Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Медіасистем та технологій
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Розробка автономного онлайн-інструменту для створення та редагування електронних видань
(тема)

Виконав:
здобувач 4 року навчання,
групи ВПВПС-21-1

Гліб СНУРНИКОВ
(власне ім'я, прізвище)

Спеціальність 186 Видавництво та поліграфія
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма
Видавничо-поліграфічна справа
(повна назва освітньої програми)

Керівник ст. викл. Лана МОРОЗОВА
(посада, власне ім'я, прізвище)

Допускається до захисту
Завідувач кафедри МСТ _____ Жанна ДЕЙНЕКО
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
Кафедра Медіасистем та технологій
Рівень вищої освіти перший (бакалаврський)
Спеціальність 186 Видавництво та поліграфія
Тип програми Освітньо-професійна
Освітня програма Видавничо-поліграфічна справа
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри МСТ_____
(підпис)
« 19 » травня 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві *Снурникову Глібу Олеговичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Розробка автономного онлайн інструменту для сторвення та редагування онлайн видань*

Затверджена наказом по університету від 19 травня 2025 р. № 385 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 9 червня 2025 р.

3. Вихідні дані до роботи
Національні та міжнародні стандарти для електронних друкованих видань; Вид і призначення Веб-видання: веб-сайт; призначення: інструменту для створення та редагування електронних видань ; Мови розробки веб-видання: HTML, CSS, Javascript, Kotlin, Scala, Python; використанні фреймворки: React, Nunjucks; середовище розповсюдження: Інтернет.

4. Перелік питань, що потрібно опрацювати в роботі
Вступ, Аналіз завдання на кваліфікаційну роботу; Визначення цільової аудиторії; Аналіз аналогів; Визначення логіки взаємодії; Обґрунтування вибору програмного забезпечення; Розробка модульної сітки; Обґрунтування дизайнерського рішення; Розробка дизайн-макету; Створення інструменту; Тестування; Розповсюдження; Економічна частина; Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п. 5 включається до завдання за рішенням випускової кафедри)
Аналіз завдання на кваліфікаційну роботу; Визначення цільової аудиторії; Аналіз аналогів; Визначення логіки взаємодії; Обґрунтування вибору програмного забезпечення; Розробка модульної сітки; Обґрунтування дизайнерського рішення; Розробка дизайн-макету; Створення інструменту; Тестування; Розповсюдження; Економічна частина; Висновки.

6. Консультанти розділів роботи (п. 6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п. 1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	ст. викл. Морозова Л.Ю.		08.06.2025
Економічна частина	ас. Легеза О.М.		06.06.2025

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	19.05.2025	Викон.
2	Аналіз завдання на кваліфікаційну роботу, визначення цілей і задач проєктування	20.05.2025	Викон.
3	Аналітичний аналіз технологій пов'язанних із темою розробки	24.05.2025	Викон.
4	Проєктування логіки взаємодії	25.05.2025	Викон.
5	Розробка дизайнерського рішення	26.05.2025	Викон.
6	Розробка веб-сайту на основі дизайн-макету	28.05.2025	Викон.
7	Економічна частина	02.06.2025	Викон.
8	Оформлення пояснювальної записки	03.06.2025	Викон.
9	Оформлення графічної частини	04.06.2025	Викон.

Дата видачі завдання 19 травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ст. викл. Лана МОРОЗОВА
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 58 с., 3 табл., 19 рис., 2 дод., 50 джерел.

ВЕБ-САЙТ, ДИЗАЙН, РОЗРОБКА, ВЕБ-СЕРВІС, КОНВЕРТАЦІЯ, РЕДАГУВАННЯ, ЕЛЕКТРОННІ ВИДАННЯ.

Робота присвячена розробці автономного онлайн-інструменту для створення, конвертації та редагування електронних видань. Метою дослідження є аналіз сучасних технологій та методик, які дозволяють створити веб-сервіс з можливістю завантаження PDF-документів, автоматичної їх конвертації в структурований Markdown і подальшим формуванням адаптивного HTML-контенту з інтуїтивно зрозумілим інтерфейсом для користувача.

У роботі детально описані етапи створення автономного інструменту, включаючи аналіз аналогів, визначення логіки взаємодії, проектування дизайну, вибір стеку технологій та архітектуру мікросервісів. Особливу увагу приділено опису механізмів збереження формату та структури вихідних документів, методам їх перетворення, а також впровадженню системи авторизації та управління контентом.

Результатом роботи став повноцінний автономний веб-інструмент, який забезпечує високий рівень кастомізації, адаптивності та безпеки, а також спрощує процес редагування та створення електронних публікацій для широкого кола користувачів.

ABSTRACT

Explanatory note of the qualification work: 58 p., 3 tab, 19 pic, 2 app, 50 sources.

WEBSITE, DESIGN, DEVELOPMENT, WEB SERVICE, CONVERSION, EDITING, ELECTRONIC PUBLICATIONS.

The work is devoted to the development of an autonomous online tool for creating, converting and editing electronic publications. The purpose of the research is to analyze modern technologies and methods that allow creating a web-service with the ability to download PDF-documents, automatically convert them into structured Markdown and further form adaptive HTML-content with an intuitive user interface.

The work describes in detail the stages of creating an autonomous tool, including the analysis of analogues, definition of interaction logic, design design, selection of a technology stack and microservices architecture. Particular attention is paid to the description of mechanisms for preserving the format and structure of source documents, methods for their conversion, as well as the implementation of an authorization and content management system.

The result of the work was a full-fledged autonomous web tool that provides a high level of customization, adaptability and security, and also simplifies the process of editing and creating electronic publications for a wide range of users

ЗМІСТ

С.

ВСТУП

У сучасному інформаційному середовищі електронні видання стали одним із головних каналів комунікації для науковців, підприємств та творчих ініціатив. Від наукових журналів та освітніх підручників до бізнес-презентацій і маркетингових брошур – користувачі все частіше обирають цифрові формати за їхню доступність, швидкість оновлення та інтерактивні можливості. Однак, незважаючи на поширеність таких рішень, багато платформ лишаються або надто складними для швидкого створення публікацій, або ж навпаки – занадто примітивними, аби задовольнити потреби професійних користувачів.

Сучасні системи для роботи з електронними виданнями часто мають жорсткі обмеження в налаштуванні дизайну й структури: користувачі змушені обирати серед готових тем і макетів, що не завжди відповідають корпоративній стилістиці чи специфічним вимогам проекту.

У зв’язку з цим особливою актуальністю вирізняються рішення, які поєднують простий інтерфейс із гнучкими можливостями кастомізації: від розміщення тексту та графіки до налаштування кольорової палітри, шрифтів та інтерактивних елементів. Автономність таким системам надає захищений режим роботи без постійного підключення до інтернету або зовнішніх API, що важливо для організацій із підвищеними вимогами до безпеки та незалежності.

Саме це й стало підґрунтям для реалізації дипломного проєкту «Розробка автономного онлайн-інструменту для створення та редагування електронних видань». Завданням роботи є створити веб-додаток із інтуїтивно зрозумілим редактором і потужним механізмом налаштування шаблонів, який дозволятиме оперативно генерувати публікації єдиного стилю. Користувачі отримають змогу легко працювати з контентом, додавати інтерактивні елементи й експортувати фінальні матеріали у різних форматах без залучення розробників на кожному етапі.

1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

Pressbooks – відкрита платформа для створення та публікації цифрових книг і освітніх ресурсів (OER). Вона дозволяє структурувати видання за допомогою розділів, глав і підрозділів, а також експортувати готовий контент у формати PDF, EPUB, MOBI чи веб-книги. Однак під час самостійного розгортання на власному сервері користувач стикається з необхідністю налаштування Docker-контейнерів або WordPress-середовища, що вимагає технічної підготовки. А базова кастомізація дизайну обмежується вибором серед готових тем та налаштуванням CSS-оваляїв, тобто тонка зміна палітри, шрифтів чи макетів вимагає знання WordPress-шаблонів і PHP. Інтерактивні елементи (відео, опитування) додаються через плагіни WordPress, що підвищує ризик конфліктів після оновлень. Підсумовуючи, Pressbooks має ряд обмежень, що обумовлюють потребу в альтернативному рішенні. Основними користувачами Pressbooks є викладачі ВНЗ та наукові співробітники, які готують навчальні матеріали. Вони мають середню комп’ютерну грамотність, очікують простоти у створенні розділів і не потребують глибокої кастомізації дизайну. Однак платформа не задовольняє вимог до автономності та гнучкого редагування, а також утруднює швидке підключення нових шаблонів і автоматичне оновлення контенту без залучення технічної команди.

GitBook – статичний генератор документації та інтерактивних підручників, що перетворює Markdown-файли в веб-сайт із пошуковою системою, навігацією по розділах і підтримкою плагінів. Він конвертує файли в веб-сайт з пошуковою системою, навігацією та базовою аналітикою переглядів. Усі зміни робляться шляхом редагування Markdown у репозиторії Git і автоматичного запуску CI/CD-скриптів, що створює бар’єр для викладачів і маркетологів, які не знайомі з Git та командним рядком. Кастомізація дизайну обмежується вибором кольорових схем і підключенням CSS-оверрайдів; інтеграція мультимедійних чи інтерактивних елементів потребує написання плагінів на JavaScript. GitBook не

підтримує автономне редагування без Інтернету та хоститься на gitbook.io у безкоштовному плані або потребує власного Node.js-сервера для self-hosting. GitBook орієнтований на технічних письменників, розробників документації та стартапи. Для викладачів і маркетологів, які не володіють Markdown, ця модель не є інтуїтивною. Інструмент потребує постійного підключення до інтернету й залежить від зовнішніх сервісів CI/CD для автогенерації контенту за пуш у репозиторій.

Joomag – комерційна хмарна платформа для створення інтерактивних цифрових публікацій з підтримкою відео, анімацій, опитувань та вбудованих форм. Інтерфейс drag-and-drop дозволяє легко розміщувати мультимедіа, проте повна кастомізація дизайну за допомогою коду відсутня: користувач може обирати лише серед наданих шаблонів і налаштовувати базові опції (кольори, шрифти). Всі дані зберігаються на серверах Joomag, що виключає автономну роботу в ізольованому середовищі та викликає сумніви стосовно захисту конфіденційної інформації. Ліцензійні плани доволі дорогі для невеликих академічних груп або підприємств з обмеженим бюджетом. Joomag використовують маркетингові відділи підприємств і видавці, які цінують інтерактивність та динамічну аналітику поведінки читачів. Для викладачів і малих організацій такі ресурси надто затратні, а відсутність автономного режиму та відкритого доступу до шаблонів не відповідає вимогам безпеки й гнучкості.

Беручи до уваги мету створення автономного веб-інструменту з гнучкими шаблонами для підготовки й редагування електронних видань, доцільно виокремити три основні категорії його користувачів:

а) викладачі вищих навчальних закладів (25-60 років):

1) профіль: науковці, асистенти та доценти, які готують лекційні й методичні матеріали;

2) потреби: простий WYSIWYG-редактор без необхідності вивчення розмітки, можливість швидко публікувати та оновлювати контент, інтеграція з LMS (Moodle, Canvas) для синхронізації доступу студентів;

3) обмеження: середній рівень комп’ютерної грамотності; нестабільне

підключення під час польових виїздів чи семінарів;

б) представники освітніх установ і дослідницькі групи (30-65 років):

1) профіль: адміністратори бібліотек, працівники науково-дослідних центрів, що формують відкриті освітні ресурси (OER);

2) потреби: self-hosting і контроль збереження даних, тонка настройка шаблонів оформлення відповідно до бренд-буку, підтримка великих обсягів контенту з аналітикою переглядів і версій;

3) обмеження: високі вимоги до безпеки й захисту персональних даних; обмежена технічна підтримка – бажано мінімальні налаштування інфраструктури;

в) підприємства та організації (28-55 років):

1) профіль: маркетологи, PR-спеціалісти і менеджери проєктів, що готують корпоративні звіти, брошури та презентації;

2) потреби: єдина візуальна стилістика, інтерактивні елементи (відео, кнопки, опитування), можливість обмежити доступ або монетизувати видання;

3) обмеження: необхідність автономної роботи без зовнішніх сервісів, швидке навчання співробітників, бюджетні обмеження на ліцензії.

Ідеальний інструмент має поєднувати простоту WYSIWYG-редактору (для викладачів), можливість self-hosting та контроль даних (для освітніх установ) і багаті опції інтерактивності з брендовими шаблонами (для підприємств). Демографічно це користувачі віком від 25 до 65 років з різним технічним рівнем, які потребують автономної, безпечної й гнучкої системи для оперативного створення та публікації контенту

2 АНАЛІТИЧНИЙ ОГЛЯД ЛІТЕРАТУРИ ЗА ТЕМОЮ

2.1 Поняття електронного видання

У науковій літературі та нормативних документах електронне видання визначається як самостійний електронний документ або сукупність електронних документів, що пройшли редакційно-видавничу обробку, мають вихідні відомості та призначені для розповсюдження в незмінному вигляді[1] Тобто електронне видання – це електронний аналог друкованого видання, підготовлений до тиражування засобами електронних технологій. Визначальною ознакою є наявність офіційних вихідних даних (назва, автор(и), видавець, рік видання тощо), що робить електронний документ виданням у видавничому розумінні. Українські дослідники І. Антоненко та О. Баркова додають, що електронне видання має бути самостійним завершеним електронним ресурсом, який може функціонувати незалежно від виробника (у тому числі через телекомунікаційні мережі), і всі його копії повністю відповідають оригіналу. Таким чином, електронне видання поєднує властивості традиційного видання (редакційна підготовка, стандартизовані вихідні відомості) з формою електронного документа, що зберігається і розповсюджується цифровими засобами.

2.2 Класифікація електронних видань

Електронні видання розрізняють за низкою ознак, що відображають їх зміст, форму та спосіб розповсюдження. У загальному випадку можна виділити такі класифікаційні критерії електронних видань [1]:

– за наявністю друкованого еквівалента. Виокремлюють електронні аналоги друкованих видань, які здебільшого відтворюють структуру та зміст

відповідного друкованого видання, та оригінальні електронні видання, що не мають друкованих аналогів;

– за природою основної інформації. Залежно від виду інформаційного наповнення розрізняють текстові видання (електронні книги, журнали, статті тощо), графічні видання (фотоальбоми, графічні колекції), аудіо-видання (цифрові аудіокниги, подкасти), відео-видання (відеоконтент, навчальні фільми) та мультимедійні видання, що поєднують різнорідні типи даних (наприклад, електронні підручники з текстом, аудіо та відео фрагментами);

– за періодичністю. Видання можуть бути неперіодичними (одноразові публікації, як-от окрема електронна книга або звіт) та періодичними (що виходять серіями або випусками з визначеною частотою, наприклад, електронні журнали, серіальні видання);

– за цільовим призначенням і аудиторією. Розрізняють наукові, навчальні, довідкові, інформаційно-новинні, розважальні та інші електронні видання залежно від мети їх створення та потреб цільової аудиторії

– за способом розповсюдження та носієм. Електронні видання можуть розповсюджуватися онлайн через мережу Інтернет (веб-сайти, онлайн-платформи) або на фізичних носіях (CD/DVD, USB-носії тощо) для локального використання;

– за характером взаємодії. Виокремлюють статичні електронні видання, що містять незмінний контент (аналогічно друкованим), та інтерактивні видання, які дозволяють користувачеві взаємодіяти з контентом (через гіперпосилання, пошук, мультимедійні елементи, інтерактивні тести тощо);

Існують і більш деталізовані класифікації. Зокрема такі ознаки, як форма функціонування (онлайн/офлайн), тип замовника або видавця, охоплення території розповсюдження, наявність філіальної мережі тощо [1]. Однак для більшості практичних цілей достатньо наведених вище основних критеріїв, які дозволяють однозначно визначити вид електронного видання за його ключовими характеристиками.

2.3 Структура та етапи створення електронного видання

Процес створення електронного видання включає низку послідовних стадій, кожна з яких охоплює визначене коло завдань від підготовки контенту до випуску готового продукту. На основі узагальнення літературних джерел [1,2] можна виділити такі основні етапи розробки електронного видання:

– планування і аналіз вимог. На початковому етапі здійснюється визначення мети видання та аналіз цільової аудиторії. Видавець має дослідити потреби майбутніх читачів, тематичну нішу та вимоги до контенту і функцій видання [2];

– збирання та підготовка контенту. На другому етапі відбувається наповнення видання матеріалами. Проводиться збір необхідних даних: текстових матеріалів, ілюстрацій, мультимедійних елементів (аудіо, відео) тощо [2]. Контент проходить редакційну підготовку – літературне редагування текстів, перевірку фактів, коректуру, технічне редагування зображень і медіафайлів;

– проєктування структури та дизайну. На цьому етапі розробляється логічна структура електронного видання і дизайн його інтерфейсу. Визначається система навігації (зміст, меню, посилання між розділами), макет сторінок або екранів, оформлення текстових і графічних елементів;

– верстка та технічна реалізація. На цьому етапі здійснюється безпосереднє створення електронного видання у вибраному форматі;

– тестування. Важливою стадією є перевірка електронного видання перед випуском. Тестування проводять як з технічної точки зору (чи коректно відображається контент у різних середовищах і на різних пристроях, чи працюють всі інтерактивні елементи, чи відповідає файл формату стандартам), так і з точки зору користувача (зручність навігації, зрозумілість інтерфейсу, відсутність помилок у тексті);

– публікація та підтримка. Завершальний етап – випуск електронного видання у світ. Залежно від виду, це може бути розміщення на веб-сайті,

додавання до онлайн-бібліотеки або каталогу, тиражування на цифрових носіях і розповсюдження серед користувачів.

2.4 Технічні вимоги до електронних видань

Забезпечення високої юзабіліті є одним із ключових технічних завдань при розробці електронних публікацій. Як зазначає Якоб Нільсен, інтерфейс повинен бути інтуїтивно зрозумілим і мінімізувати когнітивне навантаження користувача: доцільно дотримуватися принципу «не змушуйте мене думати» і забезпечити однозначне відображення всіх елементів навігації та керування в режимі реального часу [8].

Підкреслюється, що будь-який додатковий ефект на сторінці повинен мати чітку функцію й одразу транслюватися інтуїтивно зрозумілим для читача чином [9]. З точки зору технічної реалізації, це означає оптимізувати CSS і JavaScript, аби навіть при слабкому інтернет-з’єднанні елементи інтерфейсу завантажувалися швидко та без помітних затримок

До електронних видань висувається ряд технічних вимог, пов’язаних із форматами файлів, структурою даних і сумісністю з програмними засобами. Розглянемо основні з них:

– електронне видання має бути представлено у форматі, що забезпечує зручність читання та сумісність з розповсюдженими пристроями й програмами. Найпоширенішими форматами є PDF та EPUB. Формат PDF (Portable Document Format) широко застосовується для електронних книг і документів, оскільки точно зберігає макет сторінки та підтримується на будь-яких пристроях без спеціалізованого ПЗ [3]. Його перевагою є універсальність і повна відповідність друкованому вигляду, що важливо для наукових і навчальних публікацій [1]. Водночас недоліком PDF є фіксований розмір сторінки, що ускладнює читання на малих екранах та не дає можливості динамічно підлаштовувати текст (реагувати на розмір екрану). Формат EPUB (Electronic Publication) натомість забезпечує *репагінацію* і гнучкість верстки:

текст у EPUB-файлі автоматично переламається під розмір екрану чи вікна, шрифт може змінювати сам читач під свої уподобання. EPUB є міжнародним стандартом для електронних книг і підтримується більшістю пристроїв для читання (рідерів) та мобільних додатків [4];

– електронне видання повинно містити всі обов'язкові структурні елементи, притаманні виданням. Зокрема, повинні бути представлена титульна інформація (назва, автори, вихідні дані про видавця), зміст (для навігації по розділах), бібліографічні посилання (якщо це наукова праця), покажчики та інші елементи структури за потреби;

– для забезпечення єдності дизайну і спрощення процесу верстки застосовуються шаблони оформлення. Шаблон електронного видання визначає параметри форматування для різних типів контенту: стилі заголовків, основного тексту, підписів до рисунків, таблиць, блоків цитат тощо. Підтримка шаблонів у системі підготовки видання дозволяє автоматично застосувати затверджений стиль до всіх елементів, що підвищує однорідність і якість верстки;

– електронне видання має коректно відображатися на різних програмно-технічних платформах. Це означає дотримання відкритих стандартів та перевірку сумісності з популярними програмами-проглядачами (Adobe Acrobat Reader, калібрувальними програмами для EPUB тощо) і пристроями (настільні комп'ютери, планшети, смартфони, спеціалізовані e-reader). Бажано, щоб верстка була *адаптивною*: контент повинен читабельно виглядати як на великому екрані, так і на маленькому. Для веб-видання адаптивність забезпечується через технології HTML5/CSS3 (медіа-запити для різних розмірів екрану).

2.5 Вимоги до WYSIWYG-редакторів електронних видань

У системах створення та редагування електронних публікацій ключову роль відіграють WYSIWYG-редактори (англ. *What You See Is What You Get* – «що

бачиш, те і отримуєш»). Це такі редактори, які дозволяють автору та редактору працювати з контентом у тому вигляді, в якому він надалі буде представлений читачеві [5]. На відміну від редагування сирого коду розмітки, WYSIWYG-підхід надає *візуальний інтерфейс*, де користувач без знання мов розмітки може форматувати текст, вставляти зображення, таблиці та інші елементи, одразу бачачи кінцевий результат форматування. Основні вимоги до WYSIWYG-редактора в системі електронних видань такі:

– редактор повинен мати зрозумілий графічний інтерфейс з панелями інструментів, піктограмами та меню, аналогічними до стандартних текстових процесорів [5]. Користувач має змогу застосовувати стилі й форматування (напівжирний, курсив, списки, вирівнювання тексту тощо) за допомогою кнопок і меню, без ручного внесення тегів чи кодів розмітки;

– усі зміни, які вносить користувач, мають одразу відображатися в редакторі так, як вони виглядатимуть у готовому виданні [5]. Така *миттєва візуалізація* дозволяє редактору оперативно оцінити верстку – наприклад, як текст перетікає довкола зображення, чи правильно розділилися сторінки, як виглядає документ на різних розмірах екрана (якщо це онлайн-видання);

– WYSIWYG-редактор має підтримувати не тільки базове форматування тексту, а й роботу зі складними елементами контенту. Зокрема, очікується можливість створювати таблиці, оформлювати їх стилями; вставляти зображення та керувати їх властивостями (розмір, обтікання текстом, підписи); додавати посилання (як внутрішні гіперпосилання по документу, так і зовнішні URL); вставляти мультимедійні об’єкти (аудіо, відео) або інтерактивні віджети;

– система має генерувати під цим візуальним представленням правильний код (HTML-розмітку або іншу форму даних). Вимога до редактора – щоб він не створював зайвого або некоректного коду, який може призвести до проблем при відображенні на інших платформа чи несумісності з форматом. Тому якісні WYSIWYG-редактори (наприклад, CKEditor, TinyMCE для веб) особливу увагу приділяють відповідності HTML/CSS стандартам:

наприклад, автоматично закривають теги, не вставляють застарілих елементів, дозволяють очистити форматування, вставлене з Word, тощо;

– підтримка спільної роботи і контролю версій. Сучасні редактори часто інтегрують функції спільного редагування: декілька користувачів можуть одночасно працювати над виданням або документом, бачачи правки один одного (як у Google Docs чи в корпоративних CMS). Для систем електронних видань це теж стає актуальним, особливо коли над контентом працює колектив авторів і редакторів.

2.6 Сучасні тенденції у розробці інтерфейсів редагування та публікації

Область цифрового видавництва динамічно розвивається, і в останні роки з’явилися нові підходи до організації редакторських інтерфейсів та систем публікації електронних документів. Найбільш помітні сучасні тенденції включають блокову модель контенту, використання *Markdown*-розмітки та широке застосування шаблонів і готових компонентів.

Блокова модель редакторів. Один з актуальних трендів – перехід до блокової (модульної) моделі представлення контенту в редакторах. Це означає, що документ розглядається як набір окремих блоків різного типу (абзац тексту, заголовок, зображення, відео, список тощо), які можна незалежно вставляти, налаштовувати та переміщувати. Яскравим прикладом є новий редактор Gutenberg у системі WordPress, який реалізує концепцію блоків замість суцільного текстового поля: користувач конструює сторінку, додаючи потрібні блоки й налаштовуючи їх вміст [6]. Блоковий підхід надає більшу гнучкість у верстці електронних видань, наближаючи її до принципу *LEGO-конструктора*, коли з стандартних компонентів можна швидко скласти унікальний макет. Для видавничих систем це означає можливість легко вставляти інтерактивні елементи, галереї, опитування тощо як окремі модулі. Крім того, блокова модель спрощує адаптивність – кожен блок можна автоматично підлаштувати під різні розміри екрана. Тенденція до блокових редакторів простежується не лише в веб-розробці

(Gutenberg, конструктори сайтів типу Wix чи Squarespace), а й у спеціалізованих системах керування контентом для електронних видань. В перспективі блоки можуть стати універсальними *будівельними елементами* контенту, якими оперують і автори, і дизайнери без необхідності занурюватися в код.

Легковагові мови розмітки (Markdown). Паралельно з розвитком візуальних редакторів відроджується інтерес до простих мов розмітки, зокрема Markdown. Markdown – це легка текстова мова, що дозволяє за допомогою мінімального набору символів форматувати текст (розділи, виділення жирним чи курсивом, списки тощо). Спочатку Markdown був популярний серед програмістів і авторів технічної документації завдяки своїй простоті та можливості зберігати документацію в чистому тексті (наприклад, у репозиторіях на GitHub). Сьогодні ж Markdown дедалі частіше інтегрується в редакторські інтерфейси: багато блог-платформ, документаційних порталів та навіть редакторів з графічним інтерфейсом дозволяють вводити контент у вигляді Markdown-розмітки. Переваги такого підходу – мінімалізм і фокус на тексті: автор може зосередитися на змісті, не відволікаючись на кнопки форматування, а результат (відформатований документ) генерується автоматично. Для досвідчених користувачів Markdown підвищує швидкість роботи: розмітка додається відразу при написанні, без перемикання на мишку і панелі інструментів. З іншого боку, для новачків Markdown може бути не таким інтуїтивним, як WYSIWYG, тому сучасні системи шукають компроміс. Зокрема, з’являються гібридні редактори, які поєднують Markdown і WYSIWYG: користувач може вводити текст з Markdown-символами, а редактор одразу відображає форматований вигляд (або навпаки – дозволяє перемкнутися між режимами) [7,8]. Такий підхід вже реалізовано у багатьох інструментах для розробників (наприклад, редактори документації, що підтримують одночасно Markdown і живий перегляд). Загалом, тенденція використання Markdown відображає запит на *простоту і переносність контенту*: документи у Markdown легко зберігати в системах контролю версій, конвертувати в різні формати (HTML, PDF тощо) за допомогою автоматичних інструментів, вони менше “прив’язані” до конкретної платформи редагування.

Шаблони і повторно використовані компоненти. Ще одна сучасна тенденція – максимальне використання готових шаблонів та компонентів при створенні електронних публікацій. Складні видавничі платформи тепер пропонують бібліотеки шаблонів сторінок, стилів і навіть контентних блоків, які можна швидко застосувати. Наприклад, у згаданому блоковому редакторі є набір *шаблонів блоків* (block patterns) – попередньо оформлених комбінацій блоків для типових елементів сторінки (обкладинка з заголовком, галерея з підписами, двоколонковий текст з зображенням тощо). Автору достатньо вибрати потрібний шаблон, щоб отримати професійно виглядну структуру і потім лише замінити текст та ілюстрації. У системах керування контентом для електронних журналів і книг також з'являються пакети шаблонів оформлення, розроблені дизайнерами з урахуванням специфіки жанру (науковий журнал, навчальний підручник, бізнес-презентація тощо). Використання таких шаблонів підвищує продуктивність і уніфікацію: різні випуски журналу чи серії книг матимуть єдиний стиль, заданий шаблоном, а час на верстку нового випуску зменшується. Звісно, шаблони не обмежують творчість – їх можна налаштовувати, але вони дають базову структуру, від якої відштовхується автор. Тенденція до шаблонізації пов'язана і з розвитком headless-CMS (безголових систем керування контентом), де контент відокремлений від представлення: контент створюється у структурованому вигляді, а різні «голови» (веб-сайт, мобільний додаток, PDF-генератор) застосовують до нього свої шаблони оформлення. Це дає змогу публікувати один і той самий вміст у різних формах, підтримуючи цілісний стиль у кожній з них.

3 МЕТОДИКА СТВОРЕННЯ ІНСТРУМЕНТУ

3.1 Вимоги до інструменту та кінцевого продукту

Метою роботи є створення автономного онлайн-інструмента для перегляду, конвертації та редагування електронних видань, що поєднує завантаження PDF-документів, їх перетворення в упорядкований текстовий формат із можливістю подальшого формування адаптивного HTML-ресурсу. Передбачається, що користувачі отримають інтерфейс із інтуїтивними засобами навігації за сторінками, пошуку всередині документа, а також інструменти редагування вмісту без втрати оригінальної структури. До основних завдань розробки належать:

– забезпечити стабільний прийом навіть великих PDF-файлів із можливістю роботи з ними в потоці, щоб уникнути надмірного споживання пам’яті;

– створити швидкий і коректний сервіс перетворення, здатний зберігати ієрархію заголовків, списків, таблиць та зображень, – так, щоб кінцевий текст відповідав вихідному документу;

– впровадити механізм тимчасового збереження результатів конвертації для скорочення часу повторних запитів;

– розробити систему контролю доступу для користувачів із можливістю зберігати свої налаштування й шаблони;

– створити зручний веб-інтерфейс із розділами «Завантажити», «Переглянути», «Редагувати», «Зберегти», де можна легко переходити між режимами роботи з документом;

– забезпечити надійну взаємодію між складовими системи та можливість легко додавати нові функції й розширення без порушення існуючої логіки;

Очікується, що кінцевий продукт працюватиме незалежно від платформи клієнта, надаючи веб-додаток, у якому можна переглядати PDF-файл у режимі реального часу, при необхідності коригувати текст і динамічно відображати оновлений HTML-відповідник. Загальна архітектура інструмента повинна передбачати чітке розмежування обробки файлів на стороні сервера й відображення на стороні клієнта, а також гнучке масштабування у разі зростання кількості користувачів.

3.2 Методика вирішення проблеми

У проєкті було прийнято рішення розділити відповідальність між кількома мікросервісами, кожен із яких використовує найбільш придатну для своєї задачі технологію. Такий підхід дозволяє ефективно масштабувати систему та в разі потреби окремо оновлювати або замінювати технологічні компоненти.

Зокрема, для обробки великих PDF-файлів було вирішено використовувати мову програмування Scala із застосуванням технології Akka HTTP. Обрання саме цієї технології зумовлене її високими можливостями для паралельної та асинхронної обробки запитів. Scala характеризується потужною типізацією, що дозволяє мінімізувати кількість помилок вже на етапі компіляції, а Akka HTTP забезпечує надзвичайно гнучке управління потоками інформації завдяки архітектурі акторів та потоків (Streams. Використання функціонального стилю в Scala забезпечує прозорість і надійність коду, значно полегшуючи його підтримку та модифікацію в майбутньому.

Наступним важливим етапом у розробці стала реалізація механізму конвертації PDF-документів у Markdown-формат. Для цього було використано мову програмування Python разом із сучасним веб-фреймворком FastAPI та спеціалізованою бібліотекою marker-pdf. FastAPI є одним із найпопулярніших інструментів для створення REST-API у світі Python завдяки своїй простоті у використанні, високій продуктивності та можливостям швидкої розробки.

Головною перевагою FastAPI є декларативність, що дозволяє швидко й наочно описати REST-ендпоінти, які автоматично інтегруються з OpenAPI та JSON-схемами, суттєво спрощуючи процес тестування та інтеграції із зовнішніми сервісами.

Для зберігання метаданих користувачів, їх шаблонів та забезпечення авторизації було використано мову програмування Kotlin у поєднанні з фреймворком Ktor. Kotlin зарекомендував себе як безпечна й сучасна мова програмування, яка дозволяє ефективно уникати багатьох поширених помилок завдяки строгій системі типів та ретельному контролю над nullable-значеннями. Фреймворк Ktor, у свою чергу, забезпечує потужний та гнучкий механізм розробки серверних застосунків, особливо у частині авторизації та роботи з JWT-токенами. Використання вбудованих плагінів Authentication і JwtIssuer суттєво спрощує розробку та обслуговування системи авторизації, дозволяючи точно налаштувати різні рівні доступу для користувачів.

Фронтенд-частина проєкту була реалізована з використанням JavaScript-бібліотеки React разом із Vite. React був обраний через свою високу продуктивність, модульність і можливість створювати складні інтерактивні інтерфейси з чітко визначеними компонентами. Vite забезпечує надзвичайно швидкий процес розробки завдяки миттєвій заміні модулів (hot module replacement), а його ефективна обробка коду дозволяє значно прискорити завантаження додатку навіть під час активних змін у коді. Для стилізації було використано Tailwind CSS, що суттєво прискорює процес розробки завдяки утилітарному підходу.

3.3 Інструменти тестування та розгортання

Для забезпечення відтворюваності середовища та оперативної розгортки було обрано використання Docker Compose як основного інструменту контейнеризації та оркестрації. У межах одного файлу docker-compose.yml визначено чотири ключові служби, кожна з яких відповідає за окремий

функціональний модуль системи: сервіс, котрий працює із завантаженням і попередньою обробкою PDF-файлів, сервіс конвертації у текстовий формат, сервіс авторизації й управління даними та фронтенд-інтерфейс. Для кожного з цих сервісів створено багатостадійний Dockerfile, що дозволяє розбити процес побудови контейнера на кілька логічних етапів.

Автоматизацію процесів розробки, тестування й розгортання забезпечує GitHub Actions. Налаштований CI/CD-процес складається з послідовно виконуваних кроків, що охоплюють перевірку якості коду, запуск юніт- та інтеграційних тестів, збирання готових образів і, за потреби, автоматичне розгортання в QA та staging-середовищах.

3.4 Інструменти поширення

Для розгортання проєкту в хмарному середовищі було обрано платформу DigitalOcean завдяки її простоті налаштування, прозорій тарифній політиці та широкій підтримці контейнеризованих застосунків. У рамках DigitalOcean використано два Droplet’и: один для продакшн-оточення, інший – для staging, що дозволяє паралельно тестувати нові релізи без ризику вплинути на роботу користувачів. Кожен Droplet оснащено 2 vCPU та 4 ГБ оперативної пам’яті – достатньо для обробки середньої кількості одночасних запитів користувачів.

4 СТВОРЕННЯ АВТОНОМНОГО ІНСТРУМЕНТУ

4.1 Визначення структури проекту

Інтерфейс веб-інструмента організовано як односторінковий додаток (SPA) із чіткою навігацією за допомогою клієнтського роутера React Router. Усі ключові розділи доступні за фіксованими шляхами, кожен із яких відповідає за окремий етап роботи з документом – від завантаження до фінальної конвертації.

1. Лендінг.

Головна сторінка виконує роль лендінга та початкового екрану: тут розміщено загальний опис функцій сервісу, кнопки переходу до реєстрації/авторизації, а також інтерфейс для швидкого завантаження PDF.

2. Роботи.

На маршруті відображається перелік усіх завантажених користувачем документів із картками, що містять назву та дату останнього оновлення. Кожна картка дозволяє швидко повторно завантажити PDF через drag-and-drop у межах сторінки. Звідси доступний перехід до перегляду конкретної роботи або до списку шаблонів.

3. Перегляд і спроба редагування.

Сторінка реалізує перегляд PDF з підтримкою масштабування, пагінації та пошуку.

4. Робота з Markdown.

В усіх режимах перегляду та редагування за допомогою кнопки «Edit Markdown» викликається модальний діалог. Після підтвердження діалогу оновлений Markdown надсилається бекенду для збереження та повторної генерації HTML-видання.

5. Список шаблонів.

На маршруті користувачу пропонується перелік готових шаблонів оформлення електронного видання, кожен із яких описується назвою, коротким описом і прев’ю. Вибір шаблона запускає процес генерації HTML за поточним

Markdown із подальшим відображенням результату на сторінці фінальної конвертації.

6. Авторизація.

Маршрут реалізує вхід користувача через форму з полями email і пароль. Після успішної валідації за допомогою JWT-токена користувач автоматично перенаправляється на сторінку з роботами, а токен зберігається в куках для подальших запитів.

7. Сторінка помилки.

Для некоректних або неіснуючих маршрутів передбачено компонент NotFound, який відображається при переході за будь-яким шляхом, що не збігається з наведеними. Сторінка містить повідомлення про помилку та кнопки повернення на головний екран.

8. Документація.

Клієнтська сторінка документації містить вбудовану Swagger-UI документацію для FastAPI-сервісу та опис REST-ендпоінтів Akka і Ktor. Це дозволяє розробникам швидко перевіряти контракти API без сторонніх інструментів.

9. Сторінка результату конвертації.

Після вибору шаблона або підтвердження змін Markdown користувача відбувається перенаправлення на окрему сторінку результату, де відображається фінальний HTML-макет із застосованим шаблоном. Тут також доступна кнопка завантаження фінального PDF або HTML-файлу.

4.2 Визначення дизайну инструменту

У першу чергу перед початком безпосередньої розробки інтерфейсу було проведено ґрунтовний аналіз технічного завдання та вимог до користувацького досвіду. Враховуючи, що проєкт поєднує кілька функціональних модулів – від завантаження PDF до інтерактивного редагування Markdown та генерації HTML – було важливо розробити універсальний візуальний каркас, здатний вмістити різні робочі сценарії без втрати цілісності стилю.

Наступний крок – реструктуризація проєкту на логічні блоки та оновлення навігаційної схеми – було оформлено у вигляді діаграми (рис. 4.1), де чітко зазначені основні маршрути та зв’язки між ними. Бізнес логіка описана за посиланням [12]. З урахуванням узгодженої архітектури виконано повний перехід UI на shadcn/ui – сучасну бібліотеку компонентів на основі Tailwind CSS. Це дало змогу централізовано керувати стилями, легко впроваджувати темізацію та одночасно уникнути дублювання коду. У процесі інтеграції shadcn/ui відбувся рефакторинг кольорової схеми: замість жорстко закодованих значень було переведено всі ключові кольори у CSS Custom Properties (рис. 4.2).

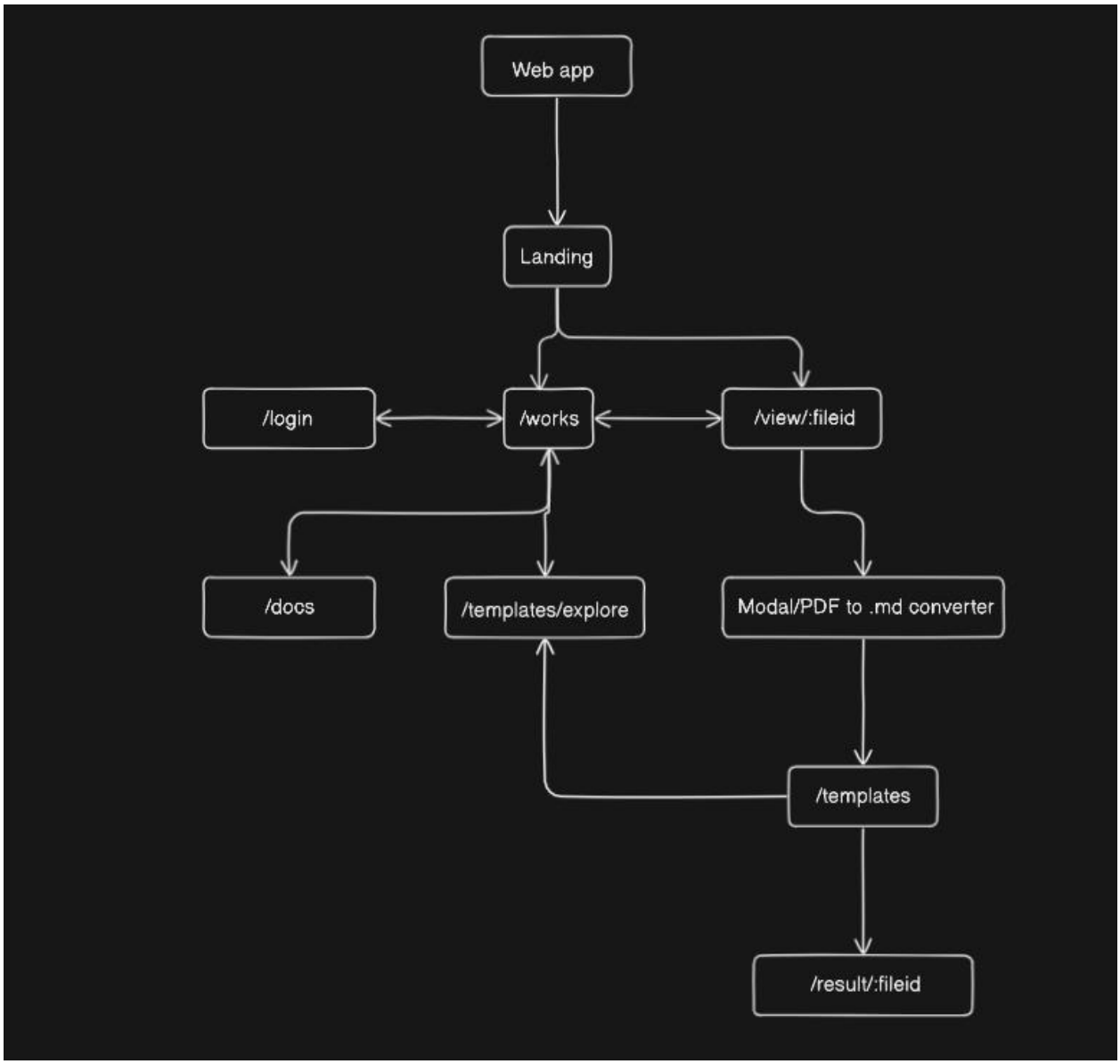

Рисунок 4.1 – Результат оновлення навігаційної схеми

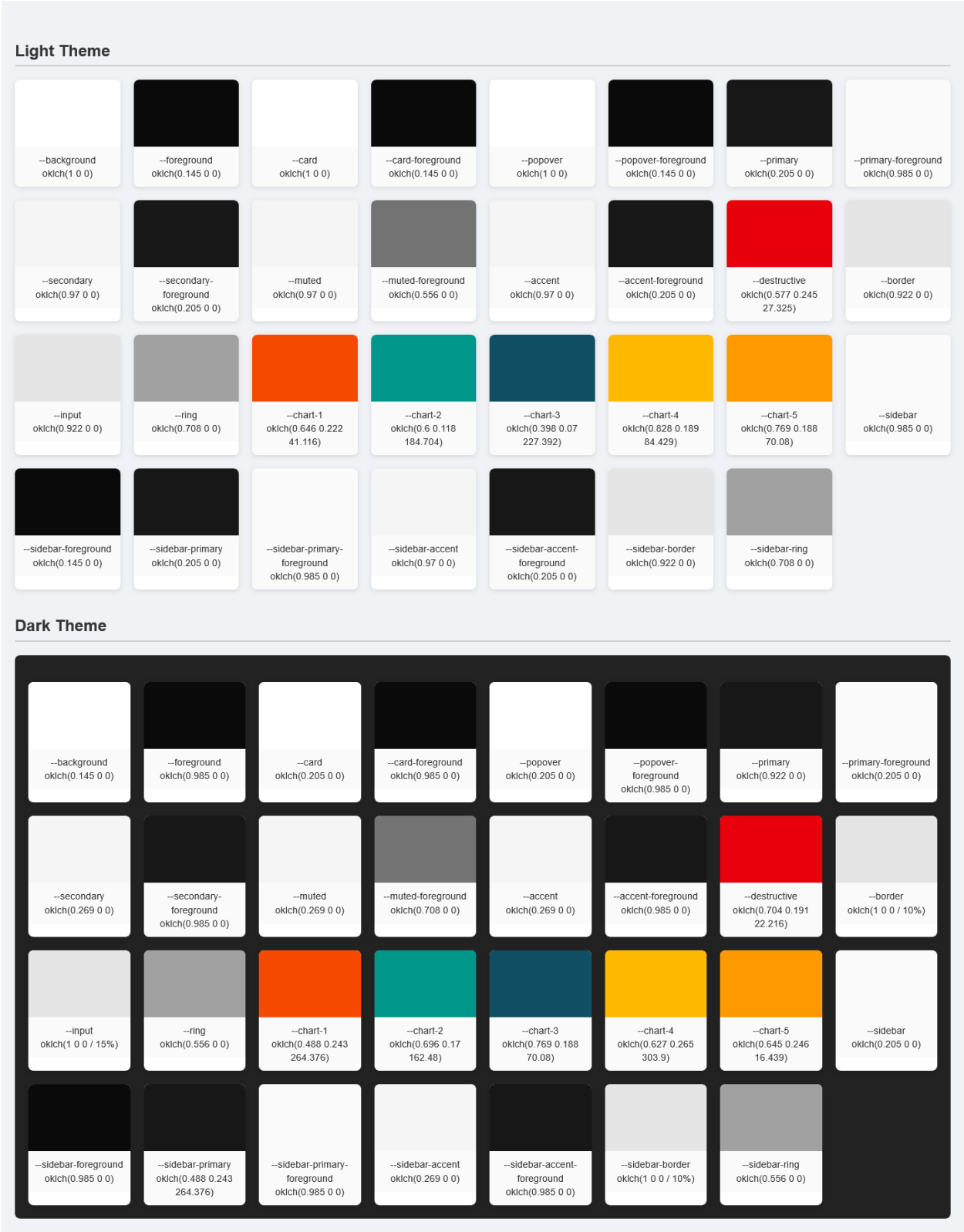

Рисунок 4.2 – Результат рефакторингу кольорової схеми

Код (розташовано у додатку А) демонструє приклад визначення змінних, які використовуються у Tailwind-конфігурації для створення утиліт bg-background, text-primary, bg-card тощо.

Далі відбувся вибір модульної сітки: з огляду на вимоги адаптивності та швидкого вирівнювання блоків обрано 12 колонну структуру з однаковими gutter по 16 px (рис. Б.1, рис. Б.2, рис. Б.3, додаток Б). Така сітка універсальна для десктопів та планшетів, дозволяє легко центрувати контент і мінімізує ризик порушення візуальної ієрархії при зміні розміру вікна браузера.

Наступним значущим етапом було розроблення логотипу проєкту у мінімалістичному стилі.

Назву логотипу використано як MustachePDF. Це не просто «ще один конвертер» – це інструмент, який надає документам свою індивідуальність, дозволяючи переглядати та редагувати їх у сучасному веб-середовищі. Телепортація у світ шляп і вусів підкреслює: із MustachePDF користувач отримує не тільки результат, а й враження, що важливо у часи, коли інтерфейси стали уніфікованими, а конкуренція за увагу – надто високою (рис. 4.3).

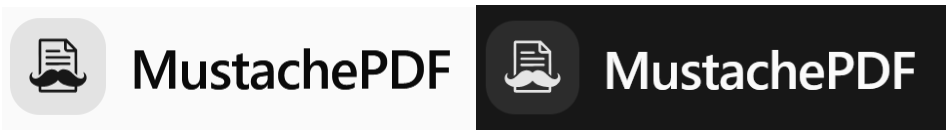

Рисунок 4.3 – Логотип з назвою створеного продукту

Відповідно до створеного логотипу та структури проекту було визначено сторінки подальшого сайдбару. Під логоптипом повинно знаходитися меню авторизації за створеною моделлю користувача. Тому користувачеві було додано вуса. Структура самого тіла сайдбару розроблено як «Your Projects», «Documentation», «Explore Templates» та візуалізовано на (рис. 4.4).

У процесі реалізації розділу «Your Works» було створено окрему сторінку, яка об’єднує функціонал завантаження, попереднього перегляду та подальшої обробки власних документів користувача. З самого початку було визначено, що цей інтерфейс має забезпечувати інтуїтивний досвід: користувач бачить єдину форму для завантаження файлів, а після успішного збереження документа одразу потрапляє на сторінку з його коротким описом і можливістю викликати операцію конвертації чи перегляду (рис. 4.5-4.6)

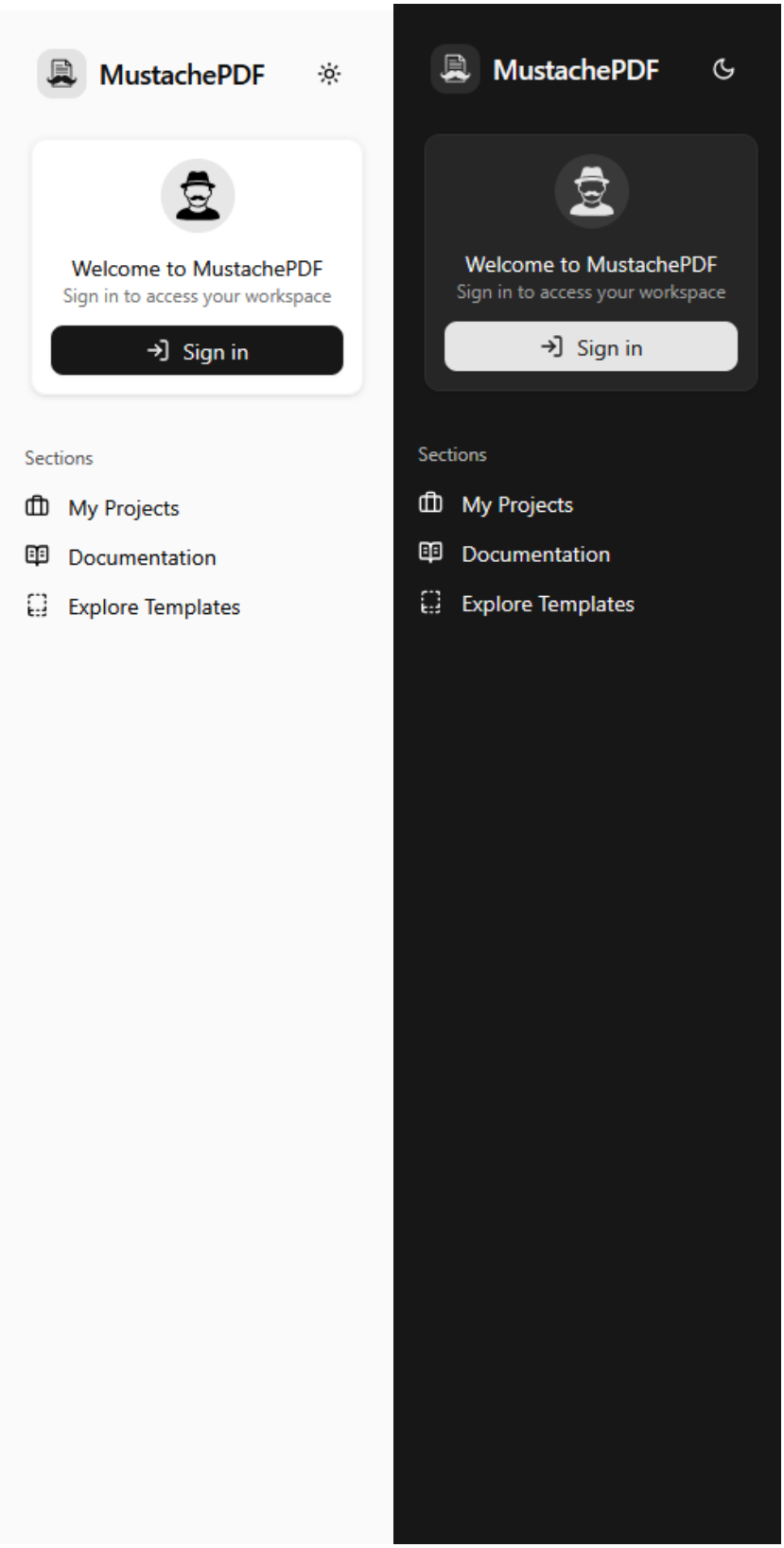

Рисунок 4.4 – Візуалізація логіки сайдбару

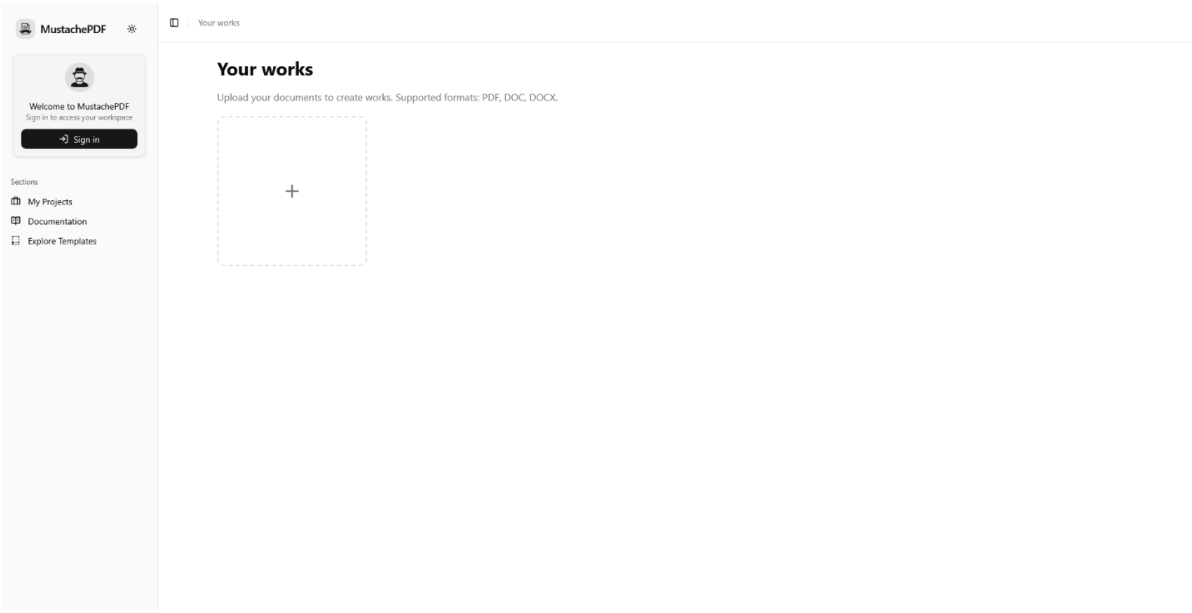

Рисунок 4.5 – Графічна складова сторінки «Your works»

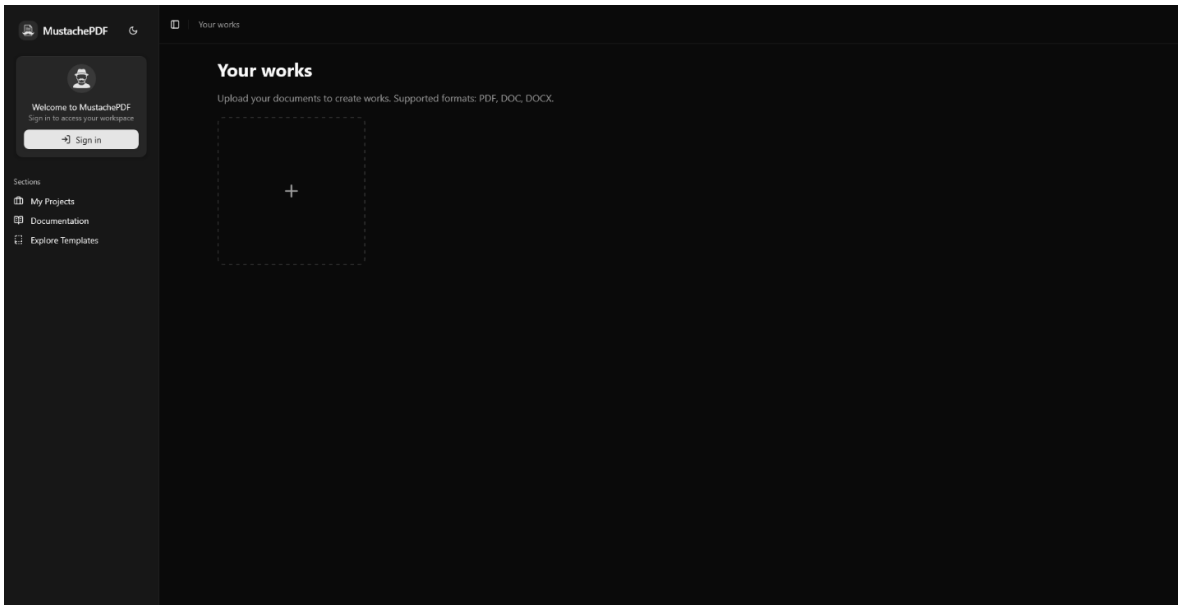

Рисунок 4.6 – Темна тема для сторінки «Your works»

У результаті успішного завершення запиту браузер отримував об’єкт із метаданими документа. За допомогою UI було розроблено процес завантаження файлу (рис. 4.7-4.8).

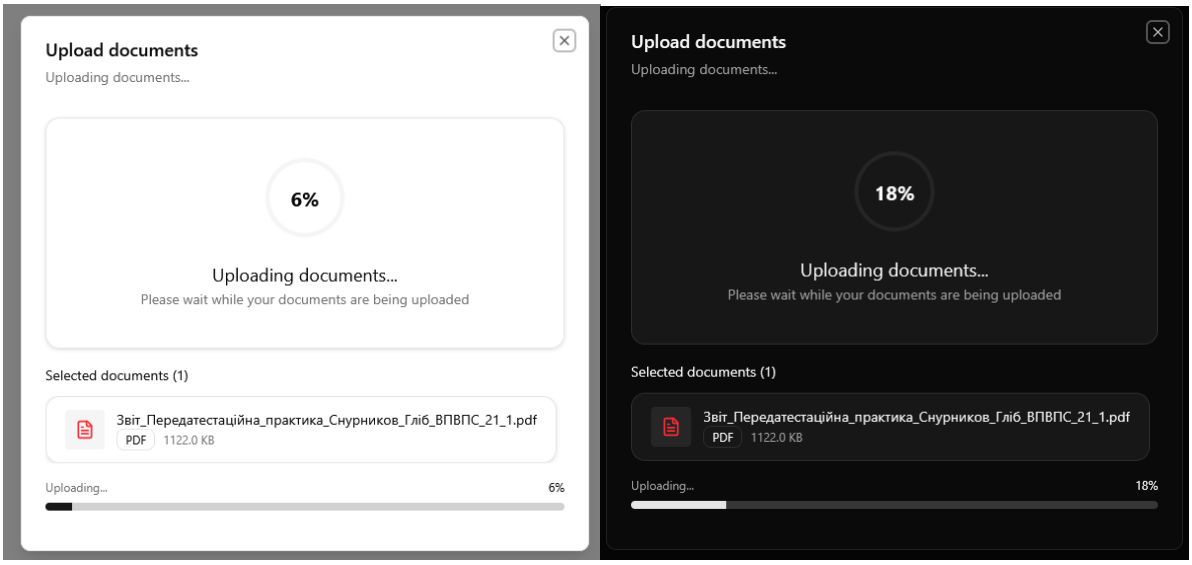

Рисунок 4.7 – Лоадер при створенні нового проекту

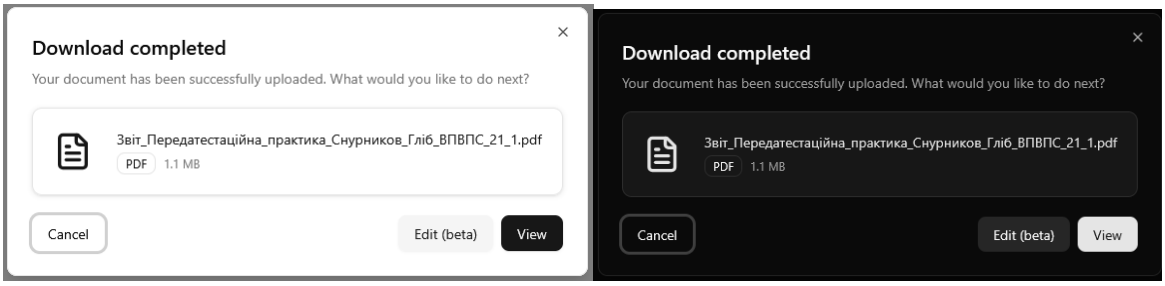

Рисунок 4.8 – Модальне вікно після завантаження файлу

Перший варіант, «Переглянути» (View), активував функцію відкриття PDF-файлу в спеціальному вбудованому переглядачі на основі бібліотеки pdfjs-dist (рис. 4.9-4.10). Під час перегляду користувач мав змогу масштабувати сторінки документа, переміщатися між ними за допомогою функції пагінації та здійснювати пошук за ключовими словами Завдяки асинхронному завантаженню і рендеру сторінок, навіть великі документи відкривалися без затримок і без значного споживання оперативної пам’яті браузера користувача.

У результаті створено 3 слайди з переглядом та редагуванням конвертованого Markdown (рис. 4.11-4.13).

Рисунок 4.9 – Сторінка «View». Світла тема

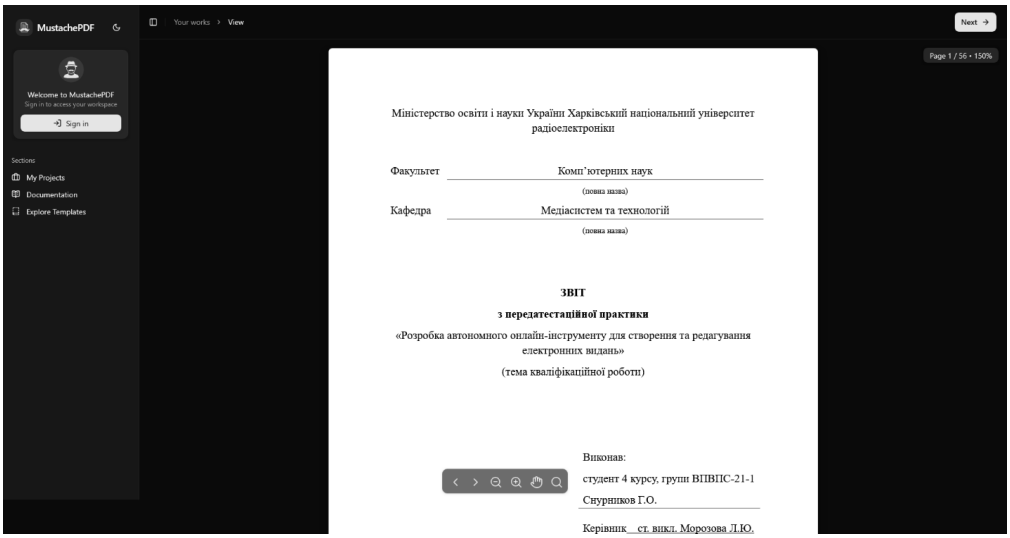

Рисунок 4.10 – Сторінка «View». Темна тема

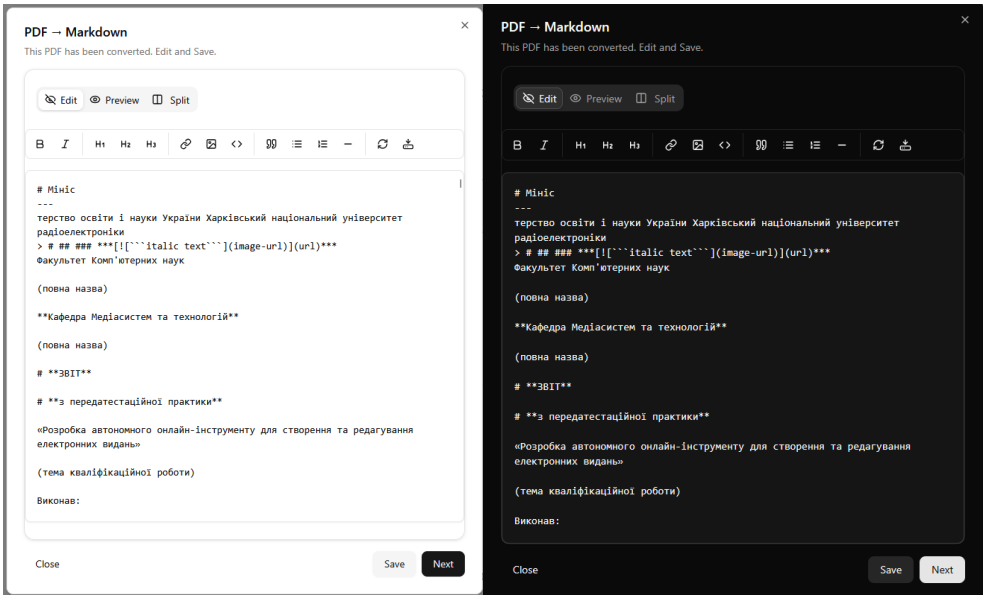

Рисунок 4.11 – Результат конвертування у режимі «Редагування»

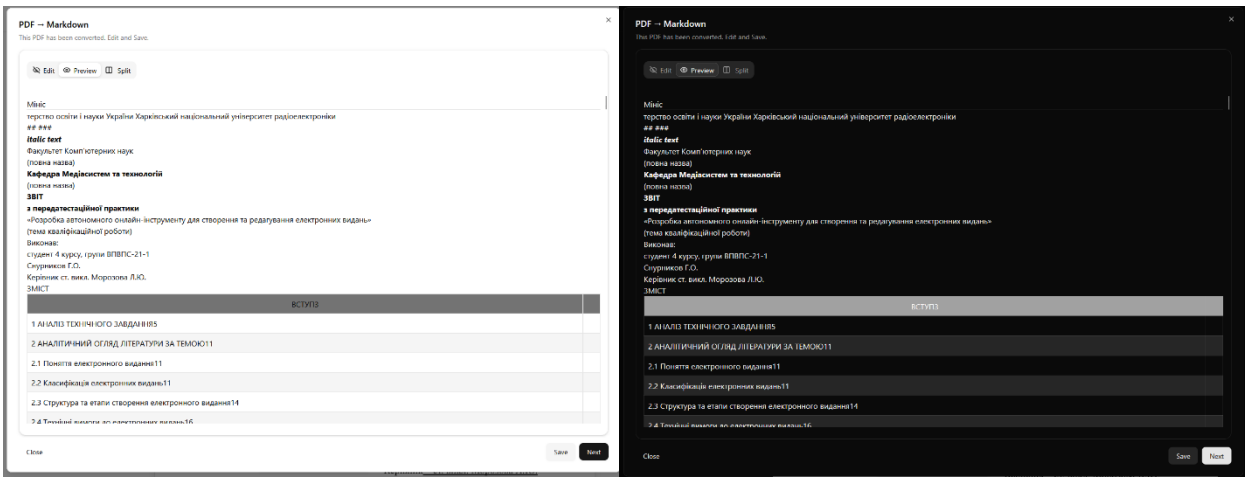

Рисунок 4.12 – Результат конвертування у режимі «Перегляд»

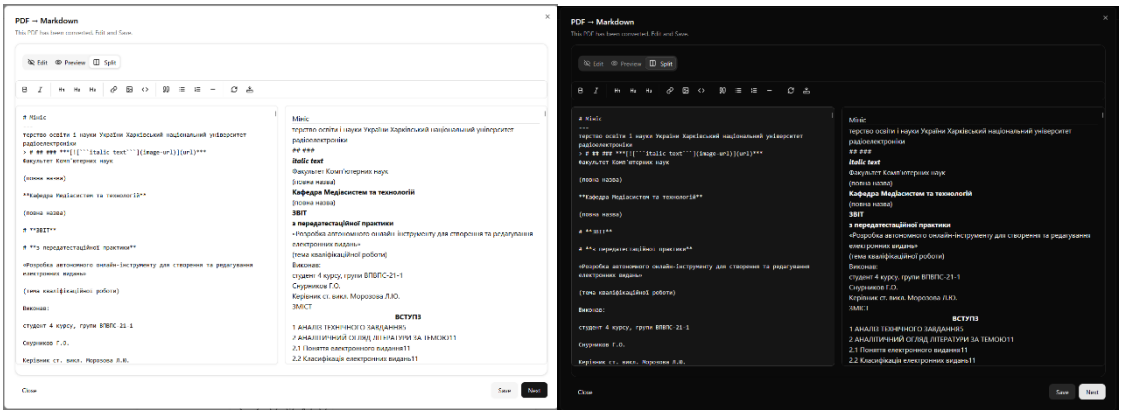

Рисунок 4.13 – Результат конвертування у режимі «Перегляд і Редагування»

4.3 Розробка дизайну шаблонів

У межах проєкту шаблони для генерації кінцевих HTML-видань були реалізовані за допомогою серверного шаблонізатора Nunjucks, а збережені у спеціальній моделі templates бази даних Supabase. Нижче наведено детальний опис підходу до роботи з шаблонами у цьому проєкті.

Першим етапом стало визначення структури таблиці templates у Supabase, яка відповідає паттерну «зберігання шаблонів приватно й версифіковано». У базі даних Supabase створено таблицю з такими полями (рис. 4.14):

Рисунок 4.14 – Структура templates у Supabase

Де:

– id (тип int8) – унікальний ідентифікатор шаблону;

– name (тип text) – коротка назва шаблону (наприклад, classic, modern, compact), за допомогою якої система виставляє пріоритетні стилі та вибір шаблона у фронтенді;

– description (тип text) – текстовий опис, що пояснює призначення та особливості конкретного шаблона (наприклад, «легкий макет для документів із мінімалістичним дизайном» або «корпоративний стиль із колонтитулами та логотипом»);

– body (тип text) – текст безпосередньо Nunjucks-шаблона, тобто вміст файлу з мітками ({{…}} та {%…%}), який використовуватиметься для рендерингу кінцевого документа;

– tags (тип text) – перелік ключових слів через кому, які описують стиль або особливості шаблона (наприклад, ["minimal", "print-ready"]), що дає змогу фільтрувати доступні шаблони у користувацькому інтерфейсі;

– created_at і updated_at (тип timestamptz) – часові позначки створення та останнього оновлення шаблона, необхідні для відстеження змін і версіювання.

В рамках системи шаблонів була реалізована класифікація за тематичними категоріями, що дозволяє користувачам швидко знаходити необхідний стиль оформлення залежно від призначення електронного видання. На даний момент доступні такі категорії:

– All – у цю категорію потрапляють усі наявні шаблони без фільтрації за іншим тегом;

– Blog – шаблони, орієнтовані на створення публікацій у форматі блогу, із відповідними стилями заголовків, абзаців та блоків зі списками чи цитатами;

– Marketing – шаблони, заточені під презентацію продуктів і послуг; містять яскраві акценти, місця для розміщення банерів і рекламних блоків;

– Minimal – шаблони з чистим макетом, невеликою кількістю графічних елементів і переважанням білих просторів, що підходять для документів із акцентом на текст;

– Docs – шаблони для технічних або довідкових матеріалів; передбачають розширені блоки для списків, схематичних зображень і таблиць;

– E-Commerce – шаблони для інтернет-каталогів та онлайн-магазинів, містять компоненти для відображення товарних карток, цін та описів, а також кнопки «Додати до кошика»;

– Email – шаблони для створення HTML-листів, із врахованою сумісністю відразу з основними поштовими клієнтами; оптимізовані під адаптивне відображення у різних поштових програмах;

– Portfolio – шаблони, призначені для демонстрації авторських робіт, галерей зображень та описів проектів; містять сітки для візуального представлення контенту;

– Admin – шаблони для внутрішніх систем керування, із акцентом на таблиці, панелі навігації й елементи інтерфейсу для редагування даних.

Під меню категорій розташовується сітка з мініатюрами шаблонів, кожна з яких містить назву, опис і зразки вигляду (рис. 4.15-4.16). При наведенні курсора на мініатюру з’являється кнопка: «Preview» (Попередній перегляд). Інтерфейс попереднього перегляду відкриває модальне вікно, у якому демонструється приклад вже згенерованого документа (рис. 4.17).

