МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних інформаційних систем
_____ Євген СІДЕНКО
«____»_____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

ІНФОРМАЦІЙНА СИСТЕМА ЕЛЕКТРОННОГО ДОКУМЕНТООБІГУ

Спеціальність 122 Комп'ютерні науки

Освітня програма «Комп'ютерні науки»

Здобувач ____Владислав КАНІРОВСЬКИЙ
«___»_____ 2026 р.

Керівник д-р техн. наук, професор _____ Олександр ГОЖИЙ
«___»_____ 2026 р.

Миколаїв – 2026

Чорноморський національний університет імені Петра Могили

(повне найменування закладу вищої освіти)

Факультет	Комп’ютерних наук
Кафедра	Інтелектуальних інформаційних систем
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступень	Бакалавр
Спеціальність	122 Комп’ютерні науки
Освітня програма	Комп’ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем

_____ Євген СІДЕНКО

«____» _____ 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

Каніровського Владислава Олександровича

((прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Інформаційна система електронного документообігу».

Керівник роботи: Гожий Олександр Петрович, д-р техн. наук, професор, професор кафедри ІІС.

Затверджена наказом ЧНУ ім. Петра Могили від «25» грудня 2025 р. № 353.

2. Строк представлення кваліфікаційної роботи «___» _____ 2025 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні: веборієнтована інформаційна система електронного документообігу для створення, зберігання, пошуку, редагування, погодження та контролю електронних

документів; опис процесів документообігу, вимоги до системи, ролі користувачів, типи документів, статуси обробки документів, початкові дані для бази даних MySQL.

4. Перелік питань, що підлягають розробці: аналіз сучасного стану електронного документообігу та проблем паперової обробки документів; огляд існуючих систем електронного документообігу та порівняння їх функціональних можливостей; формування функціональних і нефункціональних вимог до веборієнтованої інформаційної системи; проєктування архітектури системи, бази даних, ролей користувачів і сценаріїв взаємодії; розробка програмного прототипу системи на основі PHP та MySQL; тестування основних функцій системи та аналіз отриманих результатів.

5. Перелік графічних матеріалів: презентація.

Керівник роботи _____ Олександр ГОЖИЙ

(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Владислав КАНІРОВСЬКИЙ

(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата видачі завдання «25» грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН
кваліфікаційної роботи

Тема: Інформаційна система електронного документообігу

№	Найменування роботи	Початок	Закінчення	Примітки
1	Отримання завдання на виконання КР	21.12.2025	24.12.2025	
2	Аналіз предметної області та постановка задачі	25.12.2025	30.01.2026	
3	Огляд літературних джерел за темою кваліфікаційної роботи, зокрема огляд публікацій та аналогічних систем щодо організації електронного документообігу	31.01.2026	01.03.2026	
4	Огляд сучасних технологій, архітектурних підходів і засобів розробки веборієнтованої інформаційної системи електронного документообігу	02.03.2026	01.04.2026	
5	Реалізація інформаційної системи електронного документообігу з аналізом отриманих результатів	02.04.2026	24.05.2026	
6	Перший попередній захист КР на засіданні комісії кафедри	25.05.2026	25.05.2026	
7	Корегування роботи за результатами попереднього захисту	26.05.2026	04.06.2026	
8	Другий попередній захист КР на засіданні комісії кафедри	05.06.2026	05.06.2026	
9	Доробка та остаточне оформлення КР	06.06.2026	14.06.2026	
10	Подання КР, її електронної копії та інших документів (відгуку, рецензії) до захисту	15.06.2026	19.06.2026	

Керівник роботи _____ Олександр ГОЖИЙ
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Владислав КАНІРОВСЬКИЙ
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата складання календарного плану
«29» січня 2026 р.

АНОТАЦІЯ

до кваліфікаційної роботи

здобувача (ки) групи 401з ЧНУ ім. Петра Могили

Каніровського Владислава Олександровича

на тему: **«ІНФОРМАЦІЙНА СИСТЕМА ЕЛЕКТРОННОГО ДОКУМЕНТООБІГУ»**

Кваліфікаційна робота присвячена розробці веборієнтованої інформаційної системи електронного документообігу, яка дає змогу створювати документи, зберігати їх, здійснювати пошук, редагування, погодження та контроль статусів. Актуальність теми зумовлена тим, що в багатьох організаціях документи досі обробляються вручну або зберігаються у розрізнених файлах, через що ускладнюється пошук, контроль виконання та відстеження відповідальних осіб.

Об’єкт роботи — процес організації електронного документообігу в установах, підприємствах та навчальних підрозділах за допомогою веборієнтованої інформаційної системи.

Предмет роботи — методи, моделі та програмні засоби для створення, збереження, пошуку, редагування, погодження та контролю електронних документів.

Мета роботи — розробка інформаційної системи електронного документообігу з функціями авторизації користувачів, створення й редагування документів, пошуку, погодження, контролю статусів та фіксації дій у журналі.

Дана робота складається з чотирьох розділів. У першому розділі проведено аналіз предметної області, розглянуто поняття електронного документообігу, нормативно-правові та організаційні засади, а також виконано огляд сучасних систем-аналогів. Другий розділ присвячений постановці задачі, формуванню вимог, проєктуванню архітектури системи, бази даних, ролей користувачів і сценаріїв взаємодії. У третьому розділі наведено опис розробки програмного прототипу вебсистеми електронного документообігу. Загальний обсяг роботи – 92

2026 р. Каніровський Владислав

сторінки. Кваліфікаційна робота містить 5 додатків, 30 рисунків, 19 таблиць і 30 джерел посилання.

Ключові слова: електронний документообіг, інформаційна система, вебзастосунок, PHP, MySQL, база даних, авторизація, погодження документів, статус документа, журнал дій.

ABSTRACT

to the qualification work by the student of the group 401з of Petro Mohyla Black Sea National University

Kanirovskyi Vladislav

“ INFORMATION SYSTEM OF ELECTRONIC DOCUMENT MANAGEMENT”

The relevance of this qualification work lies in the need to improve the efficiency of document workflow organization in institutions, enterprises, and educational departments by automating the processes of creating, storing, searching, editing, approving, and controlling electronic documents. The use of a web-oriented information system makes it possible to reduce document processing time, minimize the risk of data loss, ensure transparency of document movement, and improve the convenience of users’ work.

The object of the work is the process of organizing electronic document workflow in information systems.

The subject of the work is the methods, models, and software tools for developing a web-oriented electronic document management information system.

As a result of the work, the subject area of electronic document workflow was studied, modern analogous systems were analyzed, functional and non-functional requirements for the system were defined, the architecture of the web application and the database structure were designed, and a software prototype of the electronic document management information system was developed using PHP, MySQL, HTML, CSS, JavaScript, and Bootstrap.

The qualification work consists of four sections. The first section analyzes the subject area, considers the concept of electronic document workflow, its organizational and legal aspects, and reviews modern analogous systems. The second section is devoted to the formulation of the task, system requirements, architecture design, database design, user roles, and interaction scenarios.

Keywords: electronic document management, information system, web application, PHP, MySQL, database, authorization, document approval, document status, activity log

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних;

БКР – бакалаврська кваліфікаційна робота;

ЕД – електронний документ;

ЕДО – електронний документообіг;

ІС – інформаційна система;

КЕП – кваліфікований електронний підпис;

ПЗ – програмне забезпечення;

СКБД – система керування базами даних;

Bootstrap – фреймворк для створення адаптивного вебінтерфейсу;

CSS – Cascading Style Sheets;

HTML – Hyper Text Markup Language;

JS – JavaScript;

MVC – Model-View-Controller;

MySQL – реляційна система керування базами даних;

OSPanel – локальне серверне середовище;

PHP – Hypertext Preprocessor;

phpMyAdmin – вебінструмент адміністрування MySQL;

SQL – Structured Query Language;

UI – User Interface;

UML – Unified Modeling Language.

ВСТУП

Будь-яка організація — підприємство, установа чи навчальний заклад — щодня має справу з потоком документів. Заяви й накази, службові записки й договори, акти, звіти, внутрішні повідомлення — усе це потребує певного порядку. Якщо такого порядку немає, робота ускладнюється: важко знайти потрібний документ, незрозуміло, хто й коли вніс зміни, де він застряг і чи взагалі виконано необхідне рішення.

Паперовий документообіг, звичний десятиліттями, поступово стає гальмом. Не тому, що паперові документи раптово втратили силу, а тому, що темп роботи змінився: запити надходять швидше, команди розподілені між офісами або працюють дистанційно, а контролювати виконання доручень через фізичні папки дедалі складніше.

Електронний документообіг давно перестав бути чимось передовим — для багатьох організацій це просто умова нормальної роботи. Система, яка дозволяє створювати документи, надсилати їх на погодження, відстежувати статуси і зберігати історію дій, скорочує час на рутину і робить управлінські процеси прозорішими.

Окрему перевагу мають веборієнтовані рішення: не потрібно встановлювати програми на кожному комп'ютері — достатньо браузера і доступу до сервера. Для невеликих організацій і навчальних проєктів такий підхід є зручним і достатньо гнучким для подальшого розвитку.

Метою кваліфікаційної роботи є розробка інформаційної системи електронного документообігу, яка забезпечує повний цикл роботи з електронними документами: від створення та зберігання до пошуку, редагування, погодження та контролю у вебсередовищі.

Об'єктом дослідження є процес організації електронного документообігу в інформаційних системах. Предметом дослідження — методи, моделі та програмні засоби розробки веборієнтованої системи на основі PHP і MySQL.

Для досягнення мети розв'язувався такий комплекс завдань: аналіз предметної області; вивчення нормативних та організаційних вимог; порівняння наявних систем-аналогів; формулювання функціональних і нефункціональних вимог; проєктування архітектури і бази даних; реалізація основних модулів вебзастосунку; опис результатів і перспектив подальшого розвитку.

У роботі застосовано методи системного аналізу, проєктування баз даних, UML-моделювання, аналізу аналогів і функціонального тестування. Практична реалізація спирається на стек PHP, MySQL, HTML, CSS, JavaScript і Bootstrap.

Практична цінність роботи полягає в тому, що розроблена система може слугувати основою внутрішнього документообігу в невеликій організації або навчальному підрозділі. Вона побудована на зрозумілій архітектурі, підтримує розподіл ролей і може бути розширена модулями електронного підпису, генерації PDF, повідомлень або резервного копіювання.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ТЕОРЕТИЧНІ ОСНОВИ ЕЛЕКТРОННОГО ДОКУМЕНТООБІГУ

1.1 Поняття, сутність і роль електронного документообігу

Документообіг — це весь шлях, який проходить документ в організації: від моменту створення до архіву. Спершу він реєструється, передається потрібним людям, розглядається, виконується і врешті зберігається. Десятиліттями цей шлях пролягав через стоси паперів, журнали реєстрації, живі підписи й полиці з папками. Ця модель добре служила свого часу — але з ростом обсягів інформації стала помітно гальмувати.

Електронний документообіг — це та сама логіка, але реалізована інакше. Документ створюється, погоджується, передається і зберігається в цифровому вигляді. Але головне тут — не те, що він електронний. Головне — що навколо нього є керований процес: статус, автор, дата, відповідальна особа, маршрут погодження і повна історія того, що з ним відбувалося.

Тому електронний документ у такій системі — це не просто файл у папці. Це повноцінний об'єкт зі змістом, реквізитами, поточним станом і зв'язками з іншими документами. Наприклад, договір може мати прикріплений файл, опис, дату, статус «на погодженні» або «затверджено» і перелік усіх, хто брав участь у його обробці.

Щоб розуміти, як це працює, варто знати кілька ключових понять. **Маршрут** — це послідовність етапів, через які проходить документ. **Статус** — де саме він перебуває зараз. **Роль** — що конкретний користувач може з ним робити. **Журнал дій** — незмінний запис про те, хто, коли і що зробив.

На практиці найвідчутніша перевага електронного документообігу — це швидкість. Паперовий документ треба фізично донести: зайти в кабінет, покласти на стіл, дочекатися підпису. Електронний стає доступним майже одразу після створення. Це особливо важливо, коли команда розподілена між різними корпусами, містами або взагалі працює дистанційно.

Окрема цінність — пошук. Знайти потрібний аркуш у паперовому архіві означає переглядати папки й журнали вручну. У цифровій системі це займає кілька секунд: за назвою, автором, датою, статусом або навіть фрагментом тексту. І це не просто зручність — це захист від ситуації, коли важливий документ просто губиться серед решти.

Є ще один ефект, який часто недооцінюють, — прозорість. Керівник у будь-який момент бачить: які документи затверджені, які ще розглядаються, а які повернуті на доопрацювання. Це допомагає реально оцінити завантаженість команди, знайти вузькі місця і розподілити роботу рівномірніше.

Разом із тим є і зворотний бік. Там, де в паперовій системі головний ризик — фізична втрата документа, в електронній виходять на перший план питання безпеки: захист облікових записів, обмеження прав доступу, резервні копії, контроль над діями користувачів. Тому зручність і захищеність тут не протиставляються — вони обидві обов'язкові.

У цій роботі електронний документообіг розглядається як керований шлях документа від створення до кінцевого рішення — «затверджено» або «відхилено». Такий підхід підходить для невеликої організації і водночас дозволяє показати ключові принципи побудови повноцінної системи без зайвого ускладнення.

Зрештою, система електронного документообігу — це не спосіб позбутися паперу. Це інший спосіб організувати роботу з інформацією, де кожен документ стає частиною чіткого і керованого процесу. Саме ця ідея лежить в основі всього подальшого проєктування.

Основні поняття предметної області наведено в таблиці 1.1.

Таблиця 1.1 – Основні поняття предметної області

Поняття	**Зміст**	**Значення для системи**
Електронний документ	Документ, інформація в якому подана в електронній формі та має службові атрибути.	Основний об’єкт зберігання, пошуку і погодження.

Кінець таблиці 1.1

Поняття	**Зміст**	**Значення для системи**
Маршрут документа	Послідовність етапів, які проходить документ.	Дозволяє організувати процес погодження.
Статус документа	Поточний стан документа в системі.	Дає можливість контролювати обробку.
Роль користувача	Набір дозволених операцій.	Використовується для розмежування доступу.
Журнал дій	Записи про операції користувачів.	Потрібний для контролю, аудиту та аналізу.

Порівняння паперового та електронного документообігу наведено в таблиці 1.2.

Таблиця 1.2 – Порівняння паперового та електронного документообігу

Критерій	**Паперовий документообіг**	**Електронний документообіг**
Швидкість	Залежить від фізичної передачі документів.	Документ доступний одразу після створення або відправлення.
Пошук	Потребує ручного перегляду архівів.	Виконується за назвою, автором, датою, статусом або текстом.
Контроль	Залежить від відповідальної особи.	Здійснюється через статуси та журнал дій.
Зберігання	Потребує фізичного місця.	Організовується у базі даних і файловому сховищі.
Ризики	Втрата або пошкодження паперу.	Потребує технічного захисту та резервування.

Загальна логіка електронного документообігу показано на рисунку 1.1.

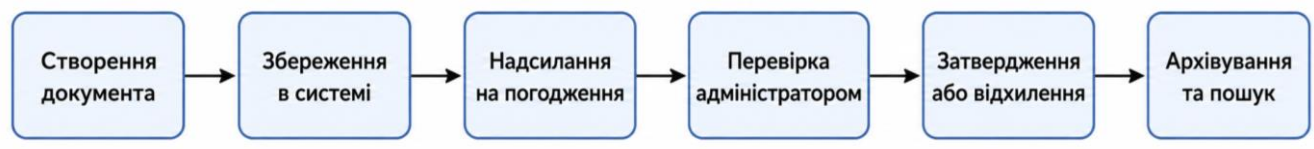

Рисунок 1.1 – Загальна логіка електронного документообігу

У практичній площині важливо, що електронний документообіг змінює не лише носій документа, а й саму культуру роботи з інформацією. Якщо у паперовій моделі значна частина часу витрачається на передачу документа між кабінетами, уточнення його поточного стану і ручний пошук у папках, то в електронній системі ці дії переходять у керований цифровий процес. Користувач бачить не абстрактний файл, а документ із конкретним статусом, автором, датою створення та історією обробки. Саме така зміна підходу робить систему документообігу корисною навіть тоді, коли її перша версія має відносно простий функціонал.

Для невеликої організації або навчального підрозділу особливо важливо, щоб система залишалася зрозумілою. Надмірно складний інтерфейс може створити більше проблем, ніж вирішити, тому при розробці прототипу було зроблено акцент на базових діях: увійти до системи, створити документ, переглянути його, передати на погодження і побачити результат. Такий підхід не перевантажує користувача зайвими налаштуваннями, але водночас демонструє головну ідею електронного документообігу — контрольований рух документа від створення до прийняття рішення.

1.2 Нормативно-правові та організаційні засади електронного документообігу

Питання електронного документообігу не зводиться лише до технічних рішень. Документ у роботі будь-якої установи несе юридичне, організаційне та управлінське навантаження. Тому проєктуючи інформаційну систему, не можна обмежитися зручним інтерфейсом — необхідно враховувати вимоги до ідентифікації користувачів, збереження даних, підтвердження авторства та забезпечення цілісності інформації.

В Україні базові засади роботи з електронними документами закріплено у Законі «Про електронні документи та електронний документообіг» [1]. Він визначає загальні принципи створення, передачі, зберігання і використання електронних документів. З погляду проєктування системи це означає одне:

електронний документ повинен мати не лише зміст, а й службові реквізити, що дозволяють ідентифікувати його в інформаційному середовищі.

Окремого значення набуває сфера електронної ідентифікації. Закон «Про електронну ідентифікацію та електронні довірчі послуги» регулює застосування електронних підписів і механізмів довірчої взаємодії [2]. На поточному етапі роботи інтеграція кваліфікованого електронного підпису не входить до переліку обов'язкового функціоналу, однак архітектура системи свідомо спроєктована так, щоб цей модуль можна було додати в майбутньому без суттєвих переробок.

З організаційного боку важливо від самого початку визначити ролі користувачів. Якщо всі мають однакові права, система швидко стає некерованою. Мінімально розумний поділ — звичайний користувач і адміністратор. Перший створює документи, працює з власними записами і надсилає їх на погодження. Другий переглядає документи, приймає рішення щодо їх затвердження або відхилення, контролює журнал дій.

Важлива організаційна вимога — фіксація дій. Якщо документ змінювався, переходив на погодження або отримував затвердження, система повинна зберігати відповідний запис. Це не формальність — у разі спірної ситуації журнал дозволяє встановити, хто і коли виконав конкретну операцію. Без такого журналу будь-яка суперечка стає нерозв'язуваною.

Ще один організаційний аспект — визначення життєвого циклу документа. У розроблюваній системі пропонується чотири базові статуси: «Створено», «На погодженні», «Затверджено» і «Відхилено». Цієї схеми цілком достатньо для навчального прототипу, і водночас вона відображає реальну логіку руху документа в організації.

З погляду безпеки діє принцип мінімально необхідних прав: кожен користувач повинен бачити лише ті функції, які йому справді потрібні для роботи. Звичайний користувач не потребує доступу до адміністративного журналу чи погодження чужих документів. Таке обмеження захищає від випадкового або навмисного втручання у роботу системи.

Для зберігання документів не менш важливі метадані. Назва, автор, дата, статус, опис і шлях до файлу — саме ці атрибути перетворюють набір файлів на керовану інформаційну систему з можливостями пошуку, фільтрації та аудиту.

Нормативні та організаційні вимоги безпосередньо визначають архітектуру програмного забезпечення. Авторизація, ролі, статуси, журнал дій, централізоване зберігання і гнучкість для майбутнього розширення — ці елементи є обов'язковими. Без них система залишалася б лише файловим каталогом

Організаційні вимоги до електронного документообігу наведено в таблиці 1.3.

Таблиця 1.3 – Організаційні вимоги до електронного документообігу

Вимога	**Пояснення**	**Відображення в системі**
Ідентифікація користувача	Кожна дія має виконуватися від імені конкретного користувача.	Авторизація за email і паролем, використання PHP-сесій.
Контроль доступу	Користувач бачить лише дозволені функції.	Ролі admin і user, перевірка прав на сервері.
Фіксація дій	Важливі операції повинні зберігатися для контролю.	Таблиця logs із записом дії та часу.
Життєвий цикл документа	Документ проходить визначені етапи обробки.	Статуси: створено, на погодженні, затверджено, відхилено.
Можливість розширення	Система повинна дозволяти додавати нові модулі.	Окремі таблиці статусів, погоджень і журналу дій.

Таким чином, організаційні вимоги показують, що система електронного документообігу повинна будуватися не лише навколо збереження файлів, а й навколо керованого процесу роботи з ними. Для подальшого проєктування це означає необхідність передбачити авторизацію, статуси, ролі користувачів і фіксацію ключових дій у журналі.

У контексті кваліфікаційної роботи ці вимоги розглядаються не як формальність, а як практичні орієнтири для побудови прототипу. Наприклад, сама наявність авторизації означає, що кожна дія в системі може бути пов’язана з конкретною особою. Наявність статусів дозволяє не губити документ у загальному списку. Журнал дій потрібний для того, щоб після певної операції можна було зрозуміти, хто саме її виконав і коли це сталося. Завдяки цьому система поступово набуває ознак не просто вебсторінки з формами, а повноцінного інструмента організації роботи.

Окремо слід зазначити, що у навчальному прототипі не потрібно відразу реалізовувати весь комплекс складних юридично значущих механізмів. Наприклад, інтеграція кваліфікованого електронного підпису є важливою перспективою, але на першому етапі доцільніше правильно спроєктувати структуру документів, користувачів, статусів і журналу. Якщо ці базові елементи закладені коректно, надалі до системи можна додавати нові модулі без повного переписування програмного коду.

1.3 Аналіз сучасних систем електронного документообігу

Перш ніж формулювати вимоги до власної системи, варто подивитися, що вже існує на ринку. Серед рішень, поширених в Україні, виділяються M.E.Doc, «Вчасно», Document.Online, а серед міжнародних платформ — DocuWare і Microsoft SharePoint. Кожне з них вирішує свої задачі, але кожне й має обмеження, важливі для розуміння контексту власної розробки.

M.E.Doc — добре знаний комплекс для електронної звітності та обміну юридично значущими документами. Його сила — орієнтація на роботу з контролюючими органами, підтримка електронного підпису, глибока інтеграція з бухгалтерськими процесами. Проте для невеликого внутрішнього документообігу цей продукт видається занадто масивним і спеціалізованим.

Сервіс «Вчасно» — веборієнтована платформа для роботи з документами онлайн. Зручна, не потребує встановлення, підтримує створення, погодження,

підписання, обмін і зберігання. Очевидний мінус — комерційна модель: власні тарифи, залежність від зовнішньої інфраструктури і відсутність контролю над даними.

DocuWare — міжнародна система управління документами та автоматизації робочих процесів. Орієнтована на організації, яким потрібні повноцінні бізнес-процеси навколо документів, а не лише їх зберігання. Широкий функціонал, але разом з ним — значні ресурси на впровадження та налаштування під конкретну організацію.

Microsoft SharePoint — платформа для спільної роботи, керування контентом і корпоративними порталами з глибокою інтеграцією в екосистему Microsoft 365. Для організацій, що вже використовують Office, це природний вибір. Для самостійного навчального прототипу — занадто масштабне рішення зі складним розгортанням.

Аналіз підтвердив очевидне: промислові системи мають широкий функціонал, але не завжди підходять для навчального або малого організаційного контексту. Саме тому розумно створити власний вебзастосунок із прозорою, зрозумілою моделлю: користувач створює документ, надсилає на погодження, адміністратор приймає рішення, система фіксує все в журналі.

Перевага власної системи — не в тому, щоб конкурувати з комерційними продуктами, а в тому, щоб мати контрольований прототип, який демонструє основні принципи ЕДО і може розвиватися поступово. Такий підхід особливо виправданий у рамках бакалаврської роботи: кожен рядок коду зрозумілий розробнику, а демонстрація системи не потребує складної інфраструктури.

Порівняння сучасних систем електронного документообігу наведено в таблиці 1.4.

Таблиця 1.4 – Порівняння сучасних систем електронного документообігу

Система	**Основні можливості**	**Переваги**	**Обмеження**
M.E.Doc	Звітність, ЕДО, обмін документами, підписання.	Юридично значущий документообіг, поширеність в Україні.	Орієнтація на бухгалтерські та звітні процеси.
Вчасно	Створення, погодження, підписання, обмін і зберігання онлайн.	Зручний вебінтерфейс, хмарна модель.	Залежність від зовнішнього сервісу і тарифів.
DocuWare	Керування документами, архів, автоматизація workflow.	Потужні можливості для бізнес-процесів.	Складність впровадження для невеликих задач.
SharePoint	Спільна робота, документи, портали, інтеграція Microsoft 365.	Інтеграція з офісними інструментами.	Надмірність для простого локального прототипу.
Розроблювана система	Авторизація, документи, статуси, погодження, журнал дій.	Простота, контрольованість, низькі вимоги до запуску.	Потребує подальшого розвитку для промислового використання.

Порівняння аналогів дало змогу визначити функціональне ядро майбутньої системи. Для бакалаврської роботи доцільно реалізувати не надмірно складну корпоративну платформу, а зрозумілий прототип, у якому наочно показано основні процеси: створення документа, його погодження, контроль статусу та збереження історії дій.

Порівняння готових систем також показує, що для різних організацій одна й та сама функція може мати різне значення. Для великого підприємства критичною є інтеграція з бухгалтерією, електронним підписом, корпоративною поштою та зовнішніми сервісами. Для невеликого підрозділу першочерговими можуть бути зовсім інші речі: простий облік документів, швидкий пошук, зрозумілий статус і можливість подивитися історію дій. Тому власний прототип не дублює промислові

системи повністю, а відбирає з них найбільш важливу логіку для базового документообігу.

Такий підхід є виправданим і з погляду подальшої демонстрації результатів роботи. Під час захисту важливо не тільки показати сторінки сайту, а й пояснити, чому саме такі функції були реалізовані. Авторизація показує контроль доступу, реєстр документів — централізоване зберігання, погодження — рух документа між етапами, а журнал дій — прозорість виконаних операцій. Разом ці модулі формують завершений, хоча й навчальний, приклад системи електронного документообігу.

1.4 Аналіз технологій розробки веборієнтованої інформаційної системи

Вибір технологій для реалізації системи електронного документообігу визначався простим критерієм: доступність і практичність у межах навчального проєкту без зайвого ускладнення. PHP як мова серверної частини і MySQL як система керування базами даних добре підтримуються, легко розгортаються в локальному середовищі OSPanel і є прийнятним вибором для CRUD-застосунку такого масштабу.

PHP обробляє запити, управляє сесіями, перевіряє авторизацію, взаємодіє з базою даних, завантажує файли та формує сторінки. Важливо, що PHP рівно однаково підходить як для простих сторінок, так і для структурованих вебзастосунків — а отже, не нав'язує зайвої складності на ранньому етапі і водночас не обмежує подальше зростання.

MySQL обрано за те, що вона природно підтримує реляційну модель, яка тут дуже до речі. Документи, користувачі, статуси, погодження і журнал дій мають чіткі зв'язки між собою. Зовнішні ключі підтримують логічну цілісність — документ не може посилатися на неіснуючого користувача або статус, яких немає в системі.

HTML і CSS відповідають за структуру та оформлення сторінок. JavaScript додає зручність взаємодії на стороні клієнта. Bootstrap, у свою чергу, дозволяє

швидко отримати адаптивний інтерфейс із таблицями, формами, кнопками та навігацією — без написання складного CSS вручну. Для системи документообігу важливо, щоб вона була не лише технічно коректною, а й зрозумілою для звичайного користувача.

Безпека паролів реалізована через password_hash() і password_verify(). Навіть якщо базу даних вдасться скомпрометувати, витягти початковий пароль із хешу значно складніше, ніж із відкритого тексту. Для SQL-запитів використовуються підготовлені вирази mysqli — вони виключають можливість SQL-ін'єкцій через відокремлення параметрів від тексту запиту.

Архітектурно система організована як спрощений модульний проєкт: конфігурація, допоміжні функції, сторінки авторизації, сторінки документів, адміністративні сторінки і каталог завантажених файлів. Це не повноцінний MVC, але вже має логічний поділ, достатній для бакалаврської роботи. Перехід до повноцінного MVC-проєкту можливий на наступному етапі розвитку.

Обґрунтування вибору технологій наведено в таблиці 1.5.

Таблиця 1.5 – Обґрунтування вибору технологій

Технологія	**Призначення**	**Причина вибору**
PHP	Серверна логіка вебзастосунку	Простота запуску, підтримка форм, сесій, файлів і MySQL.
MySQL	Зберігання структурованих даних	Реляційна модель, зовнішні ключі, висока поширеність.
HTML/CSS	Структура та оформлення сторінок	Базові технології вебінтерфейсу.
JavaScript	Додаткова взаємодія на клієнті	Можливість покращити зручність форм і дій користувача.
Bootstrap	Адаптивний інтерфейс	Готові компоненти для таблиць, форм і навігації.
OSPanel	Локальне середовище запуску	Зручний запуск Apache, PHP, MySQL та phpMyAdmin.

Отже, обраний стек технологій відповідає поставленій задачі та рівню навчального проєкту. Він дозволяє реалізувати систему у вигляді локального вебзастосунку, швидко перевірити її роботу, підготувати демонстраційні матеріали та за потреби розширити функціональність у майбутньому.

Важливою перевагою обраного технологічного стеку є його доступність для локального розгортання. Для демонстрації системи не потрібно налаштовувати складний сервер або купувати комерційну платформу. Достатньо локального середовища, браузера, бази даних і набору PHP-файлів. Це дає змогу швидко перевіряти зміни, редагувати окремі модулі й одразу бачити результат у браузері. Для бакалаврської роботи така простота є суттєвою перевагою, оскільки дозволяє зосередитися саме на логіці документообігу.

Водночас простота не означає відсутність структури. У проєкті окремо виділяються налаштування підключення, службові функції, сторінки інтерфейсу, робота з базою даних і завантаження файлів. Такий поділ не є повноцінним складним фреймворком, однак він формує зрозумілу основу, на якій можна пояснити роботу системи та поступово її розширювати.

1.5 Висновки до розділу 1

У першому розділі опрацьовано предметну область електронного документообігу: визначено ключові поняття, проаналізовано переваги переходу від паперової моделі до електронної та виявлено супутні ризики. Стало зрозуміло, що повноцінна система документообігу — це не лише файлове сховище, а процесна логіка: статуси, маршрути, ролі, журнал дій і механізми контролю доступу.

Розгляд нормативних засад показав, що проєктування не можна відривати від вимог до ідентифікації користувачів, збереження даних та можливості інтеграції електронного підпису в майбутньому. Аналіз систем-аналогів дав змогу виділити функціональне ядро, яке доцільно реалізувати у власному прототипі: авторизація, керування документами, погодження, пошук і журнал дій.

На підставі аналізу обґрунтовано стек технологій PHP, MySQL, HTML, CSS, JavaScript і Bootstrap. Він достатній для розробки веборієнтованої системи, яку можна реалізувати, протестувати і переконливо продемонструвати в межах кваліфікаційної роботи.

2 ПРОЄКТУВАННЯ ТА АРХІТЕКТУРА ІНФОРМАЦІЙНОЇ СИСТЕМИ ЕЛЕКТРОННОГО ДОКУМЕНТООБІГУ

2.1 Постановка задачі та вимоги до інформаційної системи

Результатом аналізу предметної області стала чітка постановка задачі. Потрібно розробити веборієнтовану інформаційну систему, яка реалізує базовий цикл роботи з електронними документами: їх створення, збереження, перегляд, редагування, пошук, надсилання на погодження та зміну статусів.

Цільова аудиторія системи — невелика організація або навчальний підрозділ, де потрібен зрозумілий внутрішній документообіг без корпоративної інфраструктури. Система має працювати через браузер, зберігати дані в MySQL, використовувати PHP для серверної логіки та підтримувати розподіл ролей між користувачами.

Два головні типи користувачів — звичайний і адміністратор. Звичайний користувач створює документи, редагує власні записи, відстежує статуси і надсилає матеріали на погодження. Адміністратор має ширше коло повноважень: переглядає всі документи, погоджує або відхиляє їх, контролює журнал дій і список облікових записів.

Функціональні вимоги описують конкретні можливості системи: авторизація, створення документа, завантаження файлу, редагування, перегляд, пошук, фільтрація, погодження, зміна статусу та фіксація ключових дій у журналі. Разом ці вимоги формують функціональне ядро програмного забезпечення.

Нефункціональні вимоги стосуються якостей системи, а не її конкретних функцій. Вона має бути зручною, стабільною, достатньо захищеною і придатною до подальшого розширення. Для навчального проєкту особливо важливо, щоб код мав зрозумілу структуру, яку можна пояснити під час захисту.

Оскільки система працює з документами, що можуть містити службову інформацію, контроль доступу не можна зводити лише до меню. Перевірка ролі має

виконуватися на серверному рівні — так, щоб навіть пряме введення URL у браузері не давало несанкціонованого доступу до захищених сторінок.

Вимоги до зберігання даних побудовані за принципом розділення: інформація про документи зберігається в базі даних (назва, опис, автор, статус, дата, шлях до файлу), тоді як самі файли розміщуються в каталозі uploads. Підхід простий і зрозумілий для реалізації, а в перспективі може бути замінений на хмарне сховище.

Функціональні вимоги до інформаційної системи наведено в таблиці 2.1.

Таблиця 2.1 – Функціональні вимоги до інформаційної системи

Код	**Вимога**	**Опис**
FR-1	Авторизація користувачів	Користувач входить до системи за email і паролем.
FR-2	Створення документа	Користувач може додати назву, опис і файл документа.
FR-3	Редагування документа	Автор може змінити дані власного документа.
FR-4	Пошук документів	Система підтримує пошук за назвою або описом.
FR-5	Погодження документа	Адміністратор може затвердити або відхилити документ.
FR-6	Журнал дій	Система фіксує важливі операції користувачів.
FR-7	Керування користувачами	Адміністратор переглядає облікові записи користувачів.

Нефункціональні вимоги до інформаційної системи наведено в таблиці 2.2.

Таблиця 2.2 – Нефункціональні вимоги до інформаційної системи

Група вимог	**Зміст вимоги**	**Очікуваний результат**
Зручність	Інтерфейс має бути простим і зрозумілим.	Користувач швидко знаходить потрібні функції.
Безпека	Доступ лише після авторизації.	Документи захищені від стороннього перегляду.
Надійність	Основні операції виконуються стабільно.	Документи не губляться після збереження.
Масштабованість	Система дозволяє додавати нові модулі.	Можлива інтеграція КЕП, PDF та повідомлень.
Супровід	Код і база мають логічну структуру.	Простіше вносити зміни та виправлення.

Сформовані вимоги стали основою для подальшого проєктування системи. Вони дозволяють не розглядати програму як набір окремих сторінок, а визначити її як цілісну інформаційну систему з конкретними користувачами, функціями, обмеженнями доступу та очікуваними результатами роботи.

Під час формування вимог враховувалося, що користувач не повинен витрачати багато часу на розуміння логіки роботи системи. У більшості випадків йому потрібно виконати одну з кількох типових дій: створити документ, знайти вже створений документ, перевірити його статус або передати на погодження. Саме тому функціональні вимоги сформульовано навколо реальних сценаріїв, а не навколо абстрактного набору кнопок і сторінок.

Нефункціональні вимоги у цій роботі мають не менше значення, ніж функціональні. Якщо система формально дозволяє створювати документи, але її незручно використовувати, практична цінність такого рішення буде низькою. Тому під час розробки враховувалися простота інтерфейсу, логічність меню, однакове оформлення сторінок і можливість швидко повернутися до основних розділів. Це робить прототип ближчим до реального використання, а не лише до технічної демонстрації.

2.2 Архітектура системи та структура функціональних модулів

Для системи електронного документообігу обрано клієнт-серверну архітектуру — один із найперевіреніших підходів для вебзастосунків. Логіка тут проста: клієнтська частина відповідає за те, що бачить і робить користувач у браузері, а серверна — за все, що відбувається «за лаштунками»: обробку запитів, перевірку прав доступу, роботу з базою даних і виконання бізнес-логіки. Такий поділ дозволяє чітко розмежувати відповідальність між компонентами і не змішувати різні рівні системи в одному місці.

Взаємодія виглядає так: користувач відкриває браузер, потрапляє на сторінку авторизації, вводить свої дані і після успішного входу бачить головну панель. Будь-який наступний запит — чи то до документів, чи до облікових записів — іде на сервер. Там PHP-скрипти перевіряють сесію, роль і параметри запиту, і лише після цього виконують потрібні дії.

Серверна частина не є монолітом — вона поділена на окремі модулі: підключення до бази даних, авторизація, документи, погодження, журнал дій і адміністрування. Завдяки цьому система сприймається не як набір розрізнених скриптів, а як структурований продукт, де у кожної частини є своє чітке призначення.

Клієнтська частина побудована на HTML, CSS, Bootstrap і JavaScript. Її завдання — зрозуміло і зручно відображати форми, таблиці, кнопки й повідомлення про результат операцій. Тут важливий один нюанс: видимість елементів меню залежить від ролі користувача, але це лише зручність інтерфейсу. Остаточна перевірка прав завжди відбувається на сервері — інтерфейс не може і не повинен бути єдиним захистом.

Центральним у системі є модуль документів. Саме він відповідає за створення, редагування, перегляд, видалення та пошук. Кожен документ прив'язаний до автора і має поточний статус. Якщо до документа прикріплено файл, у базі даних зберігається шлях до нього, а сам файл розміщується в каталозі uploads.

Модуль погодження відповідає за зміну статусу документа. Після того як користувач надсилає документ на погодження, він з'являється у відповідному розділі адміністратора. Той розглядає його і приймає рішення: документ отримує статус «Затверджено» або «Відхилено» — і це рішення фіксується в системі.

Журнал дій — допоміжний, але принципово важливий компонент. Він фіксує всі ключові події: створення, редагування, надсилання, затвердження, відхилення та видалення документів. У перспективі цей модуль може бути розширений до повноцінного аудиту з фільтрами і розширеною аналітикою.

Адміністративний модуль поки що відносно простий: перегляд користувачів і загальний стан системи. Але його структура дозволяє без проблем додати керування ролями, блокування акаунтів або розширені налаштування, коли виникне така потреба.

Загальна архітектура інформаційної системи показано на рисунку 2.1.

Рисунок 2.1 – Загальна архітектура інформаційної системи

Функціональні модулі системи наведено в таблиці 2.3.

Таблиця 2.3 – Функціональні модулі системи

Модуль	**Призначення**	**Основні операції**
Авторизація	Перевірка користувача і створення сесії.	Вхід, вихід, реєстрація.

Кінець таблиці 2.3

Модуль	Призначення	Основні операції
Документи	Робота з електронними документами.	Створення, редагування, перегляд, пошук.
Погодження	Опрацювання документів адміністратором.	Надсилання, затвердження, відхилення.
Журнал дій	Фіксація активності користувачів.	Запис подій, перегляд історії.
Адміністрування	Контроль користувачів і доступу.	Перегляд користувачів, контроль ролей.

Поділ системи на функціональні модулі спрощує розробку і подальше пояснення програмного продукту. Кожен модуль відповідає за окрему групу дій, а разом вони формують повний цикл роботи з документом — від авторизації користувача до погодження документа і запису події в журналі.

Обрана архітектура дозволяє легко пояснити рух даних у системі. Коли користувач заповнює форму створення документа, браузер передає введені дані на сервер. PHP-скрипт перевіряє сесію, обробляє форму, зберігає запис у базі даних і за потреби переносить прикріплений файл до окремого каталогу. Після цього користувач бачить сторінку документа або повідомлення про результат операції. Така послідовність є простою, але саме вона відображає типову роботу більшості веборієнтованих інформаційних систем.

Окремий поділ на модулі також полегшує подальше тестування. Якщо виникає помилка під час входу, перевіряється модуль авторизації; якщо не зберігається документ — модуль роботи з документами та база даних; якщо не змінюється статус — модуль погодження. Завдяки цьому проєкт легше підтримувати, адже кожна частина має своє призначення і не змішується з іншими функціями.

2.3 Проєктування бази даних і інформаційної моделі

Для системи обрано реляційну модель даних — вона природно відповідає предметній області. Основні сутності тут мають чіткі й логічні зв'язки: документ належить користувачу, документ має статус, погодження пов'язане з конкретним документом і адміністратором, а журнал дій посилається на того, хто цю дію виконав.

База даних складається з п'яти таблиць: users, documents, document_statuses, approvals і logs. Кожна виконує свою роль і не дублює логіку сусідніх. Users зберігає облікові записи, documents — відомості про документи, document_statuses — довідник можливих статусів, approvals — результати погодження, logs — журнал дій користувачів.

У таблиці users пароль зберігається у вигляді хешу — це базова вимога безпеки для будь-якої системи з авторизацією. Поле role розмежовує звичайних користувачів і адміністраторів, визначаючи, що кожен із них може робити в системі.

У таблиці documents зберігаються назва, опис, шлях до файлу, автор і статус. Самі файли в базі не зберігаються — лише шлях до них у файловому сховищі. Це не просто технічне рішення: воно полегшує резервне копіювання, зменшує навантаження на базу і спрощує розгортання системи на локальному середовищі.

Таблиця approvals спроєктована з думкою про майбутнє. Навіть якщо зараз погодження одноетапне, ця структура дозволяє без суттєвих змін додати кількох погоджувачів або прикріплювати коментарі до рішень — просто розширивши наявну логіку.

Зовнішні ключі в базі забезпечують цілісність даних: документ не може посилатися на неіснуючого користувача чи статус. Це запобігає появі «сирітських» записів і знімає з прикладного коду частину відповідальності за ручну перевірку коректності.

ER-модель бази даних інформаційної системи показано на рисунку 2.2.

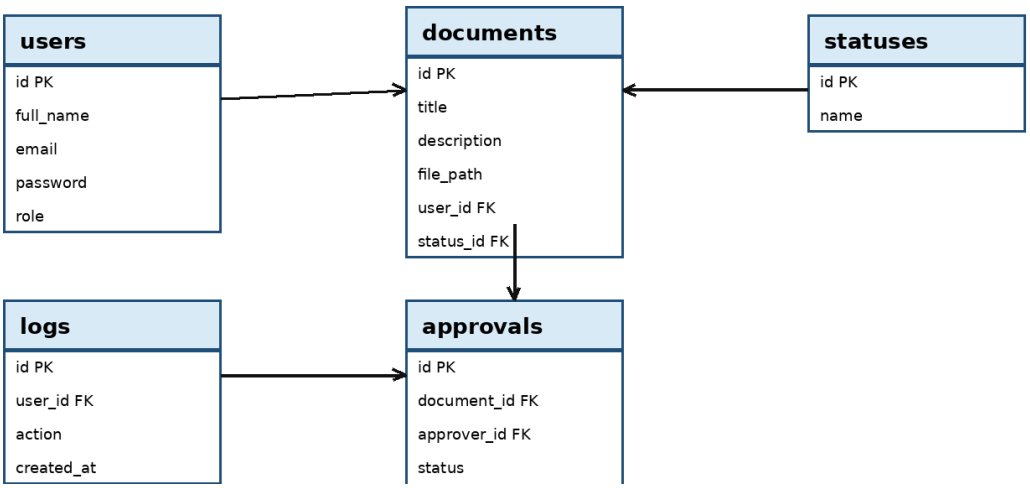

Рисунок 2.2 – ER-модель бази даних інформаційної системи

Структура основних таблиць бази даних наведено в таблиці 2.4.

Таблиця 2.4 – Структура основних таблиць бази даних

Таблиця	Призначення	Ключові поля
users	Зберігання користувачів системи.	id, full_name, email, password, role
documents	Зберігання відомостей про документи.	id, title, description, file_path, user_id, status_id
document_statuses	Довідник статусів документів.	id, name
approvals	Фіксація рішень щодо погодження.	id, document_id, approver_id, status, comment
logs	Журнал дій користувачів.	id, user_id, action, created_at

Запропонована структура є достатньою для реалізації першої робочої версії системи. Водночас вона не обмежує розвиток проєкту — за потреби її можна розширити без необхідності переробляти вже наявну логіку.

При проєктуванні бази даних важливо було не допустити змішування різних типів інформації в одній таблиці. Наприклад, дані користувача, опис документа, історія погодження і журнал активності мають різне призначення. Якщо зберігати

їх разом, система швидко стане складною для підтримки. Тому кожна сутність винесена окремо, а зв’язки між ними задаються через ідентифікатори.

Такий підхід має і практичну перевагу: він спрощує виведення інформації на сторінках сайту. Реєстр документів отримує назву документа, автора, тип і статус із пов’язаних таблиць. Сторінка перегляду документа може додатково показати історію обробки та прикріплені файли. Журнал дій, у свою чергу, дає змогу простежити, які операції виконувалися користувачами. У результаті база даних працює не як простий архів, а як основа всієї логіки системи.

2.4 Модель ролей, сценарії взаємодії та контроль доступу

Розмежування доступу — одна з ключових вимог до системи документообігу. Якщо документи містять службову інформацію, користувачі мають отримувати лише ті можливості, що відповідають їхній ролі. У розроблюваній системі передбачено дві ролі: user і admin.

Роль user — звичайний працівник, який формує документи та відстежує їх рух. Він може створювати документи, переглядати власні записи, редагувати їх, шукати потрібну інформацію і надсилати документи на погодження. Ця роль відображає типову поведінку виконавця.

Роль admin охоплює ширший спектр повноважень. Адміністратор переглядає всі документи, змінює їх статуси, погоджує або відхиляє, контролює журнал дій і список користувачів. У реальній організації цю функцію може виконувати керівник підрозділу, секретар або призначена відповідальна особа.

Типовий сценарій роботи звичайного користувача починається з авторизації. Після входу він бачить головну панель із доступними діями: може створити документ, заповнити назву та опис, прикріпити файл і зберегти запис. Коли документ готовий — надіслати на погодження.

Сценарій адміністратора схожий за початком, але після входу йому доступні додаткові пункти меню. У розділі погодження він бачить документи, що очікують

рішення, і може затвердити або відхилити кожен із них. Після рішення система автоматично змінює статус документа і фіксує подію в журналі.

Принципово важливо, що контроль доступу не може базуватися лише на прихованих кнопках. Користувач може вручну ввести URL адміністративної сторінки. Тому кожна захищена сторінка перевіряє сесію та роль на серверному рівні — незалежно від того, що видно в інтерфейсі.

Поточна модель ролей проста, але її вистачає для першої версії системи. У перспективі можна додати роль «погоджувач», «керівник», «архіваріус» і побудувати таблицю прав із гнучким налаштуванням дозволів для кожної ролі.

Сценарії взаємодії користувачів із системою показано на рисунку 2.3.

Рисунок 2.3 – Сценарії взаємодії користувачів із системою

Права доступу користувачів наведено в таблиці 2.5.

Таблиця 2.5 – Права доступу користувачів

Функція	**Користувач**	**Адміністратор**
Вхід до системи	так	так
Створення документа	так	так
Редагування власного документа	так	так

Кінець таблиці 2.5

Функція	Користувач	Адміністратор
Перегляд усіх документів	ні	так
Надсилання на погодження	так	так
Затвердження документа	ні	так
Перегляд журналу дій	ні	так
Керування користувачами	ні	так

Наведена модель доступу показує, які можливості отримує кожна роль у системі. Завдяки цьому ще на етапі проєктування визначено межі відповідальності користувачів і закладено основу для безпечної роботи з документами.

Модель ролей у цій системі свідомо зроблена простою. Це дозволяє уникнути зайвої складності в першій версії та водночас показати основний принцип розмежування доступу. Звичайний користувач працює переважно зі своїми документами, керівник або адміністратор отримує доступ до погодження, а адміністратор додатково контролює користувачів і журнал дій. Для навчального проєкту така модель є достатньо зрозумілою і легко перевіряється під час демонстрації.

Водночас навіть проста модель ролей має бути реалізована не лише в інтерфейсі. Якщо приховати пункт меню, але не перевіряти права на сервері, користувач зможе спробувати відкрити захищену сторінку напряму. Саме тому в системі передбачено серверні функції перевірки доступу. Це підкреслює, що безпека вебзастосунку не повинна залежати тільки від того, які кнопки бачить користувач.

2.5 Узагальнення проєктних рішень

У другому розділі сформульовано постановку задачі та зафіксовано функціональні й нефункціональні вимоги до інформаційної системи. Обґрунтовано клієнт-серверну архітектуру та модульну структуру програмного забезпечення.

Спроєктовано базу даних із п'ятьма таблицями, між якими встановлено зв'язки через зовнішні ключі.

Запропонована модель ролей і сценарії взаємодії користувачів дають змогу перейти від теоретичного проєктування до реальної реалізації. Обрана архітектура є достатньою для прототипу і не закриває шляху до подальшого розширення — нові модулі можна додавати без повного перепроєктування системи.

3 РОЗРОБКА ПРОГРАМНОГО ПРОТОТИПУ СИСТЕМИ

3.1 Структура програмного проєкту

Практична частина роботи — це не просто набір PHP-сторінок, а структурований проєкт, кожен компонент якого має зрозуміле призначення. Така організація полегшує як розробку, так і демонстрацію системи на захисті.

Проєкт розміщено у локальному середовищі OSPanel у каталозі доменів, у папці document_flow_system. Вона містить файли конфігурації, сторінки авторизації, сторінки роботи з документами, адміністративні сторінки, каталог стилів і каталог для завантажених файлів.

Файл config.php містить параметри підключення до бази даних. В окремому файлі functions.php зосереджені допоміжні функції: перевірка авторизації, перевірка ролі адміністратора, запис події до журналу і безпечне виведення тексту. Такий поділ усуває дублювання коду між сторінками.

Файли header.php і footer.php використовуються для спільної частини інтерфейсу: початок HTML-документа, підключення Bootstrap, меню, перевірка сесії (header) і закриття основних блоків із підключенням скриптів (footer).

Сторінки documents.php, document_create.php, document_view.php, document_edit.php і document_delete.php охоплюють весь функціонал роботи з документами. Approvals.php призначена для погодження документів адміністратором, users.php — для перегляду користувачів, logs.php — для журналу дій.

Структура не перевантажена складними фреймворками, але вже має логічний поділ, де кожен файл має чітке призначення. При необхідності проєкт можна переорганізувати у повноцінну MVC-архітектуру з окремими папками controllers, models і views. На поточному етапі обраний варіант є практичним і цілком достатнім.

Структура програмного проєкту інформаційної системи електронного документообігу показано на рисунку 3.1.

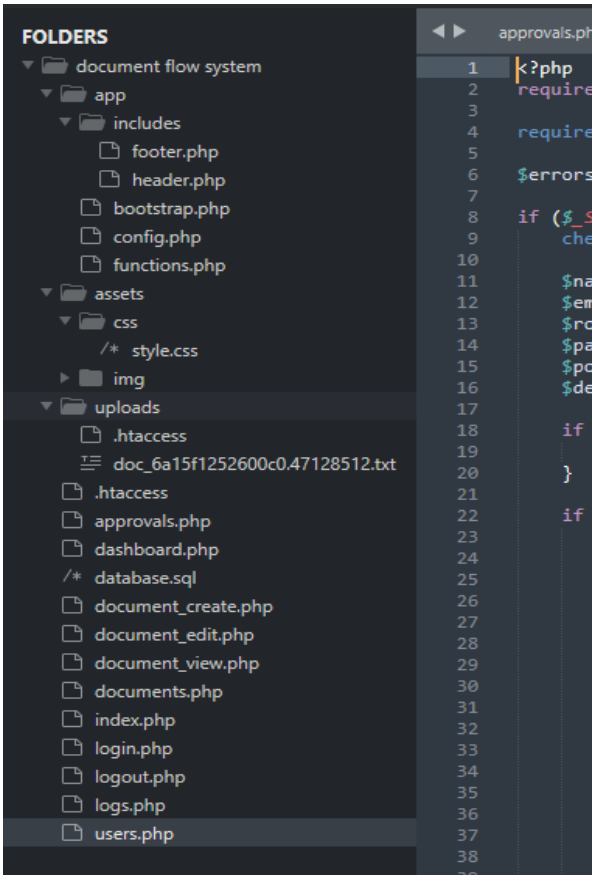

Рисунок 3.1 – Структура програмного проєкту інформаційної системи електронного документообігу

Орієнтовна структура програмного проєкту наведено в таблиці 3.1.

Таблиця 3.1 – Орієнтовна структура програмного проєкту

Файл / каталог	**Призначення**
config.php	Підключення до бази даних MySQL.
functions.php	Допоміжні функції авторизації, ролей, журналу та безпечного виведення.
login.php	Форма входу до системи.
register.php	Створення нового користувача.
dashboard.php	Головна панель після авторизації.
documents.php	Список документів і пошук.

Кінець таблиці 3.1

Файл / каталог	**Призначення**
document_create.php	Створення нового документа.
document_view.php	Перегляд документа і надсилання на погодження.
approvals.php	Погодження документів адміністратором.
logs.php	Перегляд журналу дій.
uploads/	Каталог для завантажених файлів.

Така структура проєкту робить систему зрозумілою для супроводу. Окреме розміщення конфігураційних файлів, сторінок інтерфейсу, стилів і каталогу завантажень дозволяє швидко знаходити потрібні частини коду та поступово доопрацьовувати програму без повного переписування.

Структура програмного проєкту була побудована так, щоб основні файли можна було швидко знайти й пояснити. Це особливо важливо для кваліфікаційної роботи, де потрібно не лише мати робочий код, а й уміти показати його логіку. Наприклад, файли header.php і footer.php відповідають за спільні частини сторінок, functions.php містить повторювані службові функції, а окремі сторінки document_create.php, document_view.php і approvals.php відповідають за конкретні сценарії роботи користувача.

Перевага такої структури проявляється під час внесення змін. Якщо потрібно змінити верхнє меню, достатньо відредагувати один файл header.php, а не кожну сторінку окремо. Якщо потрібно змінити стиль кнопок або карток, це робиться у style.css. Якщо потрібно уточнити перевірку ролей, редагується functions.php. Завдяки цьому система залишається керованою навіть після додавання нових сторінок і функцій.

3.2 Реалізація бази даних та підключення до MySQL

Першим практичним кроком стало створення бази даних document_flow із п'ятьма таблицями: users, documents, document_statuses, approvals і logs. Структура бази відповідає інформаційній моделі, описаній у другому розділі.

Структуру бази даних інформаційної системи у phpMyAdmin показано на рисунку 3.2.

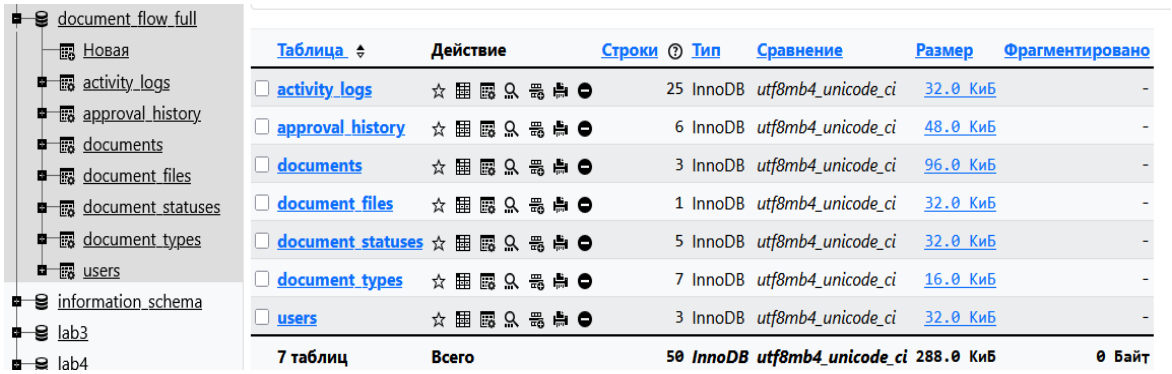

Рисунок 3.2 – Структура бази даних інформаційної системи у phpMyAdmin

Таблиця users зберігає облікові дані користувачів. Email тут є унікальним — саме він слугує ідентифікатором при вході в систему. Пароль зберігається у вигляді хешу, а поле role визначає рівень доступу. Кожен новий користувач за замовчуванням отримує роль user — без зайвих привілеїв.

Таблиця document_statuses містить чотири базові статуси: «Створено», «На погодженні», «Затверджено» і «Відхилено». Виокремлення статусів в окрему таблицю — технічно невеликий, але продуманий крок: щоб додати новий статус у майбутньому, не потрібно торкатися структури таблиці documents взагалі.

Підключення до бази реалізовано через об'єкт mysqli у файлі config.php. Там само встановлюється кодування utf8mb4 — без нього коректна робота з українськими назвами і описами документів була б неможливою. Усі PHP-сторінки, що звертаються до бази, підключають саме цей файл.

Налаштування підключення до бази даних MySQL показано на рисунку 3.3.

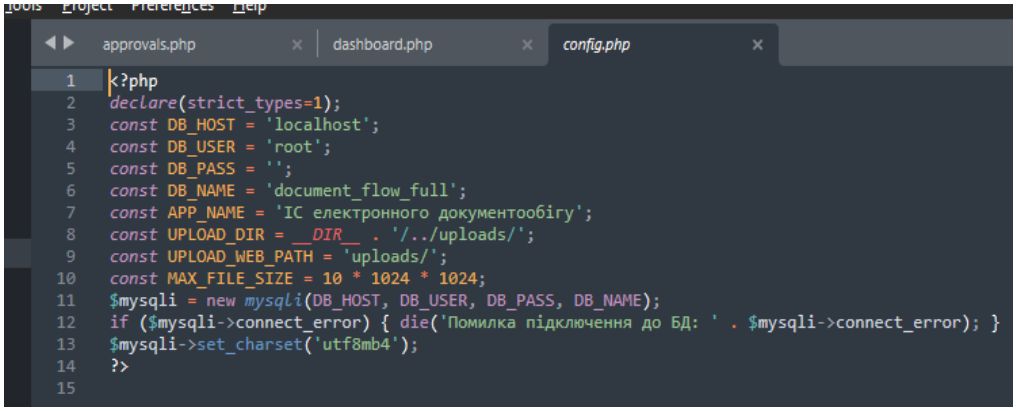

Рисунок 3.3 – Налаштування підключення до бази даних MySQL

Такий підхід має очевидну практичну перевагу: якщо параметри підключення зміняться, достатньо відредагувати лише один файл — і це відобразиться скрізь.

Усі запити, що приймають дані від користувача, реалізовані з використанням підготовлених виразів. Параметри передаються окремо від тексту SQL-запиту — це надійний і перевірений захист від SQL-ін'єкцій у будь-яких полях введення.

Лістинг 3.1 – Фрагмент SQL-структури бази даних

```
CREATE DATABASE document_flow CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
USE document_flow;

CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    full_name VARCHAR(150) NOT NULL,
    email VARCHAR(150) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    role ENUM('admin', 'user') DEFAULT 'user',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE documents (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    description TEXT,
    file_path VARCHAR(255),
    user_id INT NOT NULL,
    status_id INT DEFAULT 1,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id),
    FOREIGN KEY (status_id) REFERENCES document_statuses(id)
);
```

Лістинг 3.2 – Підключення до бази даних

```
<?php
$host = 'localhost';
$user = 'root';
$password = '';
```

```
$db = 'document_flow';

$conn = new mysqli($host, $user, $password, $db);

if ($conn->connect_error) {
    die('Помилка підключення до бази даних: ' . $conn->connect_error);
}

$conn->set_charset('utf8mb4');
?>
```

3.3 Реалізація авторизації та роботи з документами

Авторизація — перший рубіж захисту системи. Після отримання email і пароля PHP шукає відповідний запис у таблиці users і перевіряє пароль функцією password_verify(). При успішному вході в сесію записуються ідентифікатор, ім'я та роль користувача.

Сторінка авторизації користувача показано на рисунку 3.4.

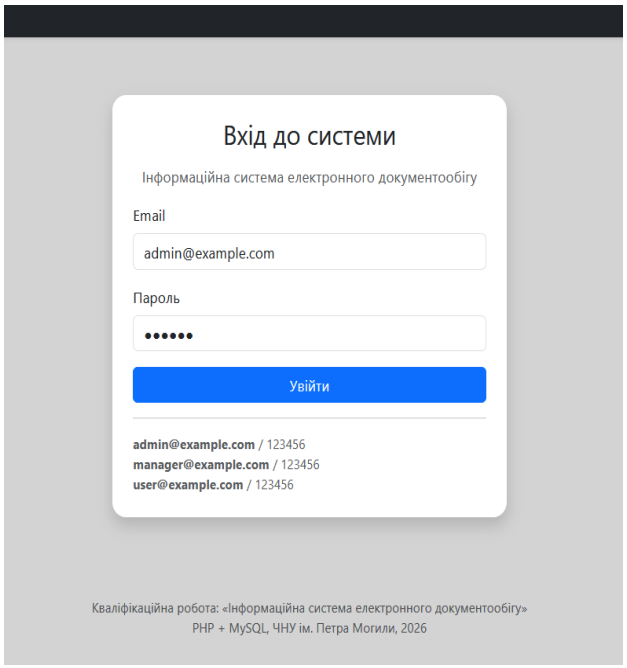

Рисунок 3.4 – Сторінка авторизації користувача

Сесія використовується для ідентифікації користувача під час роботи з системою. На кожній захищеній сторінці викликається requireLogin() — якщо сесії немає, система перенаправляє на сторінку входу. Для адміністративних сторінок додатково перевіряється роль через requireAdmin().

Створення документа відбувається через форму: назва, опис і необов'язковий файл.

Форма створення нового електронного документа показано на рисунку 3.5.

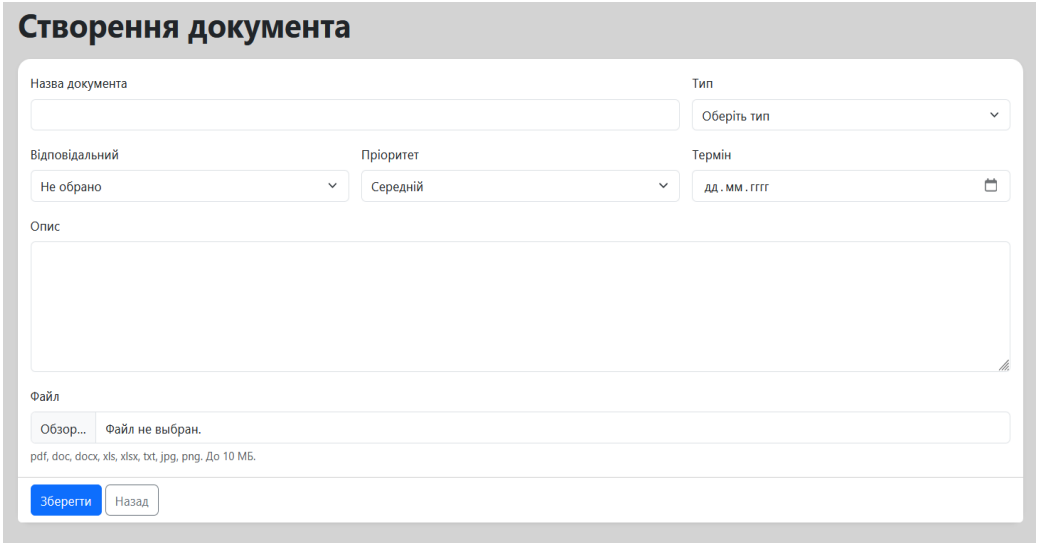

Рисунок 3.5 – Форма створення нового електронного документа

PHP перевіряє дані, якщо є файл — зберігає його в uploads і фіксує шлях. Новий документ автоматично отримує статус «Створено» і прив'язується до поточного користувача.

Список документів на сторінці documents.php відображається з урахуванням ролі: адміністратор бачить усі записи, звичайний користувач — тільки власні.

Реєстр електронних документів показано на рисунку 3.6.

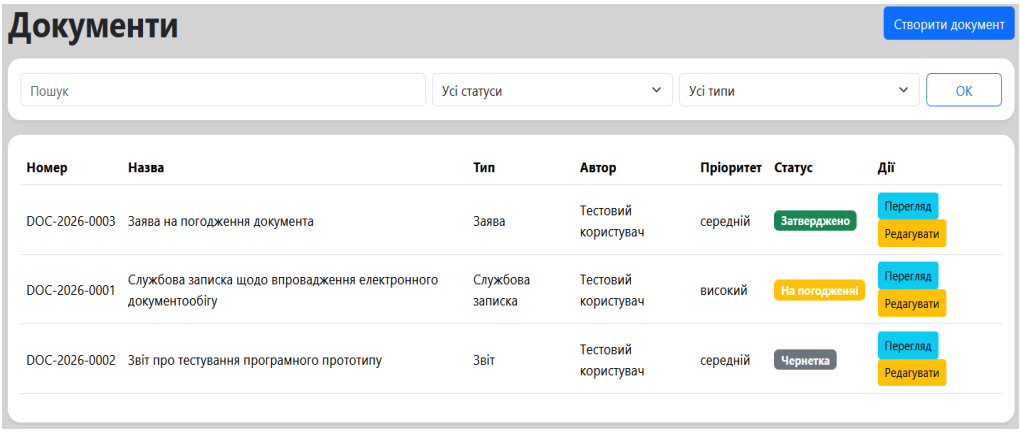

Рисунок 3.6 – Реєстр електронних документів

Це реалізовано через умову в SQL-запиті. Приховати документи лише на рівні інтерфейсу недостатньо — реальний захист забезпечується на серверному рівні.

Пошук за назвою або описом реалізовано через оператор LIKE.

Реалізація пошуку електронних документів показано на рисунку 3.7.

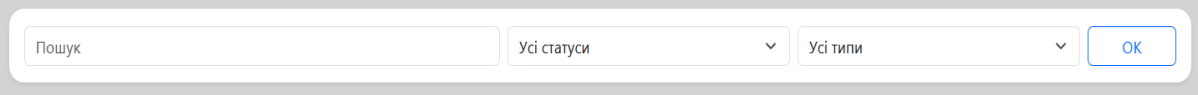

Рисунок 3.7 – Реалізація пошуку електронних документів

Навіть у простому варіанті він суттєво підвищує зручність роботи з системою, особливо коли кількість документів зростає.

Редагування доступне автору або адміністратору. Після збереження змін система фіксує подію в журналі. Видалення також потребує підтвердження і обов'язково записується в журнал. Виведення всіх даних на сторінках проходить через htmlspecialchars() — захист від XSS-атак.

Лістинг 3.3 – Перевірка авторизації користувача

```
if (password_verify($password, $userData['password'])) {
    $_SESSION['user_id'] = $userData['id'];
    $_SESSION['full_name'] = $userData['full_name'];
    $_SESSION['role'] = $userData['role'];
    header('Location: dashboard.php');
    exit;
} else {
    $error = 'Невірний пароль';
}
```

Лістинг 3.4 – Додавання нового документа

```
$stmt = $conn->prepare(
    'INSERT INTO documents (title, description, file_path, user_id) VALUES (?, ?,
?, ?)'
);
$stmt->bind_param('sssi', $title, $description, $file_path, $user_id);
$stmt->execute();
```

Реалізація авторизації тісно пов’язана з подальшою логікою роботи системи. Після входу користувача система не просто відкриває головну сторінку, а запам’ятовує його роль у сесії. Завдяки цьому на різних сторінках можна перевіряти, чи має користувач право створювати документ, редагувати його, переглядати всі записи або працювати з журналом дій. Такий механізм робить поведінку системи послідовною.

Модуль документів реалізує основний робочий цикл. Користувач вводить дані у форму, система перевіряє обов’язкові поля, створює запис у базі даних і

присвоює документу початковий статус. Якщо файл додано, він зберігається у спеціальному каталозі, а в базі залишається шлях до нього. Це рішення зручне для локального прототипу, бо дозволяє не перевантажувати базу великими файлами і водночас зберігати зв’язок між документом та його вкладенням.

Важливо, що робота з документом не завершується після створення. У системі передбачено перегляд, редагування, передавання на погодження та контроль поточного статусу. Така послідовність наближає прототип до реального документообігу, де документ проходить кілька етапів, а не просто зберігається у вигляді запису в таблиці.

3.4 Реалізація погодження та журналу дій

Механізм погодження — серцевина процесної логіки системи. Без нього вона була б лише файловим сховищем. В реалізованому прототипі погодження відбувається через зміну статусу документа: користувач надсилає документ, адміністратор бачить його у списку і приймає рішення.

Коли документ переходить у статус «На погодженні», він з'являється на сторінці approvals.php.

Сторінка погодження електронних документів показано на рисунку 3.8.

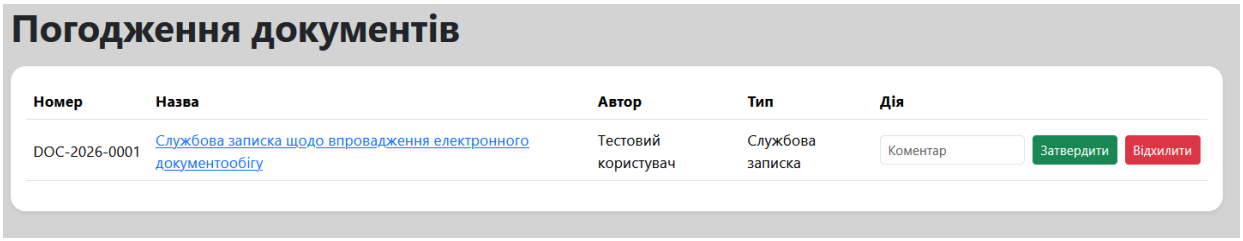

Рисунок 3.8 – Сторінка погодження електронних документів

Адміністратор може затвердити або відхилити його. У разі відхилення доцільно вказати коментар — щоб автор розумів, що потрібно виправити. Після прийняття рішення в таблицю approvals записується інформація про документ, адміністратора, рішення, коментар і дату.

Журнал дій реалізовано через функцію addLog().

Журнал дій користувачів інформаційної системи показано на рисунку 3.9.

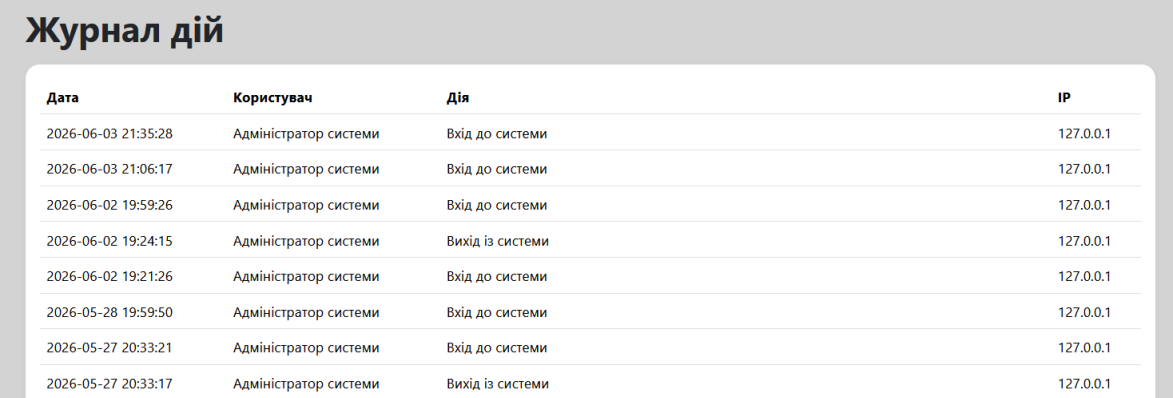

Рисунок 3.9 – Журнал дій користувачів інформаційної системи

Вона приймає ідентифікатор користувача і текст дії, додає запис до таблиці logs. Виклик функції вбудований у всі ключові операції: вхід і вихід, створення, редагування, надсилання на погодження, затвердження, відхилення і видалення.

Крім практичної цінності для безпеки, журнал корисний і на захисті: можна наочно показати, як після кожної дії в таблиці з'являється новий запис. Це підтверджує, що система не просто змінює дані, а відслідковує весь процес.

У подальшому журнал можна розширити: додати IP-адресу, тип браузера, старий і новий статус, а також фільтри за датою та типом події. На поточному етапі базового варіанта цілком достатньо для демонстрації основної ідеї.

Лістинг 3.5 – Оновлення статусу документа

```
$stmt = $conn->prepare('UPDATE documents SET status_id = ? WHERE id = ?');
$stmt->bind_param('ii', $status_id, $document_id);
$stmt->execute();
```

Лістинг 3.6 – Функція запису до журналу дій

```
function addLog($conn, $user_id, $action) {
    $stmt = $conn->prepare('INSERT INTO logs (user_id, action) VALUES (?, ?)');
    $stmt->bind_param('is', $user_id, $action);
    $stmt->execute();
}
```

Події, які фіксуються у журналі дій наведено в таблиці 3.2.

Таблиця 3.2 – Події, які фіксуються у журналі дій

Подія	Коли створюється запис	Навіщо це потрібно
Вхід користувача	Після успішної авторизації.	Контроль активності облікових записів.

Кінець таблиці 3.2

Подія	Коли створюється запис	Навіщо це потрібно
Створення документа	Після додавання нового запису.	Фіксація автора і часу створення.
Редагування документа	Після збереження змін.	Контроль зміни змісту документа.
Надсилання на погодження	Після зміни статусу на «На погодженні».	Відстеження початку процесу погодження.
Затвердження / відхилення	Після рішення адміністратора.	Фіксація управлінського рішення.
Видалення документа	Після видалення запису.	Контроль критичних операцій.

Фіксація подій у журналі є важливою частиною практичної реалізації, оскільки саме вона забезпечує прозорість роботи системи. Користувачі виконують операції через інтерфейс, але для адміністратора залишається технічний слід, який дозволяє відновити послідовність дій.

Механізм погодження є ключовим для демонстрації руху документа. Поки документ перебуває у стані чернетки, автор може працювати з ним і вносити зміни. Після надсилання на погодження документ переходить у інший статус і стає доступним для прийняття рішення відповідальною особою. Такий підхід дозволяє показати не лише зберігання документів, а й їхній маршрут у межах системи.

Журнал дій доповнює механізм погодження і робить роботу системи прозорішою. Коли користувач створює документ, надсилає його на погодження або адміністратор приймає рішення, система зберігає відповідний запис. Це важливо для контролю, адже в будь-який момент можна побачити, які дії виконувалися і ким саме. У реальних умовах такий журнал допомагає розбирати спірні ситуації та відновлювати послідовність подій.

3.5 Попередні результати реалізації

На поточному етапі підготовлено функціональний прототип, готовий до запуску в локальному середовищі OSPanel. База даних створена, авторизація працює, реалізовано головну панель, сторінки документів, модуль погодження, перегляд користувачів і журнал дій.

Головна панель інформаційної системи показано на рисунку 3.10.

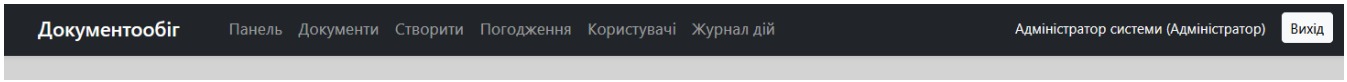

Рисунок 3.10 – Головна панель інформаційної системи

Для тестування є два облікові записи — адміністратора і звичайного користувача. Це дозволяє одразу перевірити різницю між ролями: один створює документ і надсилає на погодження, інший — переглядає і приймає рішення. Таким чином відображається базовий цикл електронного документообігу.

Робота з документами включає повний набір базових операцій: створення, прикріплення файлу, перегляд, редагування, видалення і пошук. Тип файлів обмежено: дозволено pdf, doc, docx, txt, jpg і png. Інтерфейс побудований на Bootstrap, меню адаптується до ролі — звичайний користувач не бачить адміністративних розділів.

Попереднє тестування підтвердило: основні сценарії виконуються коректно. Користувач може увійти, створити документ, знайти його в списку, надіслати на погодження; адміністратор — змінити статус; журнал — зафіксувати всі дії. Системою вже можна скористатися для демонстрації.

Приклад роботи реалізованого програмного прототипу показано на рисунку 3.11.

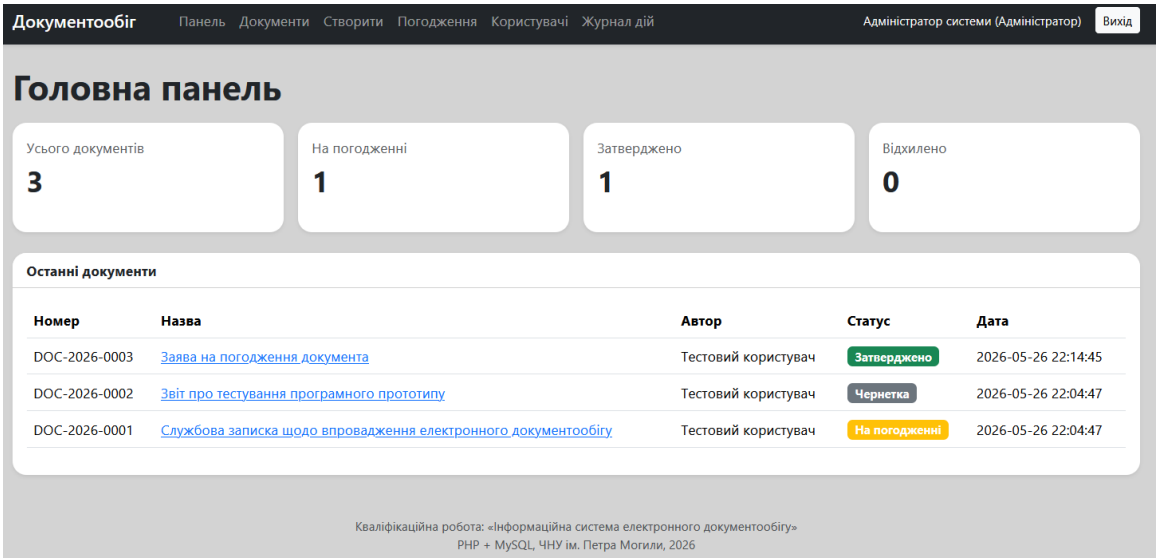

Рисунок 3.11 – Приклад роботи реалізованого програмного прототипу

Водночас є ще що доопрацювати: потрібна повноцінна перевірка розміру файлів, кращі повідомлення про помилки, резервне копіювання, можливість коментування документів, скріншоти для фінальної роботи і розширене тестування.

На рівні готовності ~75% вже сформовано теоретичну, проєктну та значну частину практичної основи кваліфікаційної роботи. Подальший етап — завершення тестування, підготовка скріншотів, оформлення додатків і доопрацювання спеціальної частини.

Стан реалізації основних модулів наведено в таблиці 3.3.

Таблиця 3.3 – Стан реалізації основних модулів

Модуль	Стан	Коментар
Авторизація	реалізовано	Підтримка входу, виходу, сесій і ролей.
База даних	реалізовано	Створено основні таблиці та зв’язки.
Документи	реалізовано частково	Працюють створення, перегляд, редагування, видалення і пошук.
Погодження	реалізовано	Адміністратор змінює статус документа.

Кінець таблиці 3.3

Модуль	Стан	Коментар
Журнал дій	реалізовано	Фіксуються основні події.
Тестування	виконується	Потрібно розширити набір тест-кейсів.
Додатки	заплановано	Потрібно додати повний SQL-код і фрагменти PHP.

Проміжний стан реалізації показує, що базова логіка системи вже сформована. Надалі основний акцент робиться на перевірці стабільності роботи, підготовці скріншотів інтерфейсу, розширенні тестових сценаріїв і більш детальному описі практичної частини.

3.6 Інструкція локального розгортання системи

Для перевірки і демонстрації системи використовується OSPanel. Цей інструмент дозволяє швидко підняти Apache, PHP, MySQL і phpMyAdmin без розміщення проєкту на зовнішньому сервері — що особливо зручно на етапі активної розробки.

Перший крок — скопіювати папку проєкту до каталогу доменів OSPanel (наприклад, D:\OSPanel\domains\localhost\document_flow_system).

Розміщення проєкту в локальному середовищі OSPanel показано на рисунку 3.12.

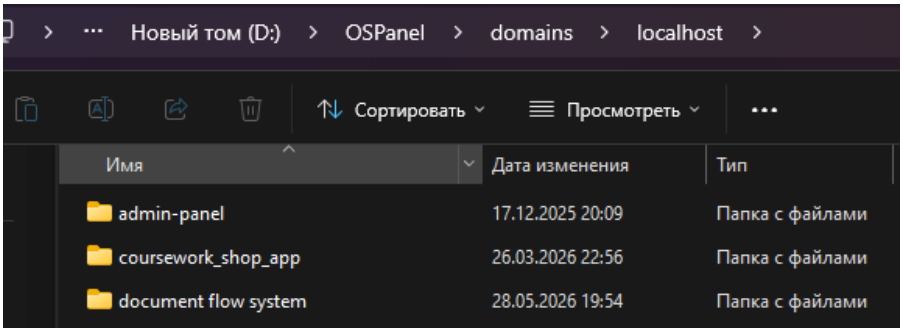

Рисунок 3.12 – Розміщення проєкту в локальному середовищі OSPanel

Туди потрапляють усі PHP-файли, каталоги assets і uploads, а також SQL-файл для розгортання бази.

Другий крок — імпорт бази даних. У phpMyAdmin створюється база document_flow, після чого виконується SQL-скрипт зі структурою таблиць. Якщо все пройшло без помилок, у базі з'являться таблиці users, documents, document_statuses, approvals і logs.

Імпорт бази даних через phpMyAdmin показано на рисунку 3.13.

Таблица	Действие	Строки	Тип	Сравнение	Размер	Фрагментировано
activity_logs		29	InnoDB	utf8mb4_unicode_ci	32.0 КиБ	-
approval_history		6	InnoDB	utf8mb4_unicode_ci	48.0 КиБ	-
documents		3	InnoDB	utf8mb4_unicode_ci	96.0 КиБ	-
document_files		1	InnoDB	utf8mb4_unicode_ci	32.0 КиБ	-
document_statuses		5	InnoDB	utf8mb4_unicode_ci	32.0 КиБ	-
document_types		7	InnoDB	utf8mb4_unicode_ci	16.0 КиБ	-
users		3	InnoDB	utf8mb4_unicode_ci	32.0 КиБ	-

Рисунок 3.13 – Імпорт бази даних через phpMyAdmin

Третій крок — перевірка config.php. У локальному середовищі зазвичай використовується localhost, користувач root і порожній пароль. Якщо налаштування OSPanel відрізняються — змінити параметри в одному файлі. Ім'я бази в config.php має точно збігатися з іменем, вказаним у phpMyAdmin.

Після цього система відкривається за адресою http://localhost/document_flow_system/. За правильних налаштувань одразу з'являється сторінка авторизації.

Запуск інформаційної системи у локальному середовищі показано на рисунку 3.14.

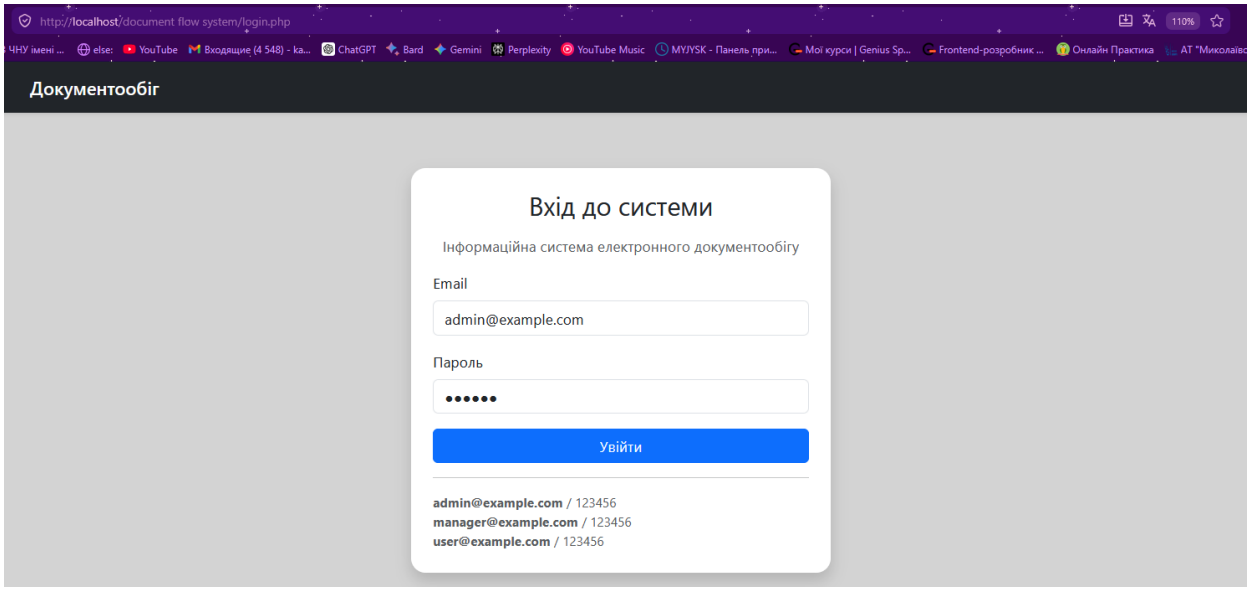

Рисунок 3.14 – Запуск інформаційної системи у локальному середовищі

Для тестування використовуються два тестові акаунти — адміністратора і звичайного користувача.

У майбутньому систему можна перенести на віддалений сервер. Для цього потрібно перемістити файли, створити базу на сервері, змінити параметри підключення і налаштувати права доступу до каталогу uploads. Додатково варто налаштувати HTTPS, резервне копіювання та обмеження доступу до службових файлів.

Послідовність локального запуску системи наведено в таблиці 3.4.

Таблиця 3.4 – Послідовність локального запуску системи

Крок	**Дія**	**Очікуваний результат**
1	Скопіювати файли проєкту до каталогу OSPanel/domains/localhost.	Проєкт доступний як локальний вебсайт.
2	Запустити Apache і MySQL в OSPanel.	Локальний сервер готовий до роботи.
3	Імпортувати database.sql у phpMyAdmin.	Створено базу document_flow і таблиці системи.

Кінець таблиці 3.4

Крок	Дія	Очікуваний результат
4	Перевірити параметри config.php.	PHP-скрипти можуть підключатися до MySQL.
5	Відкрити систему в браузері.	Відображається сторінка авторизації.
6	Увійти як користувач і як адміністратор.	Можна перевірити різні ролі системи.

Наведена послідовність запуску важлива для практичної перевірки результатів роботи. Вона дає можливість відтворити систему на локальному комп’ютері, імпортувати базу даних і продемонструвати функціонування основних модулів без додаткової серверної інфраструктури.

3.7 Попереднє функціональне тестування

Після реалізації основних модулів проведено попереднє функціональне тестування. Його мета — не навантажувальне дослідження, а перевірка базових сценаріїв: чи можна увійти до системи, створити документ, завантажити файл, знайти запис, надіслати на погодження, прийняти рішення й отримати запис у журналі.

Тестування авторизації показало: система коректно відрізняє правильні облікові дані від неправильних. При вірному вводі — відкривається головна панель, при помилці — з'являється повідомлення. Без сесії будь-яка захищена сторінка перенаправляє на вхід.

При тестуванні створення документа перевірялося: чи додається запис у базу, чи зберігається файл в uploads, чи присвоюється початковий статус. Окремо перевірявся сценарій без файлу — документ усе одно має створюватися, бо не кожен запис потребує вкладення.

Пошук тестувався за назвою та описом. Він відразу робить роботу зручнішою, особливо коли документів накопичилося вже кілька десятків. Без пошуку ручний перегляд довгої таблиці стає справжнім випробуванням.

Сценарій погодження перевірявся поетапно: користувач надсилає документ, він з'являється у розділі адміністратора, той затверджує або відхиляє — статус змінюється, дія фіксується в журналі. Вся ланцюжок відпрацьовує коректно.

Перевірка ролей підтвердила, що звичайний користувач не потрапляє на approvals.php, users.php і logs.php — навіть через пряме введення адреси в рядку браузера. Функція requireAdmin() відпрацьовує на сервері, незалежно від того, що видно у меню.

Тестування підтвердило, що основна логіка системи функціонує. Залишилися задачі для фінального етапу: розширення набору тест-кейсів, перевірка негативних сценаріїв, перевірка обмежень на файли, підготовка скріншотів

Попередні тест-кейси системи наведено в таблиці 3.5.

Таблиця 3.5 – Попередні тест-кейси системи

№	Сценарій	Очікуваний результат	Стан
1	Вхід адміністратора з правильними даними.	Відкривається головна панель адміністратора.	пройдено
2	Вхід користувача з неправильним паролем.	Показується повідомлення про помилку.	пройдено
3	Створення документа з файлом.	Запис додається до БД, файл зберігається у uploads.	пройдено
4	Створення документа без назви.	Система не повинна створювати некоректний документ.	пройдено частково
5	Пошук документа за частиною назви.	Відображаються відповідні документи.	пройдено
6	Надсилання документа на погодження.	Статус змінюється на «На погодженні».	пройдено

Кінець 3.5

№	Сценарій	Очікуваний результат	Стан
7	Затвердження документа адміністратором.	Статус змінюється на «Затверджено».	пройдено
8	Спроба користувача відкрити журнал дій.	Доступ заборонено або виконується перенаправлення.	пройдено

Початкові тестові сценарії підтверджують працездатність основних модулів. Вони охоплюють дії, які користувач виконує найчастіше, тому саме ці перевірки є першочерговими перед переходом до повнішого тестування системи.

Попередні результати реалізації показують, що система вже може виконувати роль демонстраційного прототипу. Вона не обмежується однією сторінкою або окремою формою, а має набір взаємопов’язаних модулів: авторизацію, головну панель, реєстр документів, створення, редагування, погодження, користувачів і журнал дій. Саме взаємозв’язок цих частин дозволяє розглядати розробку як інформаційну систему.

Під час роботи над прототипом особливу увагу було приділено тому, щоб користувач бачив зрозумілий результат кожної дії. Після створення документа він переходить до сторінки перегляду, після надсилання змінюється статус, після рішення адміністратора документ отримує новий стан, а відповідні операції потрапляють до журналу. Це робить систему логічною з точки зору звичайного користувача.

3.8 Оцінка поточних обмежень і план доопрацювання

Будь-який прототип на проміжному етапі має обмеження — і це нормально. Важливіше їх чесно визнати і скласти план усунення, ніж замовчувати. Поточна версія системи реалізує основні функції, але до промислового рішення їй ще далеко.

Перше обмеження — спрощена модель погодження. Один адміністратор приймає рішення щодо всіх документів. У реальних організаціях часто потрібен

багаторівневий маршрут: спочатку документ перевіряє виконавець, потім — керівник підрозділу, потім — директор. Реалізація такого маршруту — завдання наступного етапу.

Друге обмеження — відсутність електронного підпису. Для юридично значущого документообігу потрібна інтеграція КЕП або аналогічного механізму підтвердження авторства та цілісності. У рамках навчального прототипу це виправдано, але архітектура свідомо залишає цю можливість відкритою.

Третє — базова система файлів. Поки що файли зберігаються локально без перевірки розміру, контролю дублікатів і захисту від прямого доступу. Для промислового рівня це слід виправити разом із налаштуванням резервного копіювання.

Четверте — відсутність повідомлень. Користувач дізнається про зміну статусу документа лише якщо сам зайде й перевірить. Внутрішні сповіщення у панелі або email-повідомлення значно покращили б досвід роботи.

П'яте — простий журнал дій без фільтрів за датою, користувачем або типом події. Для реального контролю цього недостатньо. Додавання фільтрів — відносно нескладна задача, яка суттєво підвищить корисність журналу.

План доопрацювання охоплює: розширення тестування, покращення перевірки вхідних даних, додавання повідомлень, генерацію PDF-звітів, налаштування резервного копіювання, підготовку повного SQL-коду та PHP-фрагментів для додатків.

План подальшого доопрацювання системи наведено в таблиці 3.6.

Таблиця 3.6 – План подальшого доопрацювання системи

Напрям	**Що потрібно зробити**	**Очікуваний ефект**
Безпека файлів	Додати перевірку розміру, MIME-типу та заборону небезпечних розширень.	Зменшення ризику завантаження шкідливих файлів.
Погодження	Реалізувати багаторівневий маршрут погодження.	Наближення системи до реальних бізнес-процесів.

Кінець таблиці 3.6

Напрям	Що потрібно зробити	Очікуваний ефект
Повідомлення	Додати сповіщення про зміну статусу документа.	Користувач швидше отримує інформацію про рішення.
Журнал дій	Додати фільтри за користувачем, датою і типом дії.	Зручніший контроль активності.
Експорт	Додати генерацію PDF або CSV-звітів.	Спрощення звітності та демонстрації результатів.
Резервування	Налаштувати копіювання бази та файлів.	Зменшення ризику втрати інформації.

Заплановані доопрацювання не змінюють загальну архітектуру системи, а доповнюють її практичними деталями. Це дозволяє поступово перейти від навчального прототипу до більш завершеного програмного продукту, придатного для демонстрації та подальшого розвитку.

4 ПОПЕРЕДНЄ ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОЗРОБКИ

4.1 Мета, об’єкти та підхід до тестування

Тестування інформаційної системи електронного документообігу — це не формальність, а обов'язковий етап, навіть коли йдеться про відносно невеликий вебзастосунок. Система складається з кількох взаємопов'язаних компонентів: інтерфейс, серверна логіка, база даних, авторизація, файлова підсистема і контроль доступу. І це якраз той випадок, коли ціле не міцніше за найслабшу ланку — помилка в будь-якому з компонентів здатна порушити роботу всього процесу.

На проміжному етапі кваліфікаційної роботи мета тестування — перевірити базові сценарії: чи може користувач увійти до системи, створити документ, знайти його, надіслати на погодження; чи може адміністратор переглянути та змінити статус; чи з'являються записи в журналі після кожної дії.

Для першої версії прототипу достатньо ручного тестування. Воно дозволяє не лише перевірити технічний результат, а й оцінити інтерфейс «очима користувача» — зрозуміти, де є незручності, які потребують виправлення ще до фінального оформлення.

Об'єктами тестування є: сторінка авторизації, головна панель, список документів, форма створення, сторінка перегляду, форма редагування, модуль погодження, сторінка користувачів і журнал дій. Окремо перевіряється логіка ролей — вона відповідає за безпечний доступ до адміністративних функцій.

Тестові сценарії поділяються на позитивні (коректні вхідні дані, очікувана поведінка) та негативні (неправильний пароль, порожні поля, заборонені URL, неприпустимі розширення файлів). Обидва типи важливі: перші підтверджують правильну роботу, другі — відмовостійкість системи.

Локальне розгортання також показало, що проєкт не прив’язаний до складної інфраструктури. Його можна перенести на інший комп’ютер, якщо скопіювати папку із файлами, імпортувати базу даних і правильно вказати параметри підключення. Це важливо для демонстрації кваліфікаційної роботи, адже система

має запускатися без зайвих залежностей і пояснюватися в межах звичайного локального середовища.

Під час розгортання окремо перевіряється доступність папки uploads, оскільки саме туди потрапляють прикріплені файли. Якщо права доступу або шлях налаштовано неправильно, система може створити документ, але не збереже вкладення. Тому інструкція локального запуску охоплює не лише імпорт бази, а й перевірку структури папок та коректності адреси сайту в браузері.

4.2 Перевірка авторизації та ролей користувачів

Авторизація — перший і найважливіший бар'єр доступу. Якщо вона не спрацьовує правильно, решта захисту втрачає сенс. Тому тестування розпочалося саме з неї: вхід адміністратора, вхід звичайного користувача, неправильний пароль, неіснуючий email.

Тестування підтвердило: після успішного входу в сесію записуються id, ім'я та роль.

Успішна авторизація адміністратора в системі показано на рисунку 4.1.

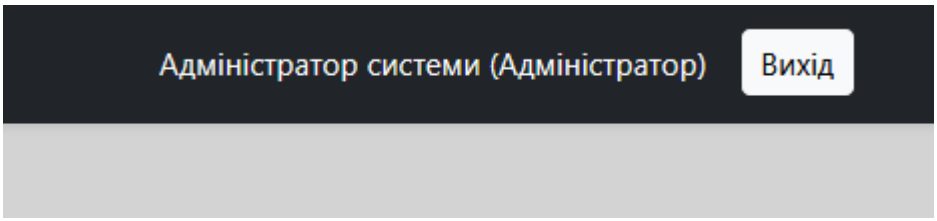

Рисунок 4.1 – Успішна авторизація адміністратора в системі

Ці дані далі управляють і відображенням меню, і серверними перевірками.

Відображення функцій адміністратора після авторизації показано на рисунку 4.2.

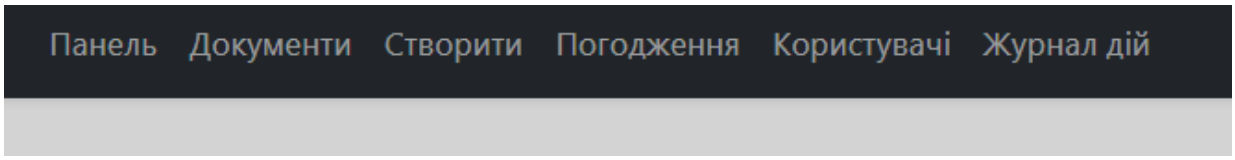

Рисунок 4.2 – Відображення функцій адміністратора після авторизації

При виході сесія знищується, і повторний доступ до захищених сторінок без нового входу стає неможливим.

Особливу увагу приділено перевірці ролей. Звичайний користувач не повинен потрапляти на approvals.php, users.php або logs.php — навіть якщо вручну введе адресу в рядку браузера.

Обмеження функцій для звичайного користувача показано на рисунку 4.3.

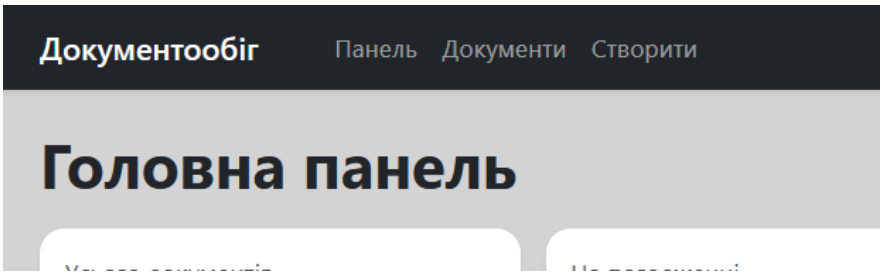

Рисунок 4.3 – Обмеження функцій для звичайного користувача

Функція requireAdmin() на сервері відпрацьовує незалежно від інтерфейсу і коректно перенаправляє або блокує доступ.

Результати тестування авторизації показали, що базова логіка доступу працює коректно. Водночас у фінальній версії системи варто доопрацювати кілька моментів: додати перевірку мінімальної довжини пароля, реалізувати тимчасове блокування облікового запису після кількох невдалих спроб входу, а також вивести користувачу явне підтвердження того, що він успішно вийшов із системи.

Тестування авторизації наведено в таблиці 4.1.

Таблиця 4.1 – Тестування авторизації

№	Тестова дія	Очікуваний результат	Результат
1	Вхід адміністратора з правильним email і паролем.	Відкриття головної панелі адміністратора.	виконано
2	Вхід користувача з правильним email і паролем.	Відкриття головної панелі користувача.	виконано
3	Введення неправильного пароля.	Повідомлення про помилку.	виконано
4	Введення неіснуючого email.	Повідомлення, що користувача не знайдено.	виконано

Кінець таблиці 4.1

№	Тестова дія	Очікуваний результат	Результат
5	Спроба відкрити сторінку без сесії.	Перенаправлення на сторінку входу.	виконано
6	Спроба user відкрити logs.php.	Доступ заборонено або перенаправлення.	виконано

Результати перевірки авторизації свідчать, що доступ до системи контролюється через облікові записи користувачів. Це є базовою умовою для будь-якої системи документообігу, оскільки всі подальші операції мають виконуватися від імені конкретного користувача.

4.3 Перевірка роботи з документами

Модуль документів — центральний, тому і тестування найширше. Перевірялися: створення з файлом і без нього, редагування, перегляд, видалення, пошук за назвою та описом. Також — збереження зв'язку документа з автором і статусом після кожної операції.

Відображення статусів документів у реєстрі показано на рисунку 4.4.

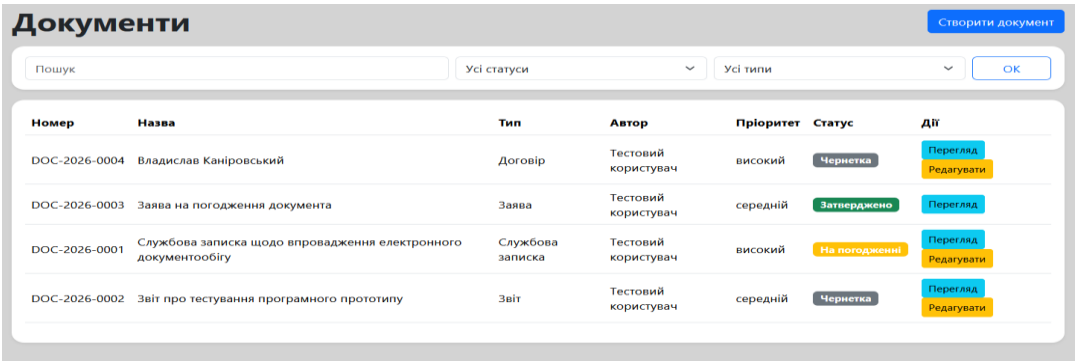

Рисунок 4.4 – Відображення статусів документів у реєстрі

При створенні система перевіряє наявність назви.

Заповнення форми створення документа показано на рисунку 4.5.

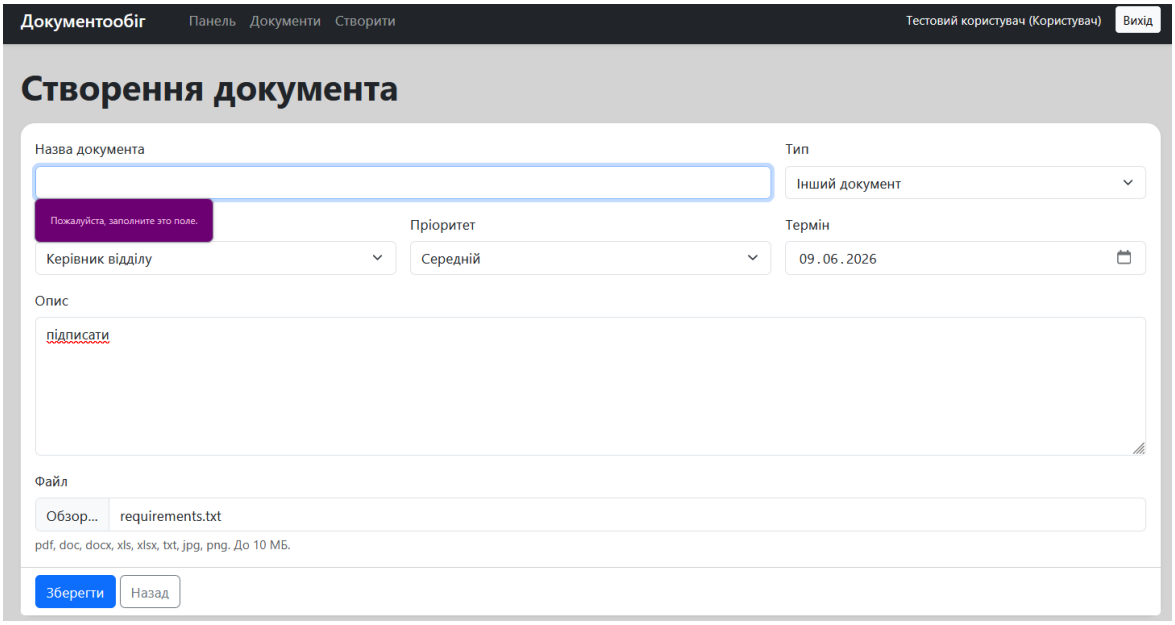

Рисунок 4.5 – Заповнення форми створення документа

Опис може бути коротким або розгорнутим, файл — необов'язковим. Якщо файл прикріплено, він зберігається в uploads, у базі фіксується шлях.

Результат створення нового електронного документа показано на рисунку 4.6.

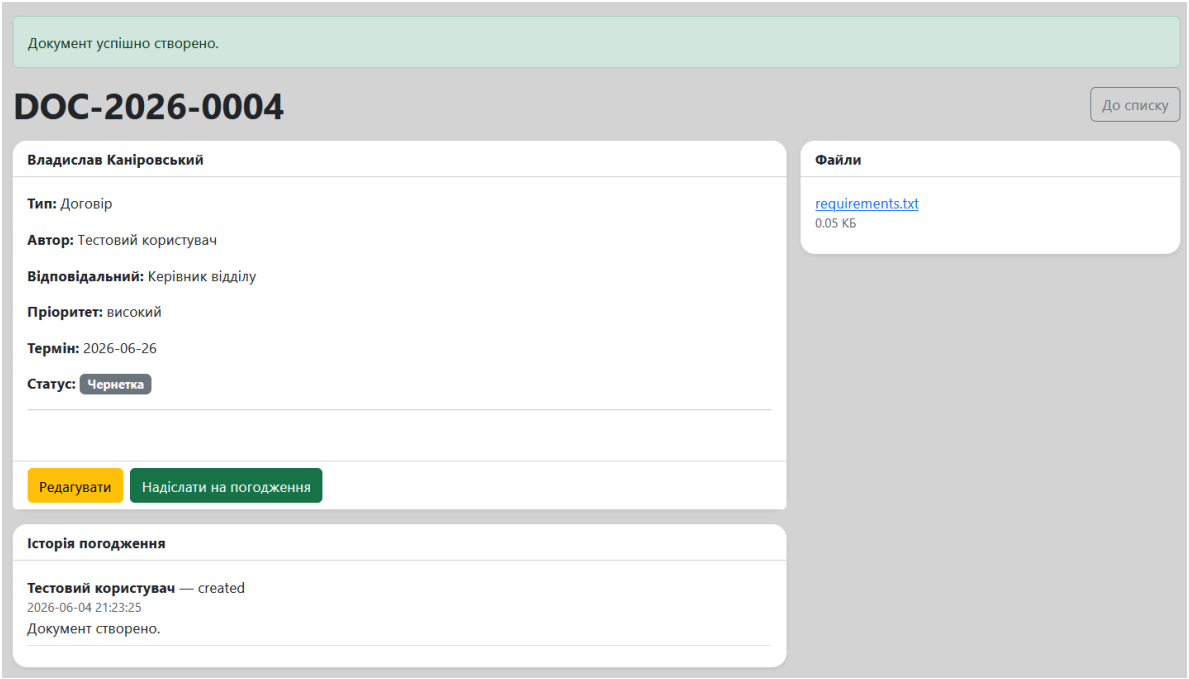

Рисунок 4.6 – Результат створення нового електронного документа

Зберігання файлів поза базою даних — свідомий вибір, що спрощує резервування і розгортання.

Сторінка перегляду показує назву, автора, статус, дату, опис і посилання на файл, якщо він є.

Перегляд електронного документа в системі показано на рисунку 4.7.

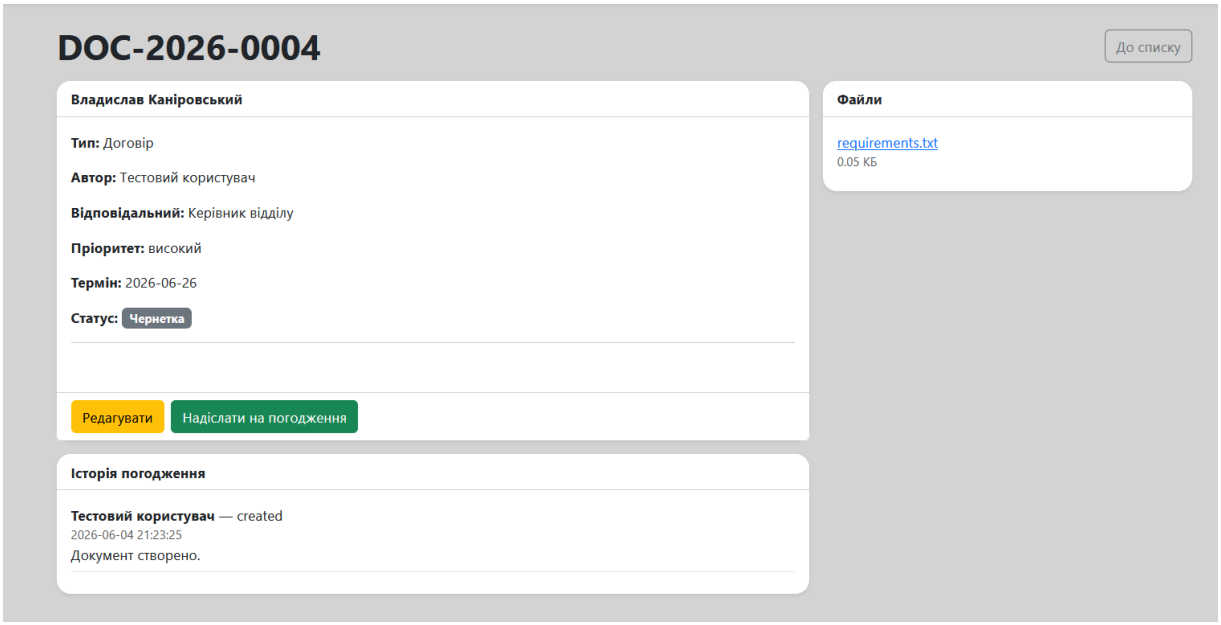

Рисунок 4.7 – Перегляд електронного документа в системі

Для користувача важливо, щоб уся потрібна інформація була доступна без додаткових кроків — без заглиблення в базу або файлову систему.

Пошук тестувався за частиною назви і ключовим словом з опису.

Перевірка пошуку документа в реєстрі показано на рисунку 4.8.

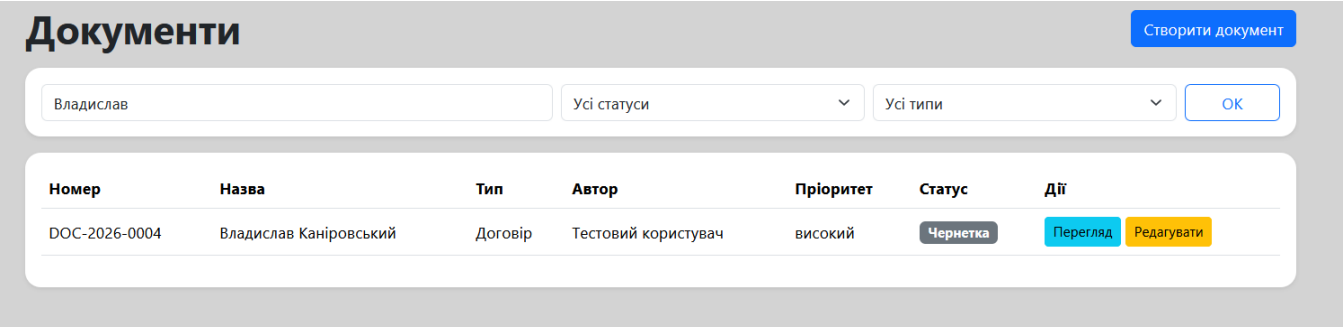

Рисунок 4.8 – Перевірка пошуку документа в реєстрі

Якщо результат знайдено — відображається таблиця записів, якщо ні — таблиця порожня. У фінальній версії бажано додати повідомлення «нічого не знайдено» для кращої зрозумілості інтерфейсу.

Редагування перевіряло, чи зберігаються зміни і чи не руйнується прив'язка до автора. Видалення — критична операція, яка потребує підтвердження і обов'язково фіксується в журналі дій.

Тестування модуля документів наведено в таблиці 4.2.

Таблиця 4.2 – Тестування модуля документів

№	Сценарій	Очікуваний результат	Стан
1	Створення документа з назвою, описом і файлом.	Документ створено, файл збережено.	виконано
2	Створення документа без файлу.	Документ створено без вкладення.	виконано
3	Створення документа без назви.	Система не дозволяє створення.	виконано
4	Редагування назви та опису.	Зміни збережено в базі даних.	виконано
5	Пошук за частиною назви.	Відображено відповідні записи.	виконано
6	Пошук за неіснуючим словом.	Результати не відображаються.	виконано
7	Видалення документа.	Документ видалено, подію записано в журнал.	виконано

Перевірка модуля документів показала, що система виконує основні операції з документами та коректно зберігає їхні атрибути. Це підтверджує працездатність центральної частини програмного прототипу.

4.4 Перевірка погодження документів і журналу дій

Тестування погодження зосереджено на правильному переміщенні документа між статусами. Вихідна точка — статус «Створено». Після натискання кнопки надсилання документ переходить до «На погодженні».

Надсилання документа на погодження показано на рисунку 4.9.

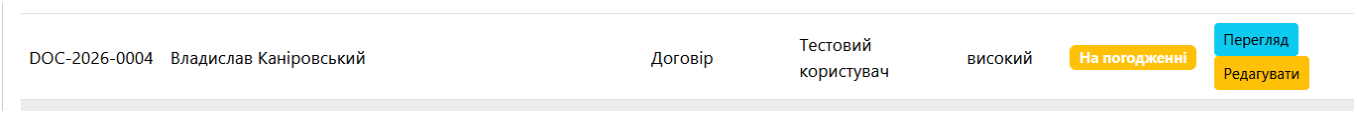

Рисунок 4.9 – Надсилання документа на погодження

Потім адміністратор приймає рішення: «Затверджено» або «Відхилено».

Зміна статусу документа після затвердження показано на рисунку 4.10.

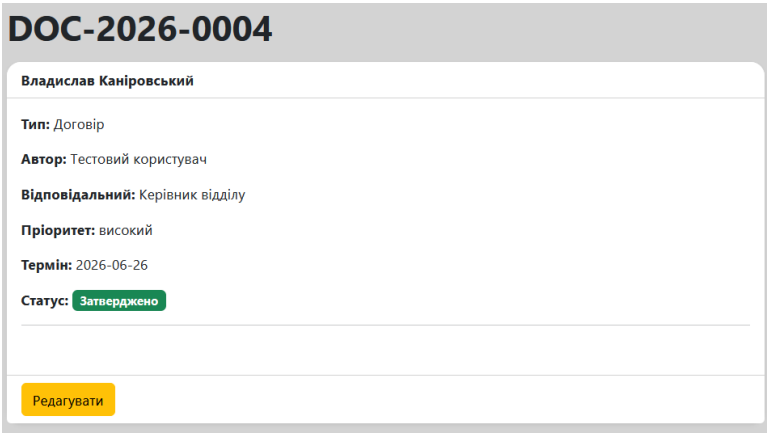

Рисунок 4.10 – Зміна статусу документа після затвердження

Перевірялася не тільки зміна статусу, а й видимість документа у відповідних розділах. Документ зі статусом «На погодженні» повинен з'явитися на approvals.php.

Відображення документа у списку погодження показано на рисунку 4.11.

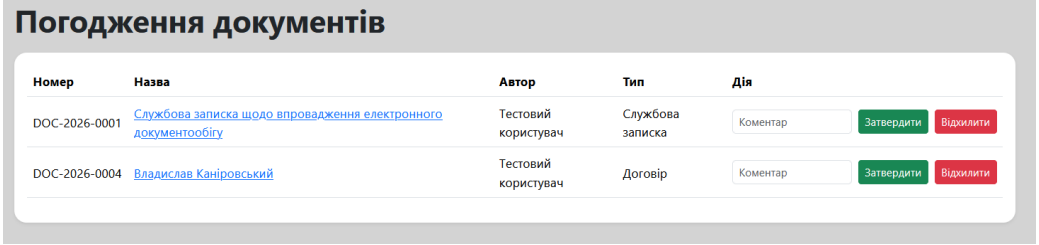

Рисунок 4.11 – Відображення документа у списку погодження

Після прийняття рішення він зникає зі списку очікуваних — і це теж перевірялося.

Журнал дій тестувався паралельно з основними операціями. Після створення, надсилання і рішення адміністратора в таблиці logs мали з'явитися відповідні записи — і вони з'являлися.

Фіксація операцій користувачів у журналі дій показано на рисунку 4.12.

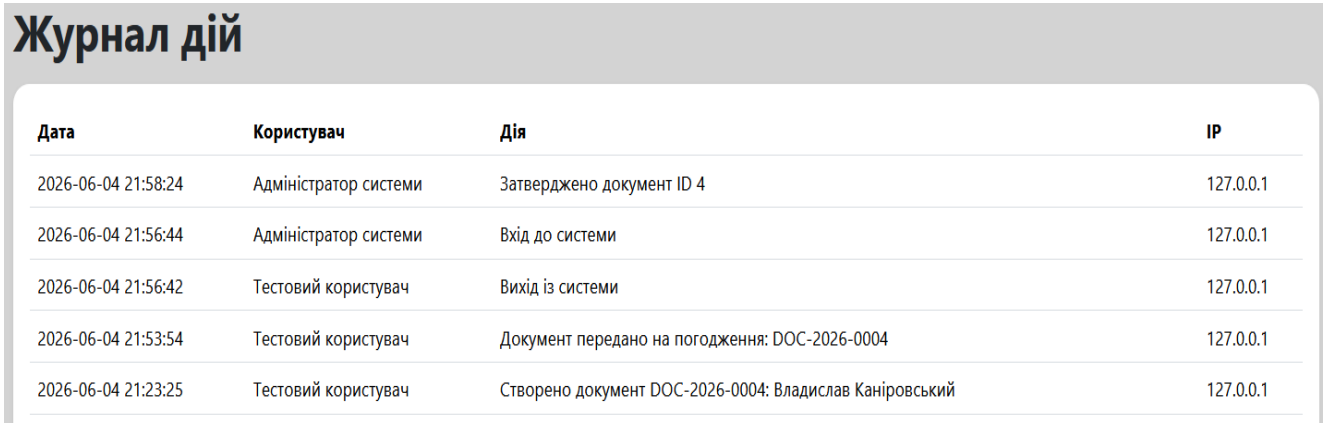

Рисунок 4.12 – Фіксація операцій користувачів у журналі дій

Це підтверджує, що система зберігає не лише поточний стан, а й повну хронологію роботи.

Поточна структура журналу: користувач, дія, час. Для демонстрації цього вистачає. Для реального використання потрібні посилання на документ, тип дії і детальна інформація про зміну статусу. Ці вдосконалення заплановані на наступний етап.

За результатами перевірки логіка погодження відповідає поставленим вимогам. Для фінальної версії залишається реалізувати коментар адміністратора до відхиленого документа і зробити його видимим для автора.

Тестування погодження та журналу дій наведено в таблиці 4.3.

Таблиця 4.3 – Тестування погодження та журналу дій

№	Дія	Очікуваний результат	Стан
1	Надсилання документа на погодження.	Статус змінюється на «На погодженні».	виконано
2	Відображення документа в approvals.php.	Адміністратор бачить документ.	виконано
3	Затвердження документа.	Статус змінюється на «Затверджено».	виконано
4	Відхилення документа.	Статус змінюється на «Відхилено».	виконано
5	Перевірка журналу після створення.	З’являється запис про створення.	виконано
6	Перевірка журналу після погодження.	З’являється запис про рішення адміністратора.	виконано

Тестування погодження та журналу дій підтвердило, що система не лише змінює статус документа, а й зберігає історію важливих операцій. Саме це дозволяє розглядати розробку як прототип електронного документообігу, а не просто як каталог файлів.

Під час тестування авторизації окремо оцінювалося те, як система поводиться з точки зору звичайного користувача. Тут важливі дві речі: після успішного входу він має одразу опинятися на зрозумілій і очікуваній сторінці, а після помилки — бачити повідомлення, яке пояснює ситуацію, але не розкриває зайвих технічних деталей. Це не просто питання зручності — це баланс між юзабіліті і безпекою: користувач розуміє, що сталося, але не отримує інформації, яка могла б бути корисною зловмиснику.

Перевірка ролей показала, що адміністративні можливості не повинні змішуватися зі звичайною роботою користувача. Якщо працівник лише створює документи, йому не потрібні сторінки керування користувачами або перегляд

журналу дій. Завдяки такому розмежуванню інтерфейс стає чистішим, а ризик випадкових дій у системі зменшується.

Тестування роботи з документами виконувалося не лише формально, а за логікою реального використання: створити запис, перевірити його у списку, відкрити, змінити, знайти через пошук і проконтролювати статус. Така послідовність дозволяє переконатися, що окремі функції працюють не ізольовано, а підтримують повний робочий сценарій.

Перевірка погодження підтвердила головну ідею системи: документ має керований життєвий цикл. Він створюється, переходить на погодження, отримує рішення адміністратора і зберігає історію виконаних дій. Саме ця логіка відрізняє систему електронного документообігу від простого каталогу файлів і показує практичну цінність розробленого прототипу.

4.5 Висновки до розділу 4

У четвертому розділі описано попереднє тестування прототипу. Перевірено авторизацію, ролі, створення і редагування документів, пошук, надсилання на погодження, зміну статусів і журнал дій.

Результати підтвердили, що основні функціональні вимоги виконано: система вже готова до демонстрації як робочий прототип.

Отримані результати підтверджують, що обрана архітектура і технологічний стек підходять для поставленої задачі. Подальше доопрацювання не потребує переосмислення концепції — воно спрямоване на поглиблення, стабілізацію і фінальне оформлення.

ВИСНОВКИ

У рамках підготовки кваліфікаційної роботи виконано значний обсяг теоретичної, проєктної та практичної роботи за темою «Інформаційна система електронного документообігу».

У першому розділі проаналізовано сутність і роль електронного документообігу, його переваги перед паперовою моделлю, нормативно-правові засади, організаційні вимоги, системи-аналоги та технологічний стек для реалізації. Зроблено висновок, що для невеликої організації найдоцільніша проста, контрольована і розширювана система, яка підтримує базовий цикл документа.

У другому розділі сформульовано постановку задачі, функціональні та нефункціональні вимоги, розроблено архітектуру системи, структуру модулів, інформаційну модель бази даних і модель ролей. Це забезпечило перехід від загального аналізу до конкретних проєктних рішень.

У третьому розділі описано програмний прототип системи: структуру PHP-проєкту, реалізацію бази даних, підключення до MySQL, авторизацію, роботу з документами, механізм погодження і журнал дій. На цьому етапі система вже виконує основні функції, достатні для демонстрації електронного документообігу.

Подальша робота спрямована на завершення тестування, підготовку скріншотів, розширення опису результатів, оформлення додатків із SQL-структурою та PHP-кодом, а також доопрацювання спеціальної частини до повного обсягу. Після цього роботу буде приведено до підсумкового формату відповідно до методичних рекомендацій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Про електронні документи та електронний документообіг : Закон України від 22.05.2003 № 851-IV // База даних «Законодавство України» / Верховна Рада України. URL: https://zakon.rada.gov.ua/go/851-15 (дата звернення: 21.04.2026).

2. Про електронну ідентифікацію та електронні довірчі послуги : Закон України від 05.10.2017 № 2155-VIII // База даних «Законодавство України» / Верховна Рада України. URL: https://zakon.rada.gov.ua/go/2155-19 (дата звернення: 22.04.2026).

3. Методичні рекомендації до виконання кваліфікаційних робіт здобувачами першого (бакалаврського) рівня вищої освіти за спеціальністю 122 Комп'ютерні науки. Миколаїв : ЧНУ ім. Петра Могили, 2026.

4. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлення. Київ, 2016.

5. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Київ, 2016.

6. M.E.Doc. Електронний документообіг. URL: https://medoc.ua/elektronniy_dokumentoobig (дата звернення: 27.04.2026).

7. Вчасно. Сервіс електронного документообігу. URL: https://vchasno.ua/ (дата звернення: 28.04.2026).

8. DocuWare. Document Management Software & Workflow Automation. URL: https://start.docuware.com/ (дата звернення: 28.04.2026).

9. Microsoft. SharePoint document management. URL: https://www.microsoft.com/en-us/microsoft-365/sharepoint/collaboration (дата звернення: 29.04.2026).

10. PHP Manual. password_hash. URL: https://www.php.net/manual/en/function.password-hash.php (дата звернення: 24.05.2026).

11. PHP Manual. password_verify. URL: https://www.php.net/manual/en/function.password-verify.php (дата звернення: 29.04.2026).

12. PHP Manual. mysqli::prepare. URL: https://www.php.net/manual/en/mysqli.prepare.php (дата звернення: 30.04.2026).

13. MySQL 8.4 Reference Manual. FOREIGN KEY Constraints. URL: https://dev.mysql.com/doc/refman/8.4/en/create-table-foreign-keys.html (дата звернення: 01.05.2026).

14. MySQL 8.4 Reference Manual. CREATE TABLE Statement. URL: https://dev.mysql.com/doc/refman/8.4/en/create-table.html (дата звернення: 01.05.2026).

15. Bootstrap Documentation. Introduction. URL: https://getbootstrap.com/docs/5.3/getting-started/introduction/ (дата звернення: 02.05.2026).

16. OWASP Foundation. Web Security Testing Guide. URL: https://owasp.org/www-project-web-security-testing-guide/ (дата звернення: 02.05.2026).

17. OWASP Foundation. SQL Injection Prevention Cheat Sheet. URL: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html (дата звернення: 02.05.2026).

18. OWASP Foundation. Authentication Cheat Sheet. URL: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html (дата звернення: 03.05.2026).

19. Sommerville I. Software Engineering. 10th ed. Pearson, 2016. 816 p.

20. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner’s Approach. 9th ed. McGraw-Hill Education, 2020. 880 p.

21. Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley, 2002. 560 p.

22. Elmasri R., Navathe S. B. Fundamentals of Database Systems. 7th ed. Pearson, 2016. 1280 p.

23. Date C. J. An Introduction to Database Systems. 8th ed. Pearson, 2003. 1024 p.

24. MySQL 8.4 Reference Manual. SELECT Statement. URL: https://dev.mysql.com/doc/refman/8.4/en/select.html (дата звернення: 03.05.2026).

25. PHP Manual. Sessions. URL: https://www.php.net/manual/en/book.session.php (дата звернення: 03.05.2026).

26. Bootstrap Documentation. Forms. URL: https://getbootstrap.com/docs/5.3/forms/overview/ (дата звернення: 04.05.2026).

27. MDN Web Docs. HTML: HyperText Markup Language. URL: https://developer.mozilla.org/en-US/docs/Web/HTML (дата звернення: 04.05.2026).

28. MDN Web Docs. CSS: Cascading Style Sheets. URL: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення: 05.05.2026).

29. OSPanel. Documentation. URL: https://ospanel.io/docs/ (дата звернення: 24.05.2026).

30. OpenAI. ChatGPT. Інструмент генеративного штучного інтелекту. URL: https://chatgpt.com/ (дата звернення: 05.06.2026).

ДОДАТОК А

Структура бази даних інформаційної системи

database.sql – SQL-скрипт створення таблиць і початкових даних

```
CREATE DATABASE IF NOT EXISTS document_flow_full CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci;
USE document_flow_full;

SET FOREIGN_KEY_CHECKS = 0;
DROP TABLE IF EXISTS approval_history;
DROP TABLE IF EXISTS document_files;
DROP TABLE IF EXISTS documents;
DROP TABLE IF EXISTS document_statuses;
DROP TABLE IF EXISTS document_types;
DROP TABLE IF EXISTS activity_logs;
DROP TABLE IF EXISTS users;
SET FOREIGN_KEY_CHECKS = 1;

CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    full_name VARCHAR(180) NOT NULL,
    email VARCHAR(180) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    role ENUM('admin', 'manager', 'user') NOT NULL DEFAULT 'user',
    position VARCHAR(120) NULL,
    department VARCHAR(120) NULL,
    is_active TINYINT(1) NOT NULL DEFAULT 1,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE document_statuses (
    id INT AUTO_INCREMENT PRIMARY KEY,
    code VARCHAR(50) NOT NULL UNIQUE,
    name VARCHAR(80) NOT NULL
);

INSERT INTO document_statuses (code, name) VALUES
('draft', 'Чернетка'),
('review', 'На погодженні'),
('approved', 'Затверджено'),
('rejected', 'Відхилено'),
('archived', 'Архівовано');

CREATE TABLE document_types (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(120) NOT NULL
);

INSERT INTO document_types (name) VALUES
('Службова записка'),
('Заява'),
('Наказ'),
('Договір'),
('Акт'),
('Звіт'),
('Інший документ');

CREATE TABLE documents (
    id INT AUTO_INCREMENT PRIMARY KEY,
    document_number VARCHAR(60) NOT NULL UNIQUE,
    title VARCHAR(255) NOT NULL,
    description TEXT NULL,
    document_type_id INT NOT NULL,
```

```
    author_id INT NOT NULL,
    responsible_id INT NULL,
    status_id INT NOT NULL DEFAULT 1,
    priority ENUM('низький', 'середній', 'високий') NOT NULL DEFAULT 'середній',
    due_date DATE NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP NULL DEFAULT NULL,
    FOREIGN KEY (document_type_id) REFERENCES document_types(id),
    FOREIGN KEY (author_id) REFERENCES users(id) ON DELETE CASCADE,
    FOREIGN KEY (responsible_id) REFERENCES users(id) ON DELETE SET NULL,
    FOREIGN KEY (status_id) REFERENCES document_statuses(id)
);

CREATE TABLE document_files (
    id INT AUTO_INCREMENT PRIMARY KEY,
    document_id INT NOT NULL,
    original_name VARCHAR(255) NOT NULL,
    stored_name VARCHAR(255) NOT NULL,
    file_path VARCHAR(255) NOT NULL,
    file_size INT NULL,
    uploaded_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (document_id) REFERENCES documents(id) ON DELETE CASCADE
);

CREATE TABLE approval_history (
    id INT AUTO_INCREMENT PRIMARY KEY,
    document_id INT NOT NULL,
    user_id INT NOT NULL,
    action ENUM('created','sent_to_review','approved','rejected','archived','edited') NOT
NULL,
    comment TEXT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (document_id) REFERENCES documents(id) ON DELETE CASCADE,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);

CREATE TABLE activity_logs (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NULL,
    action VARCHAR(255) NOT NULL,
    ip_address VARCHAR(45) NULL,
    user_agent VARCHAR(255) NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE SET NULL
);

-- Усі тестові паролі: 123456
INSERT INTO users (full_name, email, password, role, position, department) VALUES
('Адміністратор системи', 'admin@example.com',
'$2y$12$ipbvpH3i.LZrkbsx1yz/2O5FIg.k953m0mXzJCK3TjhikrzYUoEey', 'admin', 'Адміністратор',
'IT-відділ'),
('Керівник відділу', 'manager@example.com',
'$2y$12$ipbvpH3i.LZrkbsx1yz/2O5FIg.k953m0mXzJCK3TjhikrzYUoEey', 'manager', 'Керівник',
'Канцелярія'),
('Тестовий користувач', 'user@example.com',
'$2y$12$ipbvpH3i.LZrkbsx1yz/2O5FIg.k953m0mXzJCK3TjhikrzYUoEey', 'user', 'Спеціаліст',
'Навчальний відділ');

INSERT INTO documents
(document_number, title, description, document_type_id, author_id, responsible_id,
status_id, priority, due_date)
VALUES
('DOC-2026-0001', 'Службова записка щодо впровадження електронного документообігу',
'Документ створено для демонстрації роботи інформаційної системи електронного
документообігу. У ньому описано потребу в автоматизації обліку, погодження та зберігання
документів.',
1, 3, 2, 2, 'високий', '2026-06-20'),
('DOC-2026-0002', 'Звіт про тестування програмного прототипу',
```

```
'Тестовий документ, який використовується для перевірки статусів, пошуку, перегляду та
журналювання дій користувачів.',
6, 3, 2, 1, 'середній', '2026-06-25');

INSERT INTO approval_history (document_id, user_id, action, comment) VALUES
(1, 3, 'created', 'Документ створено користувачем.'),
(1, 3, 'sent_to_review', 'Документ надіслано на погодження.'),
(2, 3, 'created', 'Створено тестовий звіт.');

INSERT INTO activity_logs (user_id, action, ip_address) VALUES
(3, 'Створено демонстраційний документ DOC-2026-0001', '127.0.0.1'),
(3, 'Створено демонстраційний документ DOC-2026-0002', '127.0.0.1');
```

ДОДАТОК Б

Службові файли підключення, налаштувань і функцій

app/config.php – налаштування системи та підключення до MySQL

```
<?php
declare(strict_types=1);
const DB_HOST = 'localhost';
const DB_USER = 'root';
const DB_PASS = '';
const DB_NAME = 'document_flow_full';
const APP_NAME = 'ІС електронного документообігу';
const UPLOAD_DIR = __DIR__ . '/../uploads/';
const UPLOAD_WEB_PATH = 'uploads/';
const MAX_FILE_SIZE = 10 * 1024 * 1024;
$mysqli = new mysqli(DB_HOST, DB_USER, DB_PASS, DB_NAME);
if ($mysqli->connect_error) { die('Помилка підключення до БД: ' . $mysqli-
>connect_error); }
$mysqli->set_charset('utf8mb4');
?>
```

app/bootstrap.php – підключення конфігурації та функцій

```
<?php
if (session_status() === PHP_SESSION_NONE) session_start();
require_once __DIR__ . '/config.php';
require_once __DIR__ . '/functions.php';
?>
```

app/functions.php – службові функції системи

```
<?php

declare(strict_types=1);

function e(?string $value): string

{

    return htmlspecialchars($value ?? '', ENT_QUOTES, 'UTF-8');

}

function redirect(string $url): void

{

    header("Location: {$url}");

    exit;

}

function currentUserId(): ?int

{

    return isset($_SESSION['user']['id'])

        ? (int) $_SESSION['user']['id']

        : null;

}

function currentUserRole(): ?string
```

```
{
    return $_SESSION['user']['role'] ?? null;
}

function isLoggedIn(): bool
{
    return isset($_SESSION['user']);
}
function isAdmin(): bool
{
    return currentUserRole() === 'admin';
}
function isManager(): bool
{
    return currentUserRole() === 'manager';
}
function canApprove(): bool
{
    return in_array(currentUserRole(), ['admin', 'manager'], true);
}
function requireLogin(): void
{
    if (!isLoggedIn()) {
        redirect('login.php');
    }
}
function requireAdmin(): void
{
    requireLogin();
    if (!isAdmin()) {
        $_SESSION['flash_error'] = 'Недостатньо прав доступу.';
        redirect('dashboard.php');
    }
}
function requireApprover(): void
{
    requireLogin();
    if (!canApprove()) {
```

```
        $_SESSION['flash_error'] = 'Погодження доступне лише керівнику або
адміністратору.';
        redirect('dashboard.php');
    }
}
function flash(string $type): ?string
{
    $key = 'flash_' . $type;
    if (!empty($_SESSION[$key])) {
        $message = $_SESSION[$key];
        unset($_SESSION[$key]);
        return $message;
    }
    return null;
}
function csrfToken(): string
{
    if (empty($_SESSION['csrf_token'])) {
        $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
    }
    return $_SESSION['csrf_token'];
}

function checkCsrf(): void
{
    $token = $_POST['csrf_token'] ?? '';
    if (
        !$token ||
        !hash_equals($_SESSION['csrf_token'] ?? '', $token)
    ) {
        die('Помилка CSRF-перевірки.');
    }
}
function logAction(mysqli $db, ?int $userId, string $action): void
{
    $ip = $_SERVER['REMOTE_ADDR'] ?? 'unknown';
    $agent = substr($_SERVER['HTTP_USER_AGENT'] ?? 'unknown', 0, 250);
    $stmt = $db->prepare(
        'INSERT INTO activity_logs
```

```
            (user_id, action, ip_address, user_agent)
         VALUES
            (?, ?, ?, ?)'
    );
    $stmt->bind_param(
        'isss',
        $userId,
        $action,
        $ip,
        $agent
    );
    $stmt->execute();
}
function addApprovalHistory(
    mysqli $db,
    int $documentId,
    int $userId,
    string $action,
    ?string $comment = null
): void {
    $stmt = $db->prepare(
        'INSERT INTO approval_history
            (document_id, user_id, action, comment)
         VALUES
            (?, ?, ?, ?)'
    );
    $stmt->bind_param(
        'iiss',
        $documentId,
        $userId,
        $action,
        $comment
    );
    $stmt->execute();
}
function roleName(string $role): string
{
    switch ($role) {
        case 'admin':
```

```
            return 'Адміністратор';

        case 'manager':
            return 'Керівник';
        default:
            return 'Користувач';
    }
}
function statusBadgeClass(string $code): string
{
    switch ($code) {
        case 'draft':
            return 'secondary';

        case 'review':
            return 'warning';

        case 'approved':
            return 'success';

        case 'rejected':
            return 'danger';

        case 'archived':
            return 'dark';

        default:
            return 'secondary';
    }
}
function generateDocumentNumber(mysqli $db): string
{
    $year = date('Y');
    $prefix = "DOC-{$year}-";
    $like = $prefix . '%';
    $stmt = $db->prepare(
        'SELECT COUNT(*) AS total
         FROM documents
         WHERE document_number LIKE ?'
```

```
    );
    $stmt->bind_param('s', $like);
    $stmt->execute();
    $count = (int) (
        $stmt->get_result()->fetch_assoc()['total'] ?? 0
    ) + 1;
    return $prefix . str_pad(
        (string) $count,
        4,
        '0',
        STR_PAD_LEFT
    );
}
function allowedFileExtension(string $filename): bool
{
    $allowed = [
        'pdf',
        'doc',
        'docx',
        'xls',
        'xlsx',
        'txt',
        'jpg',
        'jpeg',
        'png',
    ];
    $extension = strtolower(
        pathinfo($filename, PATHINFO_EXTENSION)
    );
    return in_array($extension, $allowed, true);
}
function userCanEditDocument(array $doc): bool
{
    if (isAdmin()) {
        return true;
    }
    return (int) $doc['author_id'] === currentUserId()
        && ($doc['status_code'] ?? '') !== 'approved';
}
```

ДОДАТОК В

Файли інтерфейсу та авторизації користувача

app/includes/header.php – верхня частина інтерфейсу та меню

```
<?php require_once __DIR__ . '/../bootstrap.php'; ?>
<!DOCTYPE html>
<html lang="uk">
<head>
<meta charset="UTF-8">
<title><?= APP_NAME ?></title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
<link href="assets/css/style.css" rel="stylesheet">
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-dark bg-primary shadow-sm">
<div class="container">
<a class="navbar-brand fw-semibold" href="dashboard.php">Документообіг</a>
<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#mainNavbar"><span class="navbar-toggler-icon"></span></button>
<div class="collapse navbar-collapse" id="mainNavbar">
<?php if (isLoggedIn()): ?>
<ul class="navbar-nav me-auto">
<li class="nav-item"><a class="nav-link" href="dashboard.php">Панель</a></li>
<li class="nav-item"><a class="nav-link" href="documents.php">Документи</a></li>
<li class="nav-item"><a class="nav-link" href="document_create.php">Створити</a></li>
<?php if (canApprove()): ?><li class="nav-item"><a class="nav-link"
href="approvals.php">Погодження</a></li><?php endif; ?>
<?php if (isAdmin()): ?><li class="nav-item"><a class="nav-link"
href="users.php">Користувачі</a></li><li class="nav-item"><a class="nav-link"
href="logs.php">Журнал дій</a></li><?php endif; ?>
</ul>
<div class="d-flex align-items-center gap-3 text-white"><span class="small"><?=
e($_SESSION['user']['full_name']) ?> (<?= roleName($_SESSION['user']['role'])
?>)</span><a class="btn btn-light btn-sm" href="logout.php">Вихід</a></div>
<?php endif; ?>
</div></div></nav>
<main class="container my-4">
<?php if ($m=flash('success')): ?><div class="alert alert-success"><?= e($m)
?></div><?php endif; ?>
<?php if ($m=flash('error')): ?><div class="alert alert-danger"><?= e($m) ?></div><?php
endif; ?>
```

app/includes/footer.php – нижня частина інтерфейсу

```
</main>
<footer class="text-center text-muted small py-4"><div>Кваліфікаційна робота:
«Інформаційна система електронного документообігу»</div><div>PHP + MySQL, ЧНУ ім. Петра
Могили, 2026</div></footer>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></scri
pt>
</body></html>
```

login.php – сторінка авторизації

```
<?php
require_once __DIR__.'/app/bootstrap.php';
if (isLoggedIn()) redirect('dashboard.php');
$error='';
if ($_SERVER['REQUEST_METHOD']==='POST'){
```

```
    checkCsrf(); $email=trim($_POST['email'] ?? ''); $password=$_POST['password'] ?? '';
    $s=$mysqli->prepare('SELECT * FROM users WHERE email=? AND is_active=1 LIMIT 1'); $s-
>bind_param('s',$email); $s->execute(); $u=$s->get_result()->fetch_assoc();
    if ($u && password_verify($password,$u['password'])){
$_SESSION['user']=['id'=>(int)$u['id'],'full_name'=>$u['full_name'],'email'=>$u['email'],
'role'=>$u['role']]; logAction($mysqli,(int)$u['id'],'Вхід до системи');
redirect('dashboard.php'); }
    $error='Невірний email або пароль.';
}
include __DIR__.'/app/includes/header.php'; ?>
<div class="auth-wrapper"><div class="card shadow"><div class="card-body p-4"><h3
class="text-center mb-3">Вхід до системи</h3><p class="text-muted text-
center">Інформаційна система електронного документообігу</p><?php if($error): ?><div
class="alert alert-danger"><?= e($error) ?></div><?php endif; ?><form
method="post"><input type="hidden" name="csrf_token" value="<?= csrfToken() ?>"><div
class="mb-3"><label class="form-label">Email</label><input class="form-control"
type="email" name="email" value="admin@example.com" required></div><div class="mb-
3"><label class="form-label">Пароль</label><input class="form-control" type="password"
name="password" value="123456" required></div><button class="btn btn-primary w-
100">Увійти</button></form><hr><div class="small text-
muted"><div><b>admin@example.com</b> / 123456</div><div><b>manager@example.com</b> /
123456</div><div><b>user@example.com</b> / 123456</div></div></div></div></div>
<?php include __DIR__.'/app/includes/footer.php'; ?>
```

dashboard.php – головна панель системи

```
<?php
require_once __DIR__.'/app/bootstrap.php'; requireLogin();
$stats=['total'=>0,'review'=>0,'approved'=>0,'rejected'=>0];
$r=$mysqli->query("SELECT ds.code, COUNT(d.id) total FROM document_statuses ds LEFT JOIN
documents d ON d.status_id=ds.id GROUP BY ds.code");
while($row=$r->fetch_assoc()){ if(isset($stats[$row['code']]))
$stats[$row['code']]=(int)$row['total']; }
$stats['total']=(int)$mysqli->query('SELECT COUNT(*) total FROM documents')-
>fetch_assoc()['total'];
$latest=$mysqli->query("SELECT d.id,d.document_number,d.title,d.created_at,ds.name
status_name,ds.code status_code,u.full_name FROM documents d JOIN document_statuses ds ON
ds.id=d.status_id JOIN users u ON u.id=d.author_id ORDER BY d.created_at DESC LIMIT 5");
include __DIR__.'/app/includes/header.php'; ?>
<h1 class="page-title">Головна панель</h1>
<div class="row g-3 mb-4"><?php foreach([['Усього документів','total'],['На
погодженні','review'],['Затверджено','approved'],['Відхилено','rejected']] as $item):
?><div class="col-md-3"><div class="card stat-card shadow-sm"><div class="card-body"><div
class="text-muted"><?= $item[0] ?></div><div class="stat-number"><?= $stats[$item[1]]
?></div></div></div></div><?php endforeach; ?></div>
<div class="card shadow-sm"><div class="card-header bg-white"><strong>Останні
документи</strong></div><div class="card-body"><table class="table table-hover align-
middle"><thead><tr><th>Номер</th><th>Назва</th><th>Автор</th><th>Статус</th><th>Дата</th>
</tr></thead><tbody><?php while($d=$latest->fetch_assoc()): ?><tr><td><?=
e($d['document_number']) ?></td><td><a href="document_view.php?id=<?= (int)$d['id']
?>"><?= e($d['title']) ?></a></td><td><?= e($d['full_name']) ?></td><td><span
class="badge bg-<?= statusBadgeClass($d['status_code']) ?>"><?= e($d['status_name'])
?></span></td><td><?= e($d['created_at']) ?></td></tr><?php endwhile;
?></tbody></table></div></div>
<?php include __DIR__.'/app/includes/footer.php'; ?>
```

ДОДАТОК Г

Модулі роботи з електронними документами

documents.php – реєстр документів, пошук і фільтрація

```
<?php
require_once __DIR__.'/app/bootstrap.php'; requireLogin();
$search=trim($_GET['search'] ?? ''); $status=trim($_GET['status'] ?? '');
$type=(int)($_GET['type'] ?? 0); $params=[]; $types='';
$sql="SELECT d.*, ds.name status_name, ds.code status_code, dt.name type_name,
u.full_name author_name FROM documents d JOIN document_statuses ds ON ds.id=d.status_id
JOIN document_types dt ON dt.id=d.document_type_id JOIN users u ON u.id=d.author_id WHERE
1=1";
if(!isAdmin() && !isManager()){ $sql.=' AND d.author_id=?'; $params[]=currentUserId();
$types.='i'; }
if($search!==''){ $sql.=' AND (d.title LIKE ? OR d.description LIKE ? OR
d.document_number LIKE ?)'; $like='%'.$search.'%'; $params[]=$like; $params[]=$like;
$params[]=$like; $types.='sss'; }
if($status!==''){ $sql.=' AND ds.code=?'; $params[]=$status; $types.='s'; }
if($type>0){ $sql.=' AND d.document_type_id=?'; $params[]=$type; $types.='i'; }
$sql.=' ORDER BY d.created_at DESC'; $s=$mysqli->prepare($sql); if($params) $s-
>bind_param($types,...$params); $s->execute(); $docs=$s->get_result();
$statuses=$mysqli->query('SELECT * FROM document_statuses ORDER BY id');
$typeList=$mysqli->query('SELECT * FROM document_types ORDER BY name'); include
__DIR__.'/app/includes/header.php'; ?>
<div class="d-flex justify-content-between align-items-center mb-3"><h1 class="page-title
mb-0">Документи</h1><a href="document_create.php" class="btn btn-primary">Створити
документ</a></div>
<div class="card shadow-sm mb-3"><div class="card-body"><form method="get" class="row g-
2"><div class="col-md-5"><input class="form-control" name="search" placeholder="Пошук"
value="<?= e($search) ?>"></div><div class="col-md-3"><select name="status" class="form-
select"><option value="">Усі статуси</option><?php while($st=$statuses->fetch_assoc()):
?><option value="<?= e($st['code']) ?>" <?= $status===$st['code']?'selected':'' ?>><?=
e($st['name']) ?></option><?php endwhile; ?></select></div><div class="col-md-3"><select
name="type" class="form-select"><option value="0">Усі типи</option><?php
while($t=$typeList->fetch_assoc()): ?><option value="<?= (int)$t['id'] ?>" <?=
$type===(int)$t['id']?'selected':'' ?>><?= e($t['name']) ?></option><?php endwhile;
?></select></div><div class="col-md-1 d-grid"><button class="btn btn-outline-
primary">OK</button></div></form></div></div>
<div class="card shadow-sm"><div class="card-body"><table class="table table-hover align-
middle"><thead><tr><th>Номер</th><th>Назва</th><th>Тип</th><th>Автор</th><th>Пріоритет</t
h><th>Статус</th><th>Дії</th></tr></thead><tbody><?php while($d=$docs->fetch_assoc()):
?><tr><td><?= e($d['document_number']) ?></td><td><?= e($d['title']) ?></td><td><?=
e($d['type_name']) ?></td><td><?= e($d['author_name']) ?></td><td><?= e($d['priority'])
?></td><td><span class="badge bg-<?= statusBadgeClass($d['status_code']) ?>"><?=
e($d['status_name']) ?></span></td><td><a class="btn btn-sm btn-info"
href="document_view.php?id=<?= (int)$d['id'] ?>">Перегляд</a><?php
if(userCanEditDocument($d)): ?> <a class="btn btn-sm btn-warning"
href="document_edit.php?id=<?= (int)$d['id'] ?>">Редагувати</a><?php endif;
?></td></tr><?php endwhile; ?></tbody></table></div></div>
<?php include __DIR__.'/app/includes/footer.php'; ?>
```

document_create.php – створення нового документа

```
<?php
require_once __DIR__.'/app/bootstrap.php'; requireLogin();
$typeList=$mysqli->query('SELECT * FROM document_types ORDER BY name');
$managers=$mysqli->query("SELECT id,full_name FROM users WHERE role IN
('admin','manager') AND is_active=1 ORDER BY full_name"); $errors=[];
if($_SERVER['REQUEST_METHOD']==='POST'){ checkCsrf(); $title=trim($_POST['title']??'');
$description=trim($_POST['description']??'');
$documentTypeId=(int)($_POST['document_type_id']??0);
$responsibleId=(int)($_POST['responsible_id']??0);
$priority=$_POST['priority']??'середній'; $dueDate=trim($_POST['due_date']??'');
```

```
if($title===''') $errors[]='Вкажіть назву документа.'; if($documentTypeId<=0)
$errors[]='Оберіть тип документа.'; if(!$errors){
$number=generateDocumentNumber($mysqli); $authorId=currentUserId(); $statusId=1;
$dueDateDb=$dueDate!==''?$dueDate:null; $resp=$responsibleId>0?$responsibleId:null;
$s=$mysqli->prepare('INSERT INTO documents
(document_number,title,description,document_type_id,author_id,responsible_id,status_id,pr
iority,due_date) VALUES (?,?,?,?,?,?,?,?,?)'); $s-
>bind_param('sssiiiiss',$number,$title,$description,$documentTypeId,$authorId,$resp,$stat
usId,$priority,$dueDateDb); $s->execute(); $docId=$s->insert_id;
if(!empty($_FILES['file']['name']) && $_FILES['file']['error']===UPLOAD_ERR_OK){
if($_FILES['file']['size']<=MAX_FILE_SIZE &&
allowedFileExtension($_FILES['file']['name'])){ if(!is_dir(UPLOAD_DIR))
mkdir(UPLOAD_DIR,0775,true); $orig=$_FILES['file']['name'];
$ext=strtolower(pathinfo($orig,PATHINFO_EXTENSION));
$stored=uniqid('doc_',true).'.'.$ext; $target=UPLOAD_DIR.$stored;
$web=UPLOAD_WEB_PATH.$stored;
if(move_uploaded_file($_FILES['file']['tmp_name'],$target)){
$size=(int)$_FILES['file']['size']; $fs=$mysqli->prepare('INSERT INTO document_files
(document_id,original_name,stored_name,file_path,file_size) VALUES (?,?,?,?,?)'); $fs-
>bind_param('isssi',$docId,$orig,$stored,$web,$size); $fs->execute(); } } }
addApprovalHistory($mysqli,$docId,$authorId,'created','Документ створено.');
logAction($mysqli,$authorId,"Створено документ {$number}: {$title}");
$_SESSION['flash_success']='Документ успішно створено.';
redirect('document_view.php?id='.$docId); }}
include __DIR__.'/app/includes/header.php'; ?>
<h1 class="page-title">Створення документа</h1><?php if($errors): ?><div class="alert
alert-danger"><?php foreach($errors as $er): ?><div><?= e($er) ?></div><?php endforeach;
?></div><?php endif; ?>
<form method="post" enctype="multipart/form-data" class="card shadow-sm"><div
class="card-body"><input type="hidden" name="csrf_token" value="<?= csrfToken() ?>"><div
class="row g-3"><div class="col-md-8"><label class="form-label">Назва
документа</label><input class="form-control" name="title" required></div><div class="col-
md-4"><label class="form-label">Тип</label><select class="form-select"
name="document_type_id" required><option value="">Оберіть тип</option><?php
while($t=$typeList->fetch_assoc()): ?><option value="<?= (int)$t['id'] ?>"><?=
e($t['name']) ?></option><?php endwhile; ?></select></div><div class="col-md-4"><label
class="form-label">Відповідальний</label><select class="form-select"
name="responsible_id"><option value="">Не обрано</option><?php while($m=$managers-
>fetch_assoc()): ?><option value="<?= (int)$m['id'] ?>"><?= e($m['full_name'])
?></option><?php endwhile; ?></select></div><div class="col-md-4"><label class="form-
label">Пріоритет</label><select class="form-select" name="priority"><option
value="низький">Низький</option><option value="середній"
selected>Середній</option><option value="високий">Високий</option></select></div><div
class="col-md-4"><label class="form-label">Термін</label><input class="form-control"
type="date" name="due_date"></div><div class="col-12"><label class="form-
label">Опис</label><textarea class="form-control" name="description"
rows="6"></textarea></div><div class="col-12"><label class="form-
label">Файл</label><input class="form-control" type="file" name="file"><div class="form-
text">pdf, doc, docx, xls, xlsx, txt, jpg, png. До 10 МБ.</div></div></div></div><div
class="card-footer bg-white"><button class="btn btn-primary">Зберегти</button> <a
href="documents.php" class="btn btn-outline-secondary">Назад</a></div></form>
<?php include __DIR__.'/app/includes/footer.php'; ?>
```

document_view.php – перегляд документа та історії обробки

```
<?php
require_once __DIR__.'/app/bootstrap.php'; requireLogin(); $id=(int)($_GET['id']??0);
$s=$mysqli->prepare("SELECT d.*, ds.name status_name, ds.code status_code, dt.name
type_name, u.full_name author_name, r.full_name responsible_name FROM documents d JOIN
document_statuses ds ON ds.id=d.status_id JOIN document_types dt ON
dt.id=d.document_type_id JOIN users u ON u.id=d.author_id LEFT JOIN users r ON
r.id=d.responsible_id WHERE d.id=?"); $s->bind_param('i',$id); $s->execute(); $doc=$s-
>get_result()->fetch_assoc(); if(!$doc) die('Документ не знайдено.');
if(!isAdmin()&&!isManager()&&(int)$doc['author_id']!==currentUserId()) die('Доступ
заборонено.');
if($_SERVER['REQUEST_METHOD']==='POST'){ checkCsrf(); if(isset($_POST['send_to_review'])
&& (int)$doc['author_id']===currentUserId() && $doc['status_code']==='draft'){
$sid=(int)$mysqli->query("SELECT id FROM document_statuses WHERE code='review'")-
>fetch_assoc()['id']; $u=$mysqli->prepare('UPDATE documents SET status_id=?,
```

```
updated_at=NOW() WHERE id=?'); $u->bind_param('ii',$sid,$id); $u->execute();
addApprovalHistory($mysqli,$id,currentUserId(),'sent_to_review','Документ передано на
погодження.'); logAction($mysqli,currentUserId(),'Документ передано на погодження:
'.$doc['document_number']); $_SESSION['flash_success']='Документ надіслано на
погодження.'; redirect('document_view.php?id='.$id); }}
$f=$mysqli->prepare('SELECT * FROM document_files WHERE document_id=? ORDER BY
uploaded_at DESC'); $f->bind_param('i',$id); $f->execute(); $files=$f->get_result();
$h=$mysqli->prepare('SELECT ah.*, u.full_name FROM approval_history ah JOIN users u ON
u.id=ah.user_id WHERE ah.document_id=? ORDER BY ah.created_at DESC'); $h-
>bind_param('i',$id); $h->execute(); $history=$h->get_result(); include
__DIR__.'/app/includes/header.php'; ?>
<div class="d-flex justify-content-between align-items-center mb-3"><h1 class="page-title
mb-0"><?= e($doc['document_number']) ?></h1><a href="documents.php" class="btn btn-
outline-secondary">До списку</a></div><div class="row g-3"><div class="col-lg-8"><div
class="card shadow-sm mb-3"><div class="card-header bg-white"><strong><?=
e($doc['title']) ?></strong></div><div class="card-body"><p><b>Тип:</b> <?=
e($doc['type_name']) ?></p><p><b>Автор:</b> <?= e($doc['author_name'])
?></p><p><b>Відповідальний:</b> <?= e($doc['responsible_name'] ?? 'Не призначено')
?></p><p><b>Пріоритет:</b> <?= e($doc['priority']) ?></p><p><b>Термін:</b> <?=
e($doc['due_date'] ?? 'Не визначено') ?></p><p><b>Статус:</b> <span class="badge bg-<?=
statusBadgeClass($doc['status_code']) ?>"><?= e($doc['status_name'])
?></span></p><hr><div class="document-description"><?= e($doc['description'])
?></div></div><div class="card-footer bg-white d-flex gap-2"><?php
if(userCanEditDocument($doc)): ?><a href="document_edit.php?id=<?= (int)$doc['id'] ?>"
class="btn btn-warning">Редагувати</a><?php endif; ?><?php
if((int)$doc['author_id']===currentUserId() && $doc['status_code']==='draft'): ?><form
method="post"><input type="hidden" name="csrf_token" value="<?= csrfToken() ?>"><button
class="btn btn-success" name="send_to_review">Надіслати на
погодження</button></form><?php endif; ?></div></div><div class="card shadow-sm"><div
class="card-header bg-white"><strong>Історія погодження</strong></div><div class="card-
body"><?php while($r=$history->fetch_assoc()): ?><div class="border-bottom pb-2 mb-
2"><div><b><?= e($r['full_name']) ?></b> — <?= e($r['action']) ?></div><div class="small
text-muted"><?= e($r['created_at']) ?></div><?php if($r['comment']): ?><div><?=
e($r['comment']) ?></div><?php endif; ?></div><?php endwhile; ?></div></div></div><div
class="col-lg-4"><div class="card shadow-sm"><div class="card-header bg-
white"><strong>Файли</strong></div><div class="card-body"><?php if($files->num_rows===0):
?><p class="text-muted">Файли не додано.</p><?php endif; ?><?php while($file=$files-
>fetch_assoc()): ?><div class="mb-2"><a href="<?= e($file['file_path']) ?>"
target="_blank"><?= e($file['original_name']) ?></a><div class="small text-muted"><?=
round(((int)$file['file_size'])/1024,2) ?> КБ</div></div><?php endwhile;
?></div></div></div></div>
<?php include __DIR__.'/app/includes/footer.php'; ?>
```

document_edit.php – редагування документа

```
<?php
require_once __DIR__.'/app/bootstrap.php'; requireLogin(); $id=(int)($_GET['id']??0);
$s=$mysqli->prepare('SELECT d.*, ds.code status_code FROM documents d JOIN
document_statuses ds ON ds.id=d.status_id WHERE d.id=?'); $s->bind_param('i',$id); $s-
>execute(); $doc=$s->get_result()->fetch_assoc(); if(!$doc) die('Документ не знайдено.');
if(!userCanEditDocument($doc)) die('Редагування заборонено.'); $typeList=$mysqli-
>query('SELECT * FROM document_types ORDER BY name'); $managers=$mysqli->query("SELECT
id,full_name FROM users WHERE role IN ('admin','manager') AND is_active=1 ORDER BY
full_name"); $errors=[]; if($_SERVER['REQUEST_METHOD']==='POST'){ checkCsrf();
$title=trim($_POST['title']??''); $description=trim($_POST['description']??'');
$type=(int)($_POST['document_type_id']??0); $resp=(int)($_POST['responsible_id']??0);
$priority=$_POST['priority']??'середній'; $due=trim($_POST['due_date']??'');
if($title==='') $errors[]='Назва є обов’язковою.'; if(!$errors){
$dueDb=$due!==''?$due:null; $respDb=$resp>0?$resp:null; $u=$mysqli->prepare('UPDATE
documents SET title=?, description=?, document_type_id=?, responsible_id=?, priority=?,
due_date=?, updated_at=NOW() WHERE id=?'); $u-
>bind_param('ssiissi',$title,$description,$type,$respDb,$priority,$dueDb,$id); $u-
>execute(); addApprovalHistory($mysqli,$id,currentUserId(),'edited','Документ
відредаговано.'); logAction($mysqli,currentUserId(),'Відредаговано документ:
'.$doc['document_number']); $_SESSION['flash_success']='Зміни збережено.';
redirect('document_view.php?id='.$id); }} include __DIR__.'/app/includes/header.php'; ?>
<h1 class="page-title">Редагування документа</h1><?php if($errors): ?><div class="alert
alert-danger"><?php foreach($errors as $er): ?><div><?= e($er) ?></div><?php endforeach;
?></div><?php endif; ?><form method="post" class="card shadow-sm"><div class="card-
```

```
body"><input type="hidden" name="csrf_token" value="<?= csrfToken() ?>"><div class="row
g-3"><div class="col-md-8"><label class="form-label">Назва</label><input class="form-
control" name="title" value="<?= e($doc['title']) ?>" required></div><div class="col-md-
4"><label class="form-label">Тип</label><select class="form-select"
name="document_type_id" required><?php while($t=$typeList->fetch_assoc()): ?><option
value="<?= (int)$t['id'] ?>" <?=
(int)$doc['document_type_id']===(int)$t['id']?'selected':'' ?>><?= e($t['name'])
?></option><?php endwhile; ?></select></div><div class="col-md-4"><label class="form-
label">Відповідальний</label><select class="form-select" name="responsible_id"><option
value="">Не обрано</option><?php while($m=$managers->fetch_assoc()): ?><option value="<?=
(int)$m['id'] ?>" <?= (int)$doc['responsible_id']===(int)$m['id']?'selected':'' ?>><?=
e($m['full_name']) ?></option><?php endwhile; ?></select></div><div class="col-md-
4"><label class="form-label">Пріоритет</label><select class="form-select"
name="priority"><?php foreach(['низький','середній','високий'] as $p): ?><option
value="<?= $p ?>" <?= $doc['priority']===$p?'selected':'' ?>><?= $p ?></option><?php
endforeach; ?></select></div><div class="col-md-4"><label class="form-
label">Термін</label><input class="form-control" type="date" name="due_date" value="<?=
e($doc['due_date']) ?>"></div><div class="col-12"><label class="form-
label">Опис</label><textarea class="form-control" name="description" rows="6"><?=
e($doc['description']) ?></textarea></div></div></div><div class="card-footer bg-
white"><button class="btn btn-primary">Зберегти</button> <a
href="document_view.php?id=<?= (int)$doc['id'] ?>" class="btn btn-outline-
secondary">Скасувати</a></div></form><?php include __DIR__.'/app/includes/footer.php'; ?>
```

ДОДАТОК Д

Модулі погодження, користувачів і журналу дій

approvals.php – погодження або відхилення документів

```
<?php
require_once __DIR__.'/app/bootstrap.php'; requireApprover();
if($_SERVER['REQUEST_METHOD']==='POST'){ checkCsrf();
$docId=(int)($_POST['document_id']??0); $decision=$_POST['decision']??'';
$comment=trim($_POST['comment']??'');
if(!in_array($decision,['approved','rejected'],true)){
$_SESSION['flash_error']='Некоректне рішення.'; redirect('approvals.php'); }
$st=$mysqli->prepare('SELECT id FROM document_statuses WHERE code=?'); $st-
>bind_param('s',$decision); $st->execute(); $sid=(int)$st->get_result()-
>fetch_assoc()['id']; $u=$mysqli->prepare('UPDATE documents SET status_id=?,
updated_at=NOW() WHERE id=?'); $u->bind_param('ii',$sid,$docId); $u->execute();
addApprovalHistory($mysqli,$docId,currentUserId(),$decision,$comment);
logAction($mysqli,currentUserId(),($decision==='approved'?'Затверджено':'Відхиле
но').' документ ID '.$docId); $_SESSION['flash_success']='Рішення збережено.';
redirect('approvals.php'); }
$res=$mysqli->query("SELECT d.*, ds.name status_name, ds.code status_code,
dt.name type_name, u.full_name author_name FROM documents d JOIN
document_statuses ds ON ds.id=d.status_id JOIN document_types dt ON
dt.id=d.document_type_id JOIN users u ON u.id=d.author_id WHERE ds.code='review'
ORDER BY d.created_at ASC"); include __DIR__.'/app/includes/header.php'; ?>
<h1 class="page-title">Погодження документів</h1><div class="card shadow-
sm"><div class="card-body"><?php if($res->num_rows===0): ?><div class="alert
alert-info mb-0">Документів на погодженні немає.</div><?php else: ?><table
class="table table-hover align-
middle"><thead><tr><th>Номер</th><th>Назва</th><th>Автор</th><th>Тип</th><th>Дія
</th></tr></thead><tbody><?php while($d=$res->fetch_assoc()): ?><tr><td><?=
e($d['document_number']) ?></td><td><a href="document_view.php?id=<?=
(int)$d['id'] ?>"><?= e($d['title']) ?></a></td><td><?= e($d['author_name'])
?></td><td><?= e($d['type_name']) ?></td><td><form method="post" class="d-flex
gap-2"><input type="hidden" name="csrf_token" value="<?= csrfToken() ?>"><input
type="hidden" name="document_id" value="<?= (int)$d['id'] ?>"><input
class="form-control form-control-sm" name="comment"
placeholder="Коментар"><button class="btn btn-sm btn-success" name="decision"
value="approved">Затвердити</button><button class="btn btn-sm btn-danger"
name="decision" value="rejected">Відхилити</button></form></td></tr><?php
endwhile; ?></tbody></table><?php endif; ?></div></div><?php include
__DIR__.'/app/includes/footer.php'; ?>
```

users.php – керування користувачами

```
<?php
require_once __DIR__.'/app/bootstrap.php'; requireAdmin(); $errors=[];
if($_SERVER['REQUEST_METHOD']==='POST'){ checkCsrf();
$name=trim($_POST['full_name']??''); $email=trim($_POST['email']??'');
$role=$_POST['role']??'user'; $pass=$_POST['password']??'123456';
$pos=trim($_POST['position']??''); $dep=trim($_POST['department']??'');
if($name===''||$email==='') $errors[]='ПІБ та email є обов’язковими.'; if(!$errors){
$hash=password_hash($pass,PASSWORD_DEFAULT); $s=$mysqli->prepare('INSERT INTO users
(full_name,email,password,role,position,department) VALUES (?,?,?,?,?,?)'); $s-
>bind_param('ssssss',$name,$email,$hash,$role,$pos,$dep); if($s->execute()){
logAction($mysqli,currentUserId(),'Створено користувача: '.$email);
$_SESSION['flash_success']='Користувача створено.'; redirect('users.php'); } else
$errors[]='Email уже існує або виникла помилка.'; }} $users=$mysqli->query('SELECT * FROM
users ORDER BY created_at DESC'); include __DIR__.'/app/includes/header.php'; ?>
```

```
<h1 class="page-title">Користувачі</h1><?php if($errors): ?><div class="alert alert-
danger"><?php foreach($errors as $er): ?><div><?= e($er) ?></div><?php endforeach;
?></div><?php endif; ?><div class="row g-3"><div class="col-lg-5"><div class="card
shadow-sm"><div class="card-header bg-white"><b>Новий користувач</b></div><div
class="card-body"><form method="post"><input type="hidden" name="csrf_token" value="<?=
csrfToken() ?>"><div class="mb-3"><label class="form-label">ПІБ</label><input
class="form-control" name="full_name" required></div><div class="mb-3"><label
class="form-label">Email</label><input class="form-control" type="email" name="email"
required></div><div class="mb-3"><label class="form-label">Пароль</label><input
class="form-control" name="password" value="123456"></div><div class="mb-3"><label
class="form-label">Роль</label><select class="form-select" name="role"><option
value="user">Користувач</option><option value="manager">Керівник</option><option
value="admin">Адміністратор</option></select></div><div class="mb-3"><label class="form-
label">Посада</label><input class="form-control" name="position"></div><div class="mb-
3"><label class="form-label">Підрозділ</label><input class="form-control"
name="department"></div><button class="btn btn-
primary">Створити</button></form></div></div></div><div class="col-lg-7"><div class="card
shadow-sm"><div class="card-header bg-white"><b>Список користувачів</b></div><div
class="card-body"><table class="table table-
hover"><thead><tr><th>ПІБ</th><th>Email</th><th>Роль</th><th>Підрозділ</th></tr></thead><
tbody><?php while($u=$users->fetch_assoc()): ?><tr><td><?= e($u['full_name'])
?></td><td><?= e($u['email']) ?></td><td><?= roleName($u['role']) ?></td><td><?=
e($u['department']) ?></td></tr><?php endwhile;
?></tbody></table></div></div></div></div><?php include
__DIR__.'/app/includes/footer.php'; ?>
```

logs.php – журнал операцій користувачів

```
<?php require_once __DIR__.'/app/bootstrap.php'; requireAdmin(); $logs=$mysqli-
>query('SELECT l.*,u.full_name FROM activity_logs l LEFT JOIN users u ON u.id=l.user_id
ORDER BY l.created_at DESC LIMIT 300'); include __DIR__.'/app/includes/header.php'; ?><h1
class="page-title">Журнал дій</h1><div class="card shadow-sm"><div class="card-
body"><table class="table table-
hover"><thead><tr><th>Дата</th><th>Користувач</th><th>Дія</th><th>IP</th></tr></thead><tb
ody><?php while($l=$logs->fetch_assoc()): ?><tr><td><?= e($l['created_at'])
?></td><td><?= e($l['full_name'] ?? 'Система') ?></td><td><?= e($l['action'])
?></td><td><?= e($l['ip_address']) ?></td></tr><?php endwhile;
?></tbody></table></div></div><?php include __DIR__.'/app/includes/footer.php'; ?>
```