовчуванням новому користувачу надається базова роль, яка не передбачає адміністративних повноважень.

Для входу до системи користувач вводить логін і пароль у формі автентифікації. Якщо дані введено правильно, система відкриває доступ до функцій, дозволених для цього облікового запису. Якщо введені неправильні дані або обліковий запис заблоковано, користувач не зможе перейти до захищених

сторінок. У такому випадку потрібно перевірити правильність введення логіна і пароля або звернутися до адміністратора системи.

Після входу користувач може перейти до списку документів. У цьому розділі відображаються документи, які доступні користувачу відповідно до його ролі, авторства або індивідуально наданих прав. Для зручності передбачено пошук за назвою або описом, а також фільтрацію за категорією та статусом. Це дозволяє швидше знайти потрібний документ навіть за наявності значної кількості записів.

Для створення нового документа користувач повинен натиснути кнопку створення документа та заповнити форму. У ній потрібно вказати назву, опис, категорію та статус документа. Після збереження документ буде додано до системи, а користувач стане його автором. Автор документа може виконувати розширені дії з власним документом відповідно до реалізованої логіки системи.

Під час перегляду документа користувач бачить основні відомості про нього: назву, опис, категорію, статус, автора, дату створення та доступні версії. Якщо користувач має право редагування, він може змінювати атрибути документа або завантажувати нову версію файлу. Для завантаження версії необхідно відкрити відповідну форму, обрати файл на комп’ютері та підтвердити завантаження. Після цього система збереже файл і зафіксує його метадані.

Якщо користувачу надано лише право перегляду, він може ознайомитися з документом, але не повинен мати можливості змінювати його атрибути або додавати нові версії. Такий поділ прав дозволяє захистити документи від випадкових або несанкціонованих змін. Якщо користувачу необхідно отримати додаткові права, він повинен звернутися до адміністратора або автора документа.

Користувач повинен уважно працювати із завантаженням файлів. Перед додаванням нової версії необхідно переконатися, що файл відповідає змісту документа, має коректну назву та не містить зайвих або помилкових даних. Не рекомендується завантажувати файли, які не стосуються відповідного документа, оскільки це ускладнює подальше використання системи та аналіз історії версій.

Після завершення роботи користувач повинен вийти з облікового запису. Це забезпечує захист персонального доступу та зменшує ризик виконання дій іншою

особою від імені поточного користувача. Також користувач повинен зберігати пароль у таємниці, не передавати його іншим особам і не використовувати надто прості комбінації для входу.

Таким чином, інструкції для програміста, оператора й користувача визначають порядок технічного супроводу, адміністрування та повсякденного використання системи. Дотримання наведених правил забезпечує стабільну роботу програмного забезпечення, коректне ведення електронних документів, контроль доступу та збереження цілісності інформації.

Додаток Д. Методика випробувань ПЗ

Методика випробувань програмного забезпечення визначає порядок перевірки працездатності адаптивної веб-орієнтованої системи електронного документообігу, її відповідності функціональним і нефункціональним вимогам, а також коректності виконання основних користувацьких та адміністративних сценаріїв. Випробування проводяться після завершення основних етапів реалізації системи та налаштування бази даних, оскільки саме на цьому етапі можливо перевірити взаємодію між інтерфейсом, серверною логікою Django, моделями даних, файловим сховищем і механізмами контролю доступу.

Метою випробувань є підтвердження того, що розроблене програмне забезпечення забезпечує реєстрацію та автентифікацію користувачів, рольове розмежування доступу, керування документами, завантаження версій файлів, надання прав доступу, ведення журналу дій, формування інформаційної панелі та табличних звітів. Окремо перевіряється стабільність роботи веб-інтерфейсу, правильність обробки помилкових дій користувача, коректність збереження даних у базі SQLite3 та відповідність поведінки системи очікуваним результатам.

Випробування проводяться в локальному середовищі розробки. Для цього використовується персональний комп’ютер або ноутбук із встановленим інтерпретатором Python, фреймворком Django, базою даних SQLite3 та сучасним веб-браузером. Запуск системи здійснюється через сервер розробки Django командою python manage.py runserver, після чого доступ до вебзастосунку виконується через браузер за локальною адресою. Перед початком тестування необхідно застосувати міграції бази даних, створити адміністративний обліковий запис, додати демонстраційні категорії документів і підготувати кілька тестових користувачів із різними ролями.

Об’єктами випробувань є основні модулі системи: модуль користувачів, модуль документів, модуль версій, модуль прав доступу, модуль журналу дій, модуль аналітики та звітності. Модуль користувачів перевіряється з погляду реєстрації, входу, виходу, блокування облікового запису та зміни ролі. Модуль документів перевіряється через створення, редагування, перегляд, архівацію,

видалення, пошук і фільтрацію документів. Модуль версій оцінюється шляхом завантаження файлів і перевірки автоматичного збереження їх метаданих. Модуль доступу тестується через надання і скасування прав перегляду або редагування для окремих користувачів. Модуль журналу перевіряється шляхом зіставлення виконаних дій із записами, що з'являються в журналі. Модуль аналітики перевіряється через правильність підрахунку показників і формування звітів.

Перед початком випробувань необхідно підготувати тестові дані. До них належать обліковий запис адміністратора, один або кілька звичайних користувачів, категорії документів, тестові документи з різними статусами, файли для завантаження як версії документів, а також декілька записів доступу між користувачами й документами. Для перевірки різних сценаріїв доцільно використовувати документи зі статусами «Чернетка», «Активний» і «Архівний», а також файли різних типів, наприклад .docx, .pdf, .txt.

Випробування системи здійснюється за методом функціонального тестування. Це означає, що перевіряється не внутрішня реалізація окремих фрагментів коду, а фактична поведінка системи під час виконання дій користувачем або адміністратором. Для кожного тестового сценарію задаються початкові умови, виконується певна дія в інтерфейсі, після чого фіксується фактичний результат. Якщо отриманий результат відповідає очікуваному, тест вважається успішним. Якщо система працює неправильно, результат фіксується як помилка, що потребує подальшого аналізу та виправлення.

Для зручності проведення випробувань основні тестові сценарії доцільно подати у вигляді таблиці.

Таблиця Д.1– Методика функціонального випробування системи

№	Сценарій випробування	Дії під час перевірки	Очікуваний результат
1	2	3	4
1	Відкриття головної сторінки	Перейти на головну сторінку системи через браузер	Сторінка відкривається без помилок, відображаються назва системи, опис і навігаційні елементи

Продовження таблиці Д.1.

1	2	3	4
2	Реєстрація користувача	Заповнити форму реєстрації та підтвердити створення облікового запису	Користувач створюється в базі даних, отримує роль user, система перенаправляє на сторінку входу
3	Автентифікація користувача	Ввести правильний логін і пароль у формі входу	Користувач успішно входить до системи та отримує доступ до дозволених сторінок
4	Перевірка неправильного входу	Ввести неправильний пароль	Система не виконує вхід і відображає повідомлення про помилку
5	Блокування користувача	Адміністратор блокує обліковий запис користувача	Заблокований користувач не може увійти до системи
6	Зміна ролі користувача	Адміністратор змінює роль користувача	Роль оновлюється, доступ користувача змінюється відповідно до нових прав
7	Створення документа	Авторизований користувач заповнює форму створення документа	Документ створюється, прив’язується до автора та відображається у списку документів
8	Пошук документа	Ввести пошуковий запит за назвою або описом	У списку відображаються документи, що відповідають запиту
9	Фільтрація документів	Обрати категорію або статус документа	Система відображає лише документи, що відповідають вибраному фільтру
10	Перегляд документа	Відкрити сторінку детального перегляду документа	Відображаються назва, опис, категорія, статус, автор, версії та доступні дії
11	Редагування документа	Користувач із правами редагування змінює атрибути документа	Дані документа оновлюються та зберігаються в базі

Продовження таблиці Д.1.

1	2	3	4
12	Завантаження нової версії	Обрати файл і завантажити його як нову версію документа	Система створює новий запис версії, зберігає файл і його метадані
13	Перевірка заборони редагування	Користувач без прав намагається редагувати документ	Система забороняє дію та відображає повідомлення про відсутність прав
14	Надання доступу	Адміністратор або автор надає користувачу доступ до документа	У системі створюється запис доступу з визначеним рівнем прав
15	Скасування доступу	Видалити раніше надане право доступу	Користувач втрачає можливість виконувати відповідні дії з документом
16	Архівація документа	Перевести документ в архівний статус	Документ отримує статус «Архівний» і не відображається серед активних без спеціального фільтра
17	Видалення документа	Адміністратор видаляє документ після підтвердження	Документ видаляється або стає недоступним відповідно до реалізованої логіки
18	Перегляд журналу дій	Відкрити сторінку журналу дій	Відображаються записи про виконані операції користувачів
19	Фільтрація журналу	Вибрати тип події або користувача	Система відображає лише відповідні записи журналу
20	Перегляд дашборду	Відкрити інформаційну панель адміністратора	Відображаються кількісні показники та графіки
21	Формування звіту за категоріями	Відкрити звіт документів за категоріями	Таблиця містить кількість документів, активних і архівних записів за кожною категорією
22	Формування звіту активності	Відкрити звіт активності користувачів	Таблиця містить роль, кількість дій, кількість завантажених версій і дату останньої активності

Продовження таблиці Д.1.

1	2	3	4
23	Завершення сеансу	Натиснути кнопку виходу з системи	Сесія користувача завершується, захищені сторінки стають недоступними без повторного входу
24	Адаптивність інтерфейсу	Відкрити сторінки на різних розмірах екрана	Елементи інтерфейсу коректно перебудовуються, горизонтальна прокрутка не заважає роботі

Під час випробувань також необхідно перевірити обробку некоректних вхідних даних. До таких ситуацій належать спроба реєстрації з незаповненими обов’язковими полями, введення різних значень пароля та підтвердження пароля, створення документа без назви, завантаження файлу без вибору документа, спроба надати доступ користувачу, який уже має відповідне право, а також спроба виконання адміністративних дій звичайним користувачем. У кожному з таких випадків система повинна не завершувати роботу аварійно, а виводити зрозуміле повідомлення або повертати користувача до форми з поясненням помилки.

Окремою частиною методики є перевірка безпеки. Вона передбачає контроль доступу до захищених маршрутів без входу в систему, спробу відкриття адміністративних сторінок звичайним користувачем, перевірку блокування облікового запису та перевірку прав на редагування документа. Очікуваним результатом є недопущення несанкціонованого доступу, коректне перенаправлення користувача або виведення повідомлення про відсутність прав. Особливо важливо переконатися, що перевірки виконуються на серверному рівні, оскільки приховування кнопок в інтерфейсі не є достатнім механізмом захисту.

Перевірка продуктивності виконується шляхом оцінювання часу відкриття основних сторінок, виконання пошуку, фільтрації, створення документа, завантаження нової версії та формування звітів. У локальному середовищі основні сторінки повинні відкриватися без помітних затримок, а формування дашборду та звітів має виконуватися протягом прийнятного часу. Для навчального або демонстраційного проєкту достатнім є підтвердження того, що система стабільно

працює з типовим обсягом даних і не створює надмірного навантаження на комп’ютер.

Результати випробувань фіксуються у вигляді таблиці або протоколу, у якому зазначаються номер тесту, назва сценарію, вхідні дані, очікуваний результат, фактичний результат і висновок про проходження тесту. Якщо фактичний результат збігається з очікуваним, тест вважається пройденим. Якщо виявлено помилку, необхідно описати її прояв, умови виникнення та передати на виправлення. Після усунення помилки відповідний сценарій перевіряється повторно.

Критеріями успішного проходження випробувань є коректне виконання основних функціональних сценаріїв, відсутність критичних помилок, правильне збереження даних у базі, стабільна робота файлового сховища, коректне розмежування доступу, наявність записів у журналі дій і правильне формування аналітичних показників. Система вважається готовою до використання, якщо всі базові сценарії виконуються відповідно до вимог, а виявлені недоліки не перешкоджають виконанню основних функцій електронного документообігу.