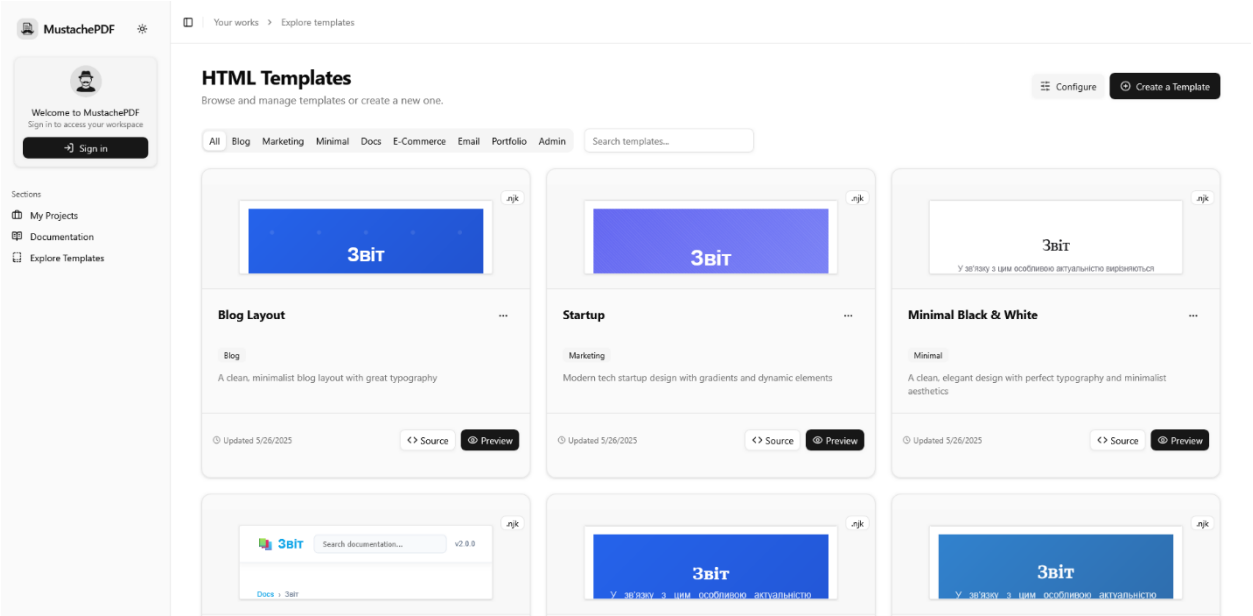

Рисунок 4.15 – Сторінка “Templates”. Світла тема

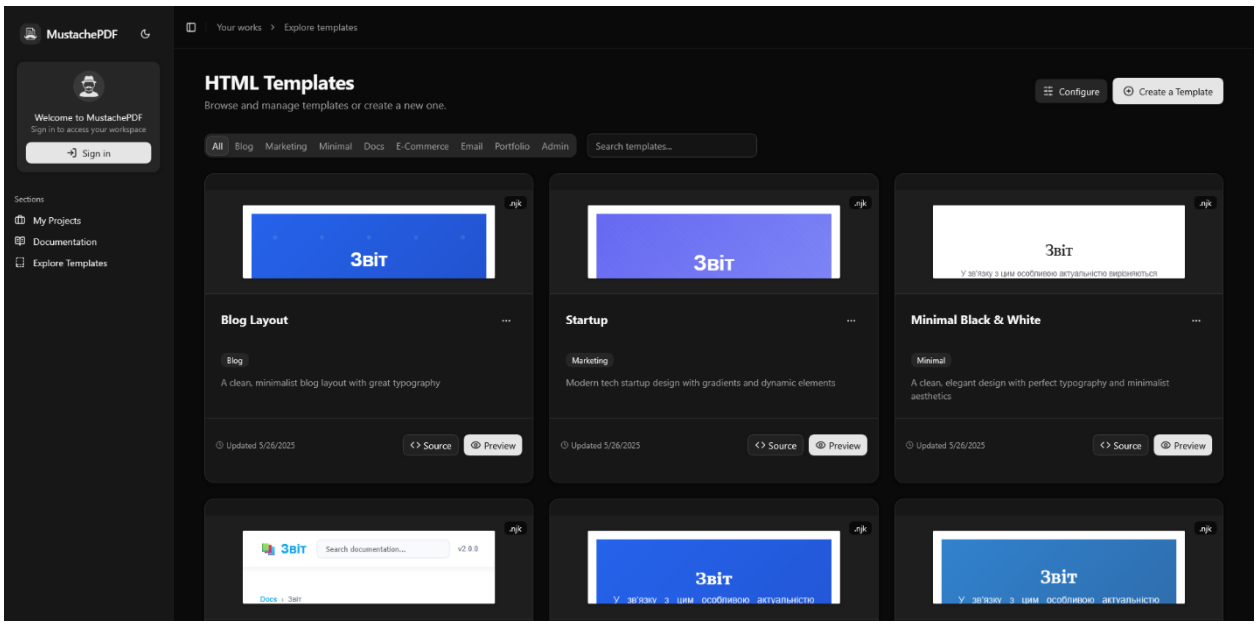

Рисунок 4.16 – Сторінка “Templates”. Темна тема

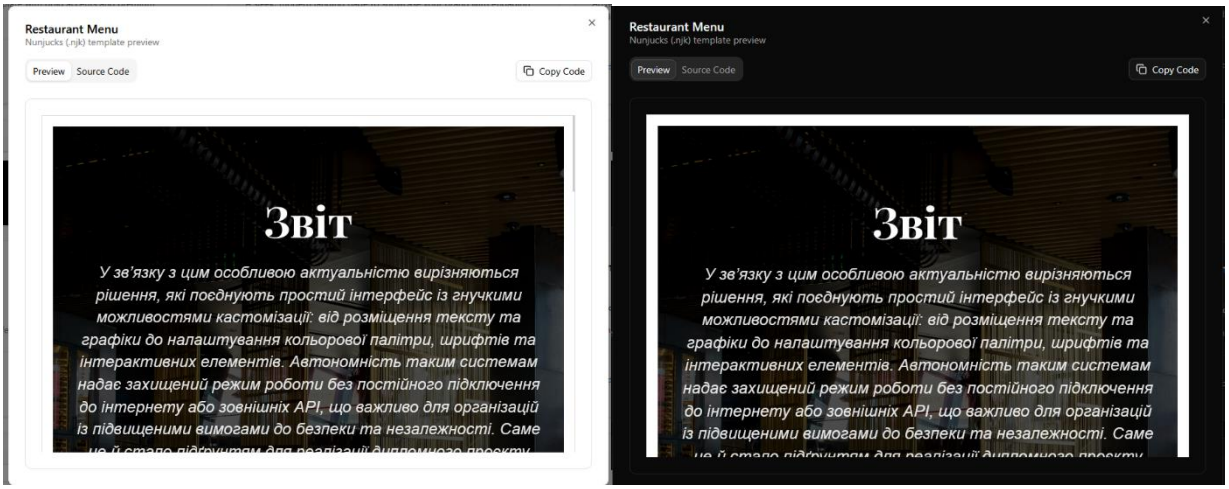

Рисунок 4.17 – Інтерфейс попереднього перегляду

4.4 Тестування до розробленного інструменту

У процесі розробки фронтенд-частини автономного онлайн-інструмента було прийнято рішення забезпечити високий рівень якості коду та стабільності роботи за допомогою комплексної системи тестування. Адже коректна робота компонентів інтерфейсу безпосередньо впливає на загальну надійність застосунку, особливо в умовах асинхронної взаємодії з мікросервісами та обробки великих PDF-файлів. Для цього використовувався сучасний тестовий фреймворк Vitest, а також його графічний інтерфейс Vitest UI

4.5 Встановлення на сервер

Публікація готового інструмента на віддаленому сервері відбувалася в кілька взаємопов’язаних кроків, кожен з яких був спрямований на забезпечення безпеки, відтворюваності середовища та можливості масштабування. Нижче наведено опис основних етапів, які дозволили розгорнути й перейти в експлуатацію мікросервісне середовище з використанням Docker Compose та Nginx. Було здійснено підключення по ssh (рис. 4.18). На сервері використовувалася OS з Ubuntu 24.04.2 LTS.

За допомогою можливостей Git було здійснено клонування репозиторію та зроблено корретну інсталяцію за мануалом у README.md (рис. 4.19).

```
Disto@egor ~ > ssh glib@20.160.234.209
glib@20.160.234.209's password:
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.11.0-1014-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sat Jun  7 17:09:01 UTC 2025

  System load:  0.0               Processes:             132
  Usage of /:   55.5% of 28.02GB  Users logged in:       0
  Memory usage: 13%               IPv4 address for eth0: 10.0.0.4
  Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

1 device has a firmware upgrade available.
Run `fwupdmgr get-upgrades` for more information.
```

Рисунок 4.18 – Піключення по ssh до серверу

```
glib@glib-test:~$ git clone https://github.com/casualmeow/dl-gen
Cloning into 'dl-gen'...
remote: Enumerating objects: 1947, done.
remote: Counting objects: 100% (254/254), done.
remote: Compressing objects: 100% (169/169), done.
remote: Total 1947 (delta 78), reused 191 (delta 61), pack-reused 1693 (from 1)
Receiving objects: 100% (1947/1947), 7.94 MiB | 31.63 MiB/s, done.
Resolving deltas: 100% (775/775), done.
```

```
glib@glib-test:~$ cd dl-gen
glib@glib-test:~/dl-gen$ docker compose up --build
Compose can now delegate builds to bake for better performance.
 To do so, set COMPOSE_BAKE=true.
[+] Building 3.1s (64/64) FINISHED                                  docker:default
 => [backendv3 internal] load build definition from Dockerfile                0.1s
 => => transferring dockerfile: 402B                                          0.0s
 => [backendv2 internal] load build definition from Dockerfile                0.1s
 => => transferring dockerfile: 1.40kB                                        0.0s
 => [backendv1 internal] load build definition from Dockerfile                0.1s
 => => transferring dockerfile: 543B                                          0.0s
 => [backendv2 internal] load metadata for docker.io/library/python:3.12-slim 0.9s
```

Рисунок 4.19 – Запуск проекту на ssh сервері

4.6 Розповсюдження інструменту

Розповсюдження створеного сервісу здійснюється за допомогою кількох взаємно доповнювальних каналів, що забезпечують його доступність, гнучкість інсталяції та можливість адаптації під різні середовища.

По-перше, джерельний код розміщено у публічному репозиторії. У ньому зберігаються компоненти проекту, включно з піддиректоріями для кожного мікросервісу (наприклад, підсервіси для прийому файлів, конвертації, авторизації та фронтенд-застосунок), а також конфігураційні файли для контейнеризації та оркестрації.

По-друге, готові образи контейнерів публікуються у реєстрах образів. Завдяки цьому стає можливим швидко завантажувати попередньо зібрані контейнери без необхідності локальної побудови з нуля.

По-третє, для локального розгортання передбачено універсальний файл оркестрації, який описує всі необхідні сервіси, залежності між ними та підключення зовнішніх томів для збереження логів і кешованих даних.

5 ЕКОНОМІЧНА ЧАСТИНА

У результаті виконання було розроблено автономний онлайн-інструмент для прийому PDF-документів, їх конвертації у Markdown і подальшої генерації адаптивного HTML-видання без втрати структури документа. З огляду на потребу освіти, малого та середнього бізнесу в безпечному й автономному інструменті, а також наявність альтернативних комерційних рішень з обмеженим функціоналом або дорогою ліцензією, було прийнято рішення про економічний аналіз доцільності впровадження розробки.

Для оцінки економічної складової проєкту спочатку визначено основні складові витрат: заробітна плата розробників, єдиний соціальний внесок (ЄСВ), витрати на хостинг та інфраструктуру, адміністративні витрати, витрати на забезпечення якості та резерв на непередбачені витрати. Далі розраховано виробничу собівартість, прибуток при цільовому рівні рентабельності 30 %, ціну реалізації без і з урахуванням ПДВ.

Для економічного обґрунтування реалізації проєкту було визначено такі основні складові витрат:

– основна та додаткова заробітна плата задіяних фахівців;
– єдиний соціальний внесок (ЄСВ);
– витрати на електроенергію й утримання обладнання;
– оренда серверної інфраструктури (VPS, хмарні сервіси Redis, Supabase);
– загальновиробничі та адміністративні витрати;
– витрати на тестування й забезпечення якості;
– непередбачені витрати (резерв).

Нижче наведено деталізований розрахунок собівартості, а також формування ціни з урахуванням нормативу прибутковості та податкових відрахувань.

5.1 Собівартість розробки

Собівартість розробки охоплювала витрати на основну заробітну плату, додаткову заробітну плату, єдиний соціальний внесок, утримання устаткування й оренду серверів, витрати на електроенергію, а також загальновиробничі, адміністративні та витрати на збут.

Реалізація проєкту складалася з таких ключових етапів (усього 240 годин):

– налаштування середовища та оркестрація (Docker Compose, CI/CD) – 20 годин (DevOps-інженер);

– розробка Backend 1 (Scala /Akka HTTP) – 40 годин (Scala-розробник);

– розробка Backend 2 (Python /FastAPI) – 40 годин (Python-розробник);

– розробка Backend 3 (Kotlin /Ktor) – 40 годин (Kotlin-розробник);

– розробка Frontend (React + Vite) – 60 годин (Front-end-розробник);

– тестування й налаштування CI/CD (юніт- та інтеграційні тести) – 20 годин (Tester/DevOps-інженер).

