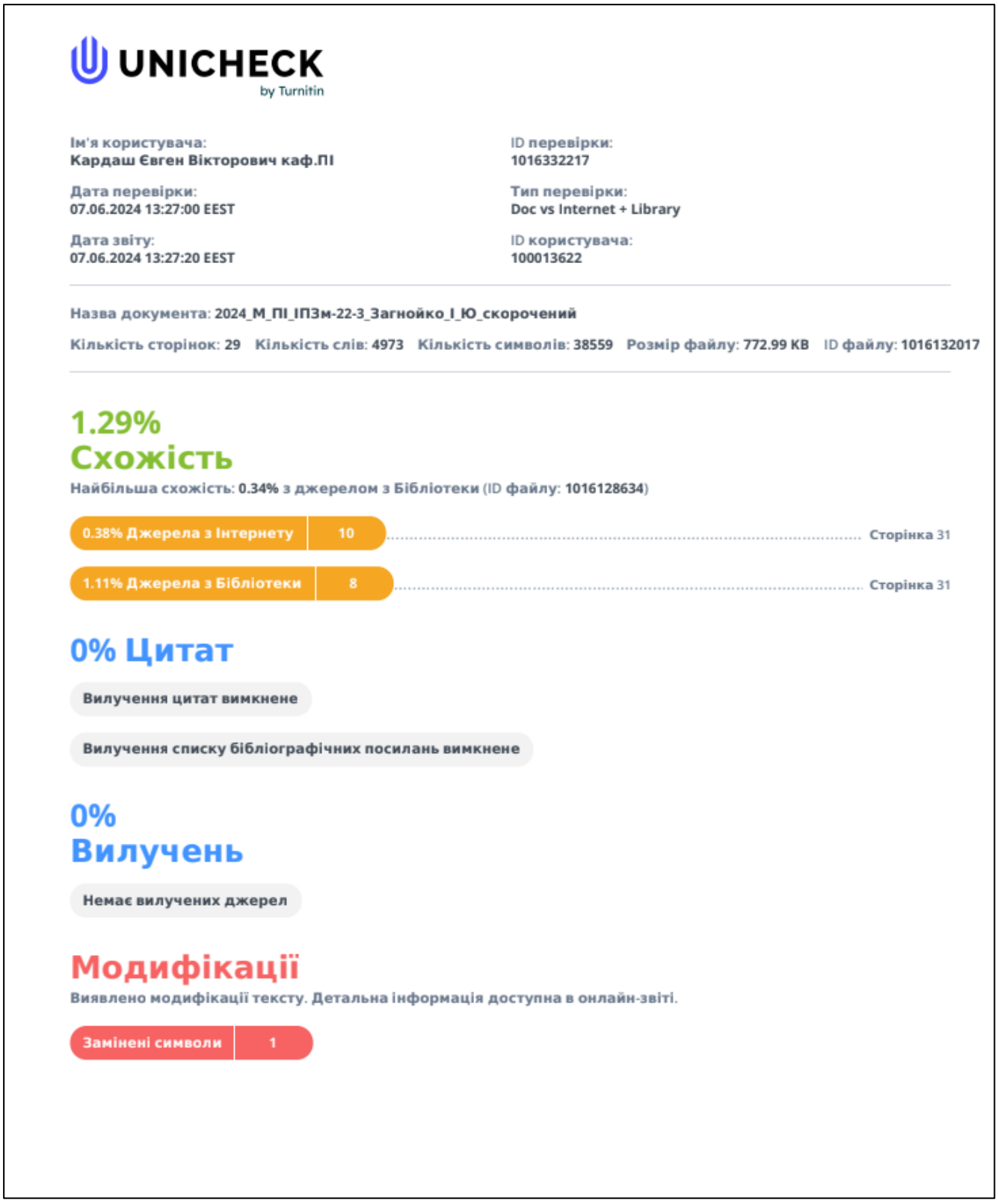
ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



UNICHECK
by Turnitin

Ім'я користувача:
Кардаш Євген Вікторович каф.ПІ

ID перевірки:
1016332217

Дата перевірки:
07.06.2024 13:27:00 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
07.06.2024 13:27:20 EEST

ID користувача:
100013622

Назва документа: 2024_М_ПІ_ІПЗм-22-3_Загнойко_І_Ю_скорочений

Кількість сторінок: 29 Кількість слів: 4973 Кількість символів: 38559 Розмір файлу: 772.99 KB ID файлу: 1016132017

1.29%
Схожість

Найбільша схожість: 0.34% з джерелом з Бібліотеки (ID файлу: 1016128634)