Середньоринкові погодинні ставки було визначено на основі джерел:

DevOps-інженер: середня заробітна плата інженера DevOps / Cloud Engineer у 2025 р. оцінювалась приблизно в $5 000 на місяць (USD), що еквівалентно близько ₴180 000 (курс 36 UAH = 1 USD) [14]. Таким чином, погодинна ставка становила:

$$\frac{₴\,180\,000}{160\,\text{год/міс}} = ₴\,1\,125\ /\ \text{год.}$$

Scala-розробник: згідно з Intsurfing, середній рівень зарплат Scala-розробників (mid-level) в Україні коливається в діапазоні $2 500-$4 000 на місяць, у середньому $3 000 (USD), або близько ₴108 000 (курс 36 UAH=1 USD) [14,16]. Погодинна ставка:

$$\frac{₴\,79\,200}{160\,\text{год/міс}} = ₴\,495\ /\ \text{год.}$$

Python-розробник: за даними Mobilunity [15], середня нетто-зарплата Python-розробника (mid-level) в Україні у 2025 р. становила близько $2 200 на місяць (USD), або ₴79 200 (курс 36 UAH). Погодинна ставка:

$$\frac{₴\,79\,200}{160\ \text{год/міс}} = ₴\,495\ /\ \text{год.}$$

Kotlin-розробник: середня заробітна плата розробника (Software Developer) у Києві становить близько $3 114 на місяць (USD), що еквівалентно приблизно ₴112 104 (курс 36 UAH) [16]. Для Kotlin, як сучасної JVM-технології, можна прийняти цю ставку за основу. Погодинна ставка:

$$\frac{₴\,112\,104}{160\ \text{год/міс}} \approx ₴\,700\ /\ \text{год.}$$

Front-end-розробник: зазначається середня сукупна зарплата розробника (Developer) у Києві близько $2 400 на місяць (USD), або ₴86 400 (курс 36 UAH) [17]:

$$\frac{₴\,86\,400}{160\ \text{год/міс}} = ₴\,540\ /\ \text{год.}$$

На основі вищенаведених даних було створено таблицю 5.1 основних заробітних плат. Згідно з господарською практикою, додаткова заробітна плата становить 20 % від основної [37]:

$$\text{Додаткова ЗП} = 159\,260 \times 0{,}20 = 31\,852\ ₴.$$

Ставка ЄСВ – 22 % від суми основної та додаткової ЗП [38]:

$$\text{База ЄСВ} = 159\,260 + 31\,852 = 191\,112\ ₴,$$
$$\text{ЄСВ} = 191\,112 \times 0{,}22 = 42\,044{,}64\ ₴.$$

Таблиця 5.1 – Основна заробітна плата

Етап	Виконавець	Ставка, ₴/год	Тривалість, год	Заробітна плата, ₴
Налаштування середовища, CI/CD, Docker Compose	DevOps-інженер	1 125	40	45 000
Розробка Backend 1 (Scala/Akka HTTP)	Scala-розробник	675	40	27 000
Розробка Backend 2 (Python/FastAPI)	Python-розробник	495	40	19 800
Розробка Backend 3 (Kotlin/Ktor)	Kotlin-розробник	700	40	28 000
Розробка Frontend (React/Vite)	Front-end-розробник	540	60	32 400
Тестування, налаштування GitHub Actions	QA-інженер	353	20	7 060
Разом (основна ЗП)			240	159 260

Використовувалися дві власні робочі станції (Front-end, Back-end) із середньою потужністю 0,2 кВт/год та одна – для DevOps (0,25 кВт/год). Первісна вартість кожної робочої станції складала ₴50 000, термін експлуатації – 36 місяців (залишкова вартість – 0). Амортизаційні відрахування за один місяць розраховано так:

$$\text{Місячна амортизація однієї станції} = \frac{50\,000\,₴}{36\,\text{міс}} \approx 1\,389\,₴/\text{міс.}$$

Отже, для трьох станцій:

$$\text{Амортизаційні відрахування (3 станції)} = 1\,389 \times 3 = 4\,167\,₴.$$

Крім того, до амортизації додаються витрати на технічне обслуговування та ремонт (5 % від амортизації):

$$\text{Технічне обслуговування} = 4\,167 \times 0{,}05 = 208{,}35\,₴.$$

Таким чином:

$$\text{Утримання обладнання} = 4\,167 + 208{,}35 \approx 4\,375{,}35\ ₴.$$

Витрати на електроенергію. Розрахунок здійснено на основі фактичного споживання робочих станцій протягом 240 годин (по 0,2 кВт/год для двох станцій і 0,25 кВт/год для однієї):

$$\text{Загальна потужність} = (0{,}2 + 0{,}2 + 0{,}25)\ \text{кВт} = 0{,}65\ \text{кВт},$$

$$\text{Енергія за 240 годин} = 0{,}65 \times 240 = 156\ \text{кВт} \cdot \text{год},$$

$$\text{Вартість 1 кВт} \cdot \text{год} = \frac{4{,}50\ ₴}{1\ \text{кВт} \cdot \text{год}},$$

$$\text{Електроенергія} = 156 \times \frac{4{,}50\ ₴}{1\ \text{кВт} \cdot \text{год}} = 702\ ₴.$$

Під час розробки використовувалися як безкоштовні (VSCode, IntelliJ Community, React, Node.js, Docker, GitHub Free), так і комерційні інструменти:

– IntelliJ IDEA Ultimate (підписка JetBrains) – ₴1 500/міс. (на одного розробника);

– GitHub Actions (Pro-план) – ₴1 000/міс. (для достатньої кількості мінут CI/CD);

– Домени й SSL: реєстрація домену mustachepdf.com – ₴400/рік, SSL (ка Let's Encrypt) – безкоштовно.

API-ключі для AI/OCR: плата за хмарні сервіси OCR (Billing Model Free Tier) – безкоштовно; у разі перевищення безкоштовного ліміту – додаткова плата ≈ ₴2 000/міс. (резерв).

Отже, загальні витрати за місяць на ліцензії/підписки:

$$\text{Ліцензії} = 1\,500 + 1\,000 + \frac{400}{12} + 2\,000\ (\text{резерв}) = 4\,900\ ₴.$$

Для staging / QA-середовищ орендовано:

– VPS (DigitalOcean Droplet, 2 vCPU, 2 GB RAM) – ₴1 500/міс.;
– Redis Enterprise Basic Tier – ₴540/міс.;
– Supabase (free tier) – безкоштовно.

$$\text{Інфраструктура} = 1\,500 + 540 = 2\,040\ ₴.$$

Загальновиробничі витрати. Загальновиробничими витратами вважають витрати на електроенергію, утримання обладнання, оренду офісу (якщо б був), амортизацію ПЗ, внутрішні комунальні послуги тощо. Для спрощеного розрахунку прийнято, що загальновиробничі витрати становлять 10 % від суми: основної ЗП + утримання обладнання + електроенергія + інфраструктура + ліцензії:

$$S_{\text{виробничі}} = 159\,260 + 4\,375{,}35 + 702 + 2\,040 + 4\,900 = 171\,277{,}35\ ₴,$$

$$\text{Загальновиробничі} = 171\,277{,}35 \times 0{,}10 = 17\,127{,}74\ ₴.$$

Адміністративні витрати. Адміністративні витрати включають управлінські функції, бухгалтерію, канцелярію, послуги зв’язку. Прийнято, що обсяг цих витрат становить 5 % від суми: основної ЗП + утримання обладнання + ліцензії:

$$S_{\text{адміністр.}} = 159\,260 + 4\,375{,}35 + 4\,900 = 168\,535{,}35\ ₴,$$

$$\text{Адміністративні} = 168\,535{,}35 \times 0{,}05 = 8\,426{,}77\ ₴.$$

Маркетингові заходи для запуску SaaS-інструмента включають:

– онлайн-реклама (контекстна, соцмережі) – ₴10 000;
– підготовка презентаційних матеріалів (брошури, слайд-деки) – ₴5 000;
– участь у конференціях/іт-зустрічах – ₴7 000.

Загалом:

$$\text{Маркетинг} = 10\,000 + 5\,000 + 7\,000 = 22\,000\,₴.$$

Це окремо від «витрат на збут» як % від виробничої собівартості.

Також закладається загальна сума “витрат на збут” у розмірі 5 % від виробничої собівартості:

$$\text{Збутові} = 754\,206{,}64 \times 0{,}05 = 37\,710{,}33\,₴.$$

До виробничої собівартості входять:

– основна ЗП: 159 260 ₴;

– додаткова ЗП: 31 852 ₴;

– ЄСВ: 42 044,64 ₴;

– утримання обладнання: 4 375,35 ₴;

– амортизація ПЗ: (IntelliJ IDEA Ultimate + GitHub Actions) не виділена окремо в собівартості, бо взято в п. 6.1.6 як «ліцензії»;

– електроенергія: 702 ₴;

– інфраструктура (VPS + Redis): 2 040 ₴;

– ліцензії/підписки: 4 900 ₴;

– загальновиробничі витрати: 17 127,74 ₴;

– маркетингові витрати (онлайн advert & матеріали): 22 000 ₴;

– збутові витрати (5 % від виробничої): 37 710,33 ₴.

$$\text{Виробнича собівартість} = 159\,260 + 31\,852 + 42\,044{,}64 + 4\,375{,}35 + 702 + 2\,040 + 4\,900 + 17\,127{,}74 + 22\,000 + 37\,710{,}33 = 321\,011{,}06\,₴.$$

Повна собівартість розробки. До виробничої собівартості (321 011,06 ₴) додаються:

– адміністративні витрати (8 426,77 ₴);

– збутові витрати (37 710,33 ₴).

$$\text{Повна собівартість} = 321\,011{,}06 + 8\,426{,}77 + 37\,710{,}33 = 367\,148{,}16\ ₴.$$

5.2 Розрахунок ціни розробки

У IT-секторі України норматив рентабельності зазвичай становить 20-30%. Для створеного проекту взято консервативний показник – 25 % (для середніх компаній).

$$\text{Прибуток} = 367\,148{,}16 \times 0{,}25 = 91\,787{,}04\ ₴.$$

Ціна без ПДВ:

$$\text{Ціна без ПДВ} = 367\,148{,}16 + 91\,787{,}04 = 458\,935{,}20\ ₴.$$

Ставка ПДВ для IT-послуг в Україні – 20 %. Розрахунок ПДВ:

$$\text{ПДВ} = 458\,935{,}20 \times 0{,}20 = 91\,787{,}04\ ₴.$$

Ціна з ПДВ:

$$\text{Ціна з ПДВ} = 458\,935{,}20 + 91\,787{,}04 = 550\,722{,}24\ ₴.$$

Підсумовуючи, нижче створено табл. 5.2 після розрахунку ціни розробки.

Таблиця 5.2 – Витрати з урахуванням ціни розробки

№	Стаття витрат	Сума, ₴
1	Повна собівартість (без ПДВ)	367 148,16
2	Прибуток (25 %)	91 787,04
3	Ціна без ПДВ	458 935,20
4	ПДВ (20 %)	91 787,04
5	Ціна з ПДВ	550 722,24

5.3 Аналіз чутливості та ризиків

Коливання курсу UAH/USD. Якщо курс змінюється з 38 UAH/USD до 45 UAH/USD, то всі розрахунки зарплат (USD → UAH) слід перерахувати:

$$\text{DevOps} = 5\,000 \times 45 = 225\,000\,₴,\ \text{ставка} = \frac{225\,000}{160} = 1\,406{,}25\,₴/\text{год},$$

$$\text{Backend 1} = 3\,000 \times 45 = 135\,000\,₴,\ \text{ставка} = \frac{135\,000}{160} = 843{,}75\,₴/\text{год},$$

$$\text{Backend 2} = 2\,200 \times 45 = 99\,000\,₴,\ \text{ставка} = \frac{99\,000}{160} = 618{,}75\,₴/\text{год},$$

$$\text{Backend 3} = 3\,114 \times 45 = 140\,130\,₴,\ \text{ставка} = \frac{140\,130}{160} = 875{,}81\,₴/\text{год},$$

$$\text{Front-end} = 2\,400 \times 45 = 108\,000\,₴,\ \text{ставка} = \frac{108\,000}{160} = 675\,₴/\text{год},$$

$$\text{QA} = 56\,500\,₴,\ \text{ставка} = \frac{56\,500}{160} = 353{,}12\,₴/\text{год}.$$

Збільшення курсу призведе до зростання повної собівартості приблизно на 5 %-10 %. Затримка термінів розробки. Якщо кожен виконавець витрачає не 40 годин, а 48 годин (×1,2), то основна ЗП збільшується:

$$159\,260 \times 1{,}20 = 191\,112\,₴.$$

Це додасть до витрат ще приблизно 31 852 ₴ (20 % від цієї суми) на додаткову ЗП і 22 % ЄСВ, тобто загалом до 50 000 ₴ «зверху».

Зростання вартості хмарних сервісів (Redis, VPS). Якщо оренда VPS підвищиться з ₴1 500 до ₴2 000, а Redis – з ₴540 до ₴800, то інфраструктурні витрати зростуть на 760 ₴. Ураховуючи ці ризики, додано до розрахунку:

– встановилено резервну суму 5 % від повної собівартості для покриття непередбачених витрат (зміна курсу, додаткові години розробки, зростання цін на послуги);

– переглядануто ставки щоквартально та укладено довгострокові договори (зафіксовані ціни) на хмарні сервіси.

5.4 Підтримка та супровід після запуску

Після запуску SaaS-інструменту щомісячно планується виділяти ресурс на підтримку:

– 10 годин DevOps (моніторинг, оновлення ОС, резервне копіювання) → 10 год × 1 188 ₴/год ≈ 11 880 ₴/міс.;

– 10 годин Python-розробника (дрібні правки, усунення багів) → 10 год × 495 ₴/год = 4 950 ₴/міс.;

– 5 годин QA (регресійне тестування) → 5 год × 353 ₴/год = 1 765 ₴/міс.

Загалом:

$$\text{Підтримка} \approx 11\,880 + 4\,950 + 1\,765 = 18\,595\ ₴/\text{міс.}$$

5.5 Окупність проєкту

Окупність проекту включає у себе:

– очікувана вартість SaaS-ліцензії: ₴200 000/рік (≈ ₴16 667/міс.);

– щомісячні витрати на підтримку: ≈ ₴18 595;

– чистий грошовий потік (якщо 1 клієнт): ₴16 667 − ₴18 595 = −₴1 928 (від’ємний).

Для беззбиткового рівня щомісячного чистого потоку потрібно принаймні 2-3 постійних клієнти. Наприклад, при двох клієнтах:

$$\text{Дохід} = 2 \times 16\,667 = 33\,334\ ₴,$$

$$\text{Чистий потік} = 33\,334 - 18\,595 = 14\,739\ ₴/\text{міс.}$$

Місячні витрати на утримання (після запуску): ₴18 595.

Повернення інвестицій (ROI):

$$\text{Термін окупності} = \frac{\text{Повна собівартість}}{\text{Середньомісячний чистий потік}}.$$

За умови залучення трьох клієнтів (доходи 50 001 ₴/міс.):

$$\text{Чистий потік (3 клієнти)} = 50\,001 - 18\,595 = 31\,406\ ₴/\text{міс},$$

$$\text{Термін окупності} = \frac{367\,148{,}16}{31\,406} \approx 11{,}7\ \text{міс}.$$

Якщо ж до кінця першого півріччя залучити не менше 5 клієнтів, окупність настане раніше, приблизно за 7-8 місяців.

5.6 Підсумок до економічної частини

У результаті проведеного економічного аналізу розробки автономного онлайн-інструмента для створення та редагування електронних видань встановлено, що виробнича собівартість проєкту становить 321 011,06 ₴. До цієї суми включено заробітну плату всіх залучених фахівців (основну й додаткову), нарахування на фонд заробітної плати, амортизаційні відрахування та утримання обладнання, витрати на електроенергію, ліцензії й підписки, оренду хмарної інфраструктури, а також загальновиробничі, маркетингові та збутові витрати. З метою визначення повної собівартості до виробничих витрат додано адміністративні та збутові витрати, що за прийнятими нормативами складають 8 426,77 ₴ та 37 710,33 ₴ відповідно. Як наслідок, вартість розробки «під ключ» без урахування прибутку й податків дорівнює 367 148,16 ₴.

Згідно з обраним для даної галузі нормативом рентабельності 25 %, до повної собівартості додається сума прибутку в розмірі 91 787,04 ₴. Тому ціна без ПДВ складає 458 935,20 ₴. Після нарахування ПДВ (20 %) вартість проєкту для кінцевого замовника зростає до 550 722,24 ₴. Ця сума є орієнтирами для

подальших переговорів з потенційними клієнтами або інвесторами, оскільки вона включає всі необхідні ресурси для завершення розробки та початкового запуску продукту.

Окрему увагу приділено питанням окупності та підтримки інструмента після виходу на ринок. Для покриття щомісячних витрат на технічну підтримку (приблизно 18 595 ₴) необхідно не менше двох платних облікових записів із річною вартістю ліцензії 200 000 ₴ (що еквівалентно 16 667 ₴ на місяць). За умови залучення трьох клієнтів проєкт матиме позитивний щомісячний грошовий потік у розмірі близько 31 406 ₴. Це дозволить досягти повної окупності за приблизно 11-12 місяців з моменту запуску. Якщо ж вдасться залучити п’ять платних користувачів, термін окупності скоротиться до 7 - 8 місяців.

Враховуючи можливість коливань курсу UAH/USD, зростання ставок розробників та зміни цін на хмарні сервіси, доцільно передбачити резерв у розмірі 5 % від повної собівартості (приблизно 18 357 ₴). Це забезпечить захист від непередбачених фінансових втрат у разі затримок у виконанні проєкту чи додаткових витрат на оновлення обладнання й ПЗ. Щоквартальний перегляд умов оренди хмарної інфраструктури, підписок і ліцензій сприятиме мінімізації ризиків подорожчання витрат. Паралельно слід планувати витрати на технічне обслуговування обладнання та оновлення програмного забезпечення, оскільки своєчасне реагування на потреби хостингу й інструментальної бази допоможе уникнути додаткових простоїв і витрат. Повний розрахунок витрат створено у таблиці 5.3.

Таблиця 5.3 – Розрахунок витрат та ціни проєкту

№	Стаття витрат	Сума, ₴
1	Основна заробітна плата	159 260,00
2	Додаткова заробітна плата (20 %)	31 852,00
3	Єдиний соціальний внесок (22 % від основної + додаткової)	42 044,64
4	Амортизаційні відрахування (обладнання)	4 167,00
5	Технічне обслуговування обладнання (5 % від амортизації)	208,35
6	Утримання обладнання (амортизація + технічне обслуговування)	4 375,35

Продовження таблиці 5.3

№	Стаття витрат	Сума, ₴
7	Витрати на електроенергію	702,00
8	Ліцензії та підписки (IDE, CI/CD, домен/SSL, резерв OCR)	4 900,00
9	Оренда інфраструктури (VPS + Redis)	2 040,00
10	Загальновиробничі витрати (10 % від пунктів 1+6+7+8+9)	17 127,74
11	Адміністративні витрати (5 % від пунктів 1+6+8)	8 426,77
12	Маркетингові витрати (онлайн-реклама, матеріали, участь)	22 000,00
13	Збутові витрати (5 % від виробничої собівартості)	37 710,33
	Виробнича собівартість (сума пунктів 1…9 + 10 + 12 + 13)	321 011,06
	Повна собівартість (виробнича + адміністр. + збутова)	367 148,16
14	Прибуток (25 % від повної собівартості)	91 787,04
	Ціна без ПДВ (повна собівартість + прибуток)	458 935,20
15	ПДВ (20 % від ціни без ПДВ)	91 787,04
	Ціна з ПДВ	550 722,24

Таким чином, отримані числові показники підтверджують економічну доцільність розробки та впровадження автономного онлайн-інструмента для створення й редагування електронних видань.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було проведено комплексний аналіз існуючих рішень для роботи з електронними виданнями та визначено основні вимоги до автономного онлайн-інструменту. Було досліджено такі сервіси, як Pressbooks, GitBook та Joomag, їхні переваги та обмеження, що дозволило виокремити ключові вимоги до гнучкості, кастомізації й автономності системи.

На підставі отриманих висновків здійснено обґрунтування вибору стеку технологій: для бекенду розроблено мікросервіси на Scala (обробка PDF-файлів з використанням Akka HTTP) та Python (конвертація PDF у Markdown за допомогою FastAPI і marker-pdf), а також сервіс на Kotlin і Supabase для управління метаданими, авторизації та збереження шаблонів. Такий поділ обов’язків між сервісами забезпечив високу масштабованість та надійність системи.

Розроблено архітектуру мікросервісного рішення з чітким розмежуванням функцій: сервіс прийому та парсингу PDF, сервіс конвертації в Markdown, сервіс зберігання користувацьких даних і шаблонів, що дозволило легко додавати нові модулі без порушення існуючої логіки. Впроваджено механізм back-pressure для безперебійної обробки великих файлів і зменшення ризику перевантаження сервера.

Фронтенд-частину реалізовано на основі React і Vite за архітектурою Feature-Sliced Design. Використання Tailwind CSS та shadcn/ui дозволило швидко створити адаптивний інтерфейс із інтуїтивно зрозумілими компонентами: завантаження PDF, перегляд сторінок, відображення результатів конвертації в Markdown та інтерактивне редагування, збереження змін і генерація адаптивного HTML.

Було інтегровано механізм drag-and-drop для завантаження файлів, реалізовано попередній перегляд PDF за допомогою pdfjs-dist, а також

розроблено систему тимчасового кешування результатів конвертації для скорочення часу повторних запитів. Впроваджено обробку помилок і перевірку валідності вхідних даних, що забезпечило стабільну роботу клієнтської частини навіть за нестабільного інтернет-з’єднання.

Для забезпечення безпеки та контролю доступу користувачів розроблено систему авторизації з JWT-токенами, рольовими правами та зберіганням налаштувань у Supabase. Це дозволило користувачам створювати власні шаблони, зберігати історію конвертацій та мати персоналізований досвід роботи з інструментом.

Проведено тестування кожного компонента системи: юніт-тести для мікросервісів, інтеграційні тести для REST-ендпоінтів та E2E-перевірки для фронтенду з використанням React Testing Library і Vitest. Результати тестування підтвердили коректність роботи основних функцій: завантаження файлів, конвертація структури документа, відображення й редагування вмісту без втрати форматування.

Здійснено розгортання проєкту у середовищі контейнеризації (Docker Compose), налаштовано CI/CD-конвеєр для автоматизованої збірки, тестування та деплою на сервер. Розроблений механізм моніторингу стану сервісів і логування дозволяє оперативно виявляти та усувати помилки в продакшені.

В результаті роботи створено автономний веб-інструмент, який забезпечує повний цикл роботи з PDF-документами: від завантаження та конвертації до редагування й експорту у формати HTML або PDF. Інструмент відрізняється високою продуктивністю, гнучкістю та можливістю масштабування, що робить його корисним для науковців, видавців та освітніх закладів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Електронне видання. URL: https://uk.wikipedia.org/wiki/Електронне_видання#:~:text=Стабільну%20версію%20було%20перевірено%2019,для%20розповсюдження%20в%20незмінному%20вигляді. (дата звернення: 25.05.2025).

2. OpenArchive NURE. Аналіз затребуваності серед читачів та видавців. URL: https://openarchive.nure.ua/server/api/core/bitstreams/afcb5d9f-b6f2-4d3d-b595-f881c9dca010/content (дата звернення: 25.05.2025).

3. Формати електронних книг і текстових документів. URL: https://xn--80aak1d.xn--j1amh/formati-elektronnih-knig-i-tekstovih-dokumentiv.html (дата звернення: 25.05.2025).

4. Intelgr Publishing. Формати електронних книг. URL: https://publishing.intelgr.com/index.php/статьи/33-formatyi-elektronnyih-knig (дата звернення: 25.05.2025).

5. UTMB.edu. WYSIWYG-редактор. URL: https://www.utmb.edu/web/basics/wysiwyg-editor (дата звернення: 25.05.2025).

6. WordPress.org. The Block Editor. URL: https://developer.wordpress.org/block-editor/ (дата звернення: 25.05.2025).

7. Hacker News. Thoughts on Markdown. URL: https://news.ycombinator.com/item?id=30395130 (дата звернення: 25.05.2025).

8. K15t.com. Markdown vs. Rich Formatting: Who Will Win the Ultimate Text-Processing Competition. URL: https://www.k15t.com/blog/2015/04/markdown-vs-rich-formatting-who-will-win-the-ultimate-text-processing-competition (дата звернення: 25.05.2025).

9. Нільсен Я. Designing Web Usability: практичний посібник з юзабіліті. Берклі: New Riders Publishing, 2000. 352 с.

10. Круг С. Don’t Make Me Think: a Common Sense Approach to Web Usability: практичний посібник; 3-є вид. Берклі: New Riders Publishing, 2013. 216 с.

11. Закон України «Про охорону праці» від 14.10.1992 № 2694-XII. Київ: Верховна Рада України, 1992. 12 с.

12. ДСТУ ISO 9241-5:2012 «Ергономічні вимоги до офісної діяльності з обробки інформації»: стандарт. Київ: Держспоживстандарт, 2012. 45 с.

13. Eraser.io. Workspace. URL: https://app.eraser.io/workspace/dTMhTh2xvWPsveHqhMwy (дата звернення: 25.05.2025).

14. INTSURFING. Scala Market Overview 2025. URL: https://www.intsurfing.com/blog/scala-market-overview-2025/ (дата звернення: 25.05.2025).

15. Mobilunity. Hire Developers in Ukraine: Salaries, Costs & Rates Comparison 2025. URL: https://mobilunity.com/blog/cost-to-hire-developers-in-ukraine/ (дата звернення: 25.05.2025).

16. Glassdoor. Salary: Developer in Kyiv, Ukraine 2025. URL: https://www.glassdoor.com/Salaries/kyiv-ukraine-developer-salary-SRCH_IL.0%2C12_IM1186_KO13%2C22.htm (дата звернення: 25.05.2025).

17. Glassdoor. Salary: Software Developer in Kyiv, Ukraine 2025. URL: https://www.glassdoor.com/Salaries/kyiv-ukraine-software-developer-salary-SRCH_IL.0%2C12_IM1186_KO13%2C31.htm (дата звернення: 25.05.2025).

18. Glassdoor. Middle QA Engineer Salary in Kyiv, Ukraine 2025. URL: https://www.glassdoor.com/Salaries/kyiv-middle-qa-engineer-salary-SRCH_IL.0%2C4_IM1186_KO5%2C23.htm (дата звернення: 25.05.2025).

19. Jobicy. Software Developer Salary in Ukraine 2025. URL: https://jobicy.com/salaries/ua/software-developer (дата звернення: 25.05.2025).

20. PayScale. Average Software Engineer Salary in Ukraine 2025. URL: https://www.payscale.com/research/UA/Job%3DSoftware_Engineer/Salary (дата звернення: 25.05.2025).

21. Qubit Labs. Hire Developers in Ukraine: How Much Does It Cost? 2025. URL: https://qubit-labs.com/developer-salary-ukraine/ (дата звернення: 25.05.2025).

22. DigitalOcean. Droplet Pricing 2025. URL: https://www.digitalocean.com/pricing (дата звернення: 25.05.2025).

23. DigitalOcean. Redis Enterprise Basic Tier Pricing 2025. URL: https://www.digitalocean.com/products/managed-databases-valkey (дата звернення: 25.05.2025).

24. Jobicy. QA Engineer Salary in Ukraine 2025. URL: https:// jobicy.com/salaries/ua/quality-assurance-engineer (дата звернення: 25.05.2025).

25. List of European countries by average wage. URL: https://en.wikipedia.org/wiki/List_of_European_countries_by_average_wage (дата звернення: 25.05.2025).

26. ERI Economic Research Institute. Quality Assurance Engineer Salary in Ukraine. URL: https://www.erieri.com/salary/job/qa-engineer/ukraine (дата звернення: 25.05.2025).

27. Glassdoor. Scala Developer Salary in Kyiv 2025. URL: https://www.glassdoor.com.hk/Salaries/kyiv-senior-scala-developer-salary-SRCH_IL.0%2C4_IM1186_KO5%2C27.htm (дата звернення: 25.05.2025).

28. Коваленко О.М. Заробітна плата програмістів: практичний посібник. Харків: ITBooks, 2022. 200 с.

29. Попова Н.І. Менеджмент ІТ-компаній: економічний аспект: підручник. Одеса: Фенікс, 2021. 300 с.

30. Шевченко М.П. Хмарні технології та ціноутворення: підручник. Київ: Наука, 2023. 350 с.

31. Гнатюк Т.Р. QA-інженер в Україні: заробітна плата та кар’єрний ріст. Дніпро: Промінь, 2024. 150 с.

32. Волков С.С. Порівняння зарплат ІТ-фахівців у Європі: дослідження. Київ: Центр досліджень, 2022. 220 с.

33. Клименко Н.С. Інвестиції в ІТ: цифри та прогнози: підручник. Київ: Економіка, 2024. 350 с.

34. Савченко О.М. Практичний посібник із оцінки праці в ІТ-компаніях: навч. посіб. Харків: Право, 2021. 220 с.

35. Бондаренко Л.В. Управління персоналом в ІТ: заробітна плата та мотивація: посіб. Львів: Піраміда, 2023. 240 с.

36. ДСТУ 8772:2015. Бухгалтерський облік. План рахунків бухгалтерського обліку неподільних активів та витрат. Київ: ДП «УкрНДНЦ», 2015. 24 с.

37. ДСТУ 9378:2015. Податок на додану вартість. Методичні рекомендації. Київ: ДП «УкрНДНЦ», 2015. 32 с.

38. Антоненко І.В., Баркова О.І. Електронне видання: теорія та практика. Київ: Наукова думка, 2020. 320 с.

39. Маркдаун: практичний посібник з розмітки тексту. Львів: Ліра-К, 2019. 220 с.

40. Шварц Я. Веб-дизайн: принципи та методи. Київ: BSU, 2017. 310 с.

41. Бойко С.Т. Клієнт-серверні веб-додатки. Дніпро: Поліграфіст, 2018. 400 с.

42. Розробка електронних бібліотек: методологія та практика. Одеса: Фенікс, 2019. 360 с.

43. Теорія і практика видавничої справи. Харків: ITBooks, 2021. 450 с.

44. Сапронов А.В. HTML та CSS: повний посібник. Київ: А.Б.К., 2019. 500 с.

45. Учительство та цифрові технології: електронні ресурси в освітньому процесі. Київ: Літера ЛТД, 2020. 280 с.

46. Павленко І.О. Основи інформаційного дизайну. Львів: Сполом, 2018. 320 с.

47. Шостак В.В. Сучасні мультимедійні технології. Дніпро: Моноліт, 2020. 370 с.

48. Янішевський О. Розробка веб-сайту: від ідеї до реалізації. Харків: Фоліо, 2019. 400 с.

49. Стандарти видавничої справи: ДСТУ та міжнародні: навч. посіб. Київ: Стандарт-Захід, 2016. 260 с.

50. Теорія та практика UX/UI: від макету до коду. Львів: Растр, 2021. 290 с.