0.38% Джерела з Інтернету 10 ..... Сторінка 31

1.11% Джерела з Бібліотеки 8 ..... Сторінка 31

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

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

Замінені символи 1

Рисунок А.1 – Результат перевірки на унікальність тексту

ДОДАТОК Б

Слайди презентації

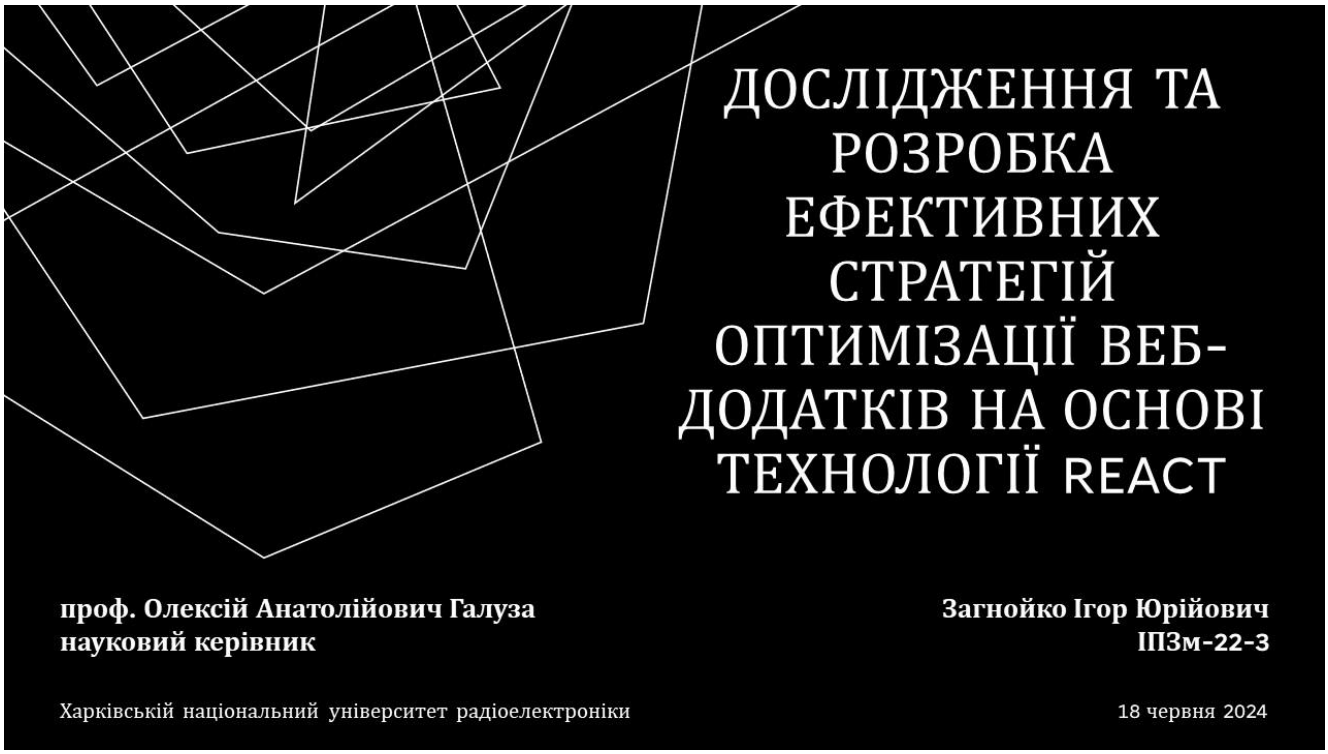


Рисунок Б.1 – Слайд 1

Рисунок Б.2 – Слайд 2

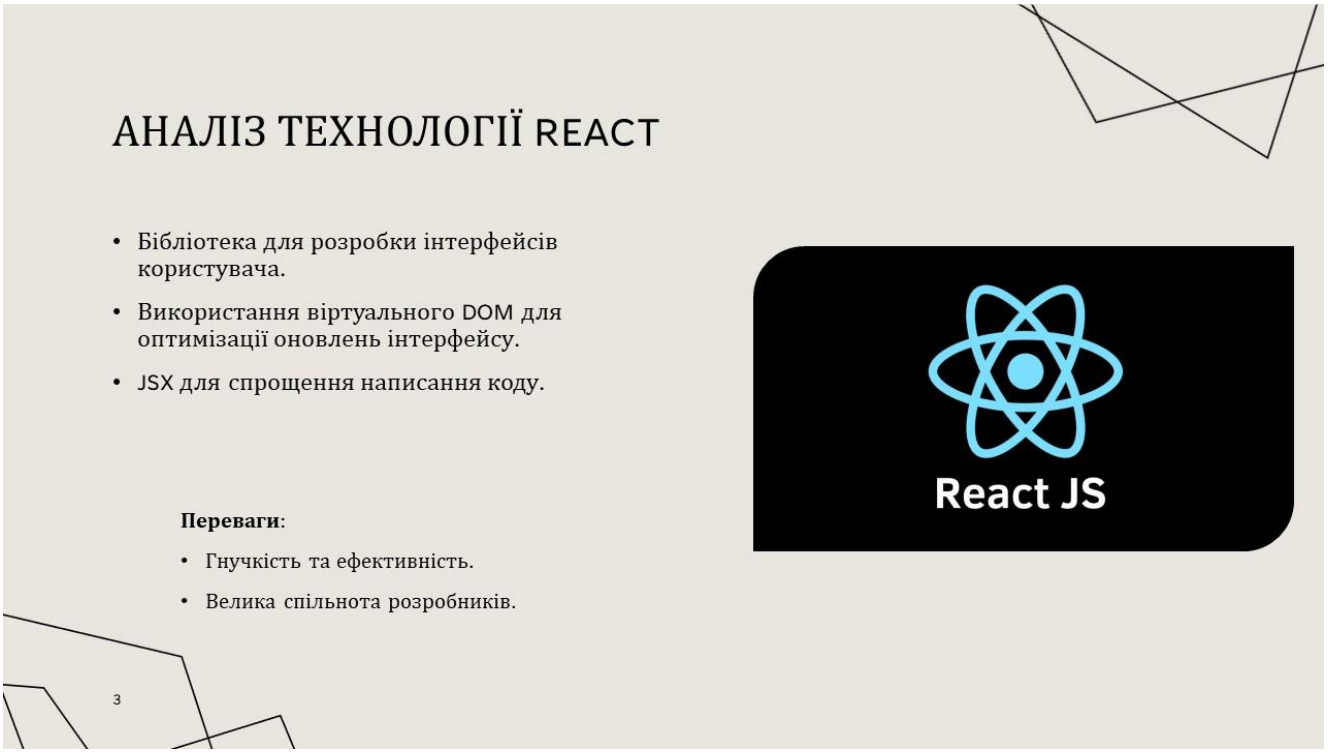


Рисунок Б.3 – Слайд 3

Рисунок Б.3 – Слайд 4

Рисунок Б.5 – Слайд 5

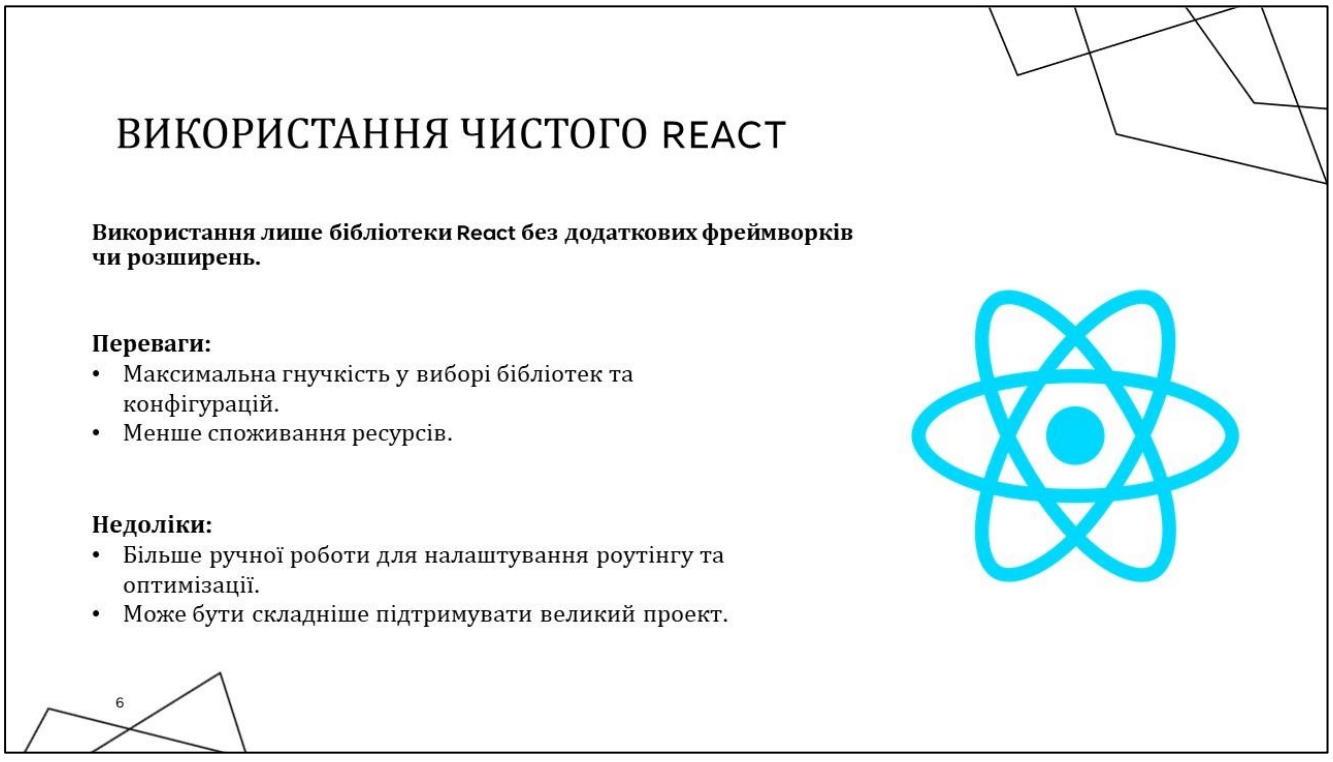


Рисунок Б.6 – Слайд 6

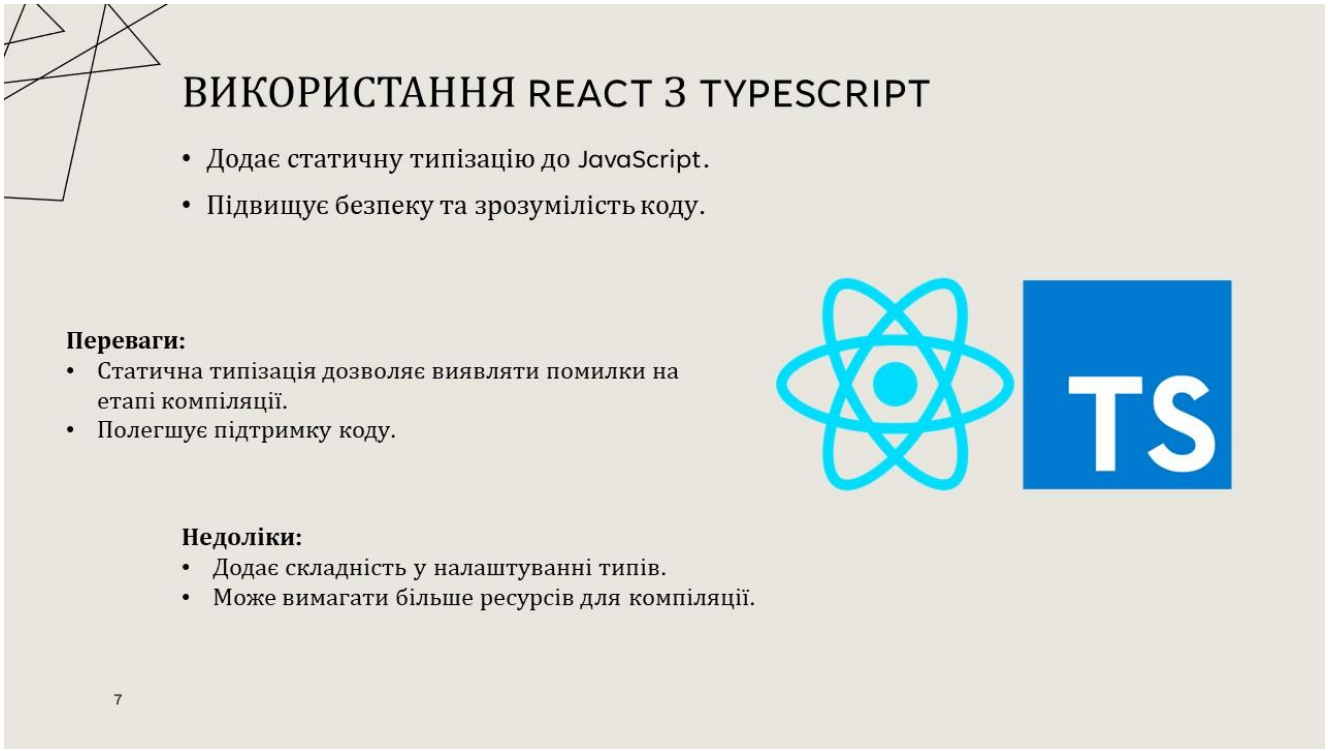


Рисунок Б.7 – Слайд 7

Рисунок Б.8 – Слайд 8

ПОРІВНЯННЯ ТЕХНОЛОГІЙ

	СТУПІНЬ СТАТИЧНОЇ ТИПІЗАЦІЇ	МОЖЛИВОСТІ ОПТИМІЗАЦІЇ ШВИДКОСТІ ЗАВАНТАЖЕННЯ	ГНУЧКІСТЬ ТА ПРОСТОТА РОЗРОБКИ	СПОЖИВАННЯ РЕСУРСІВ	ЛЕГКІСТЬ ІНТЕГРАЦІЇ З ІНШИМИ БІБЛІОТЕКАМИ ТА РОЗШИРЕННЯМИ
React + TypeScript	5	3	3	2	5
TypeScript + Next.js	4	5	4	3	4
Чистий React	3	3	4	2	4

Ваги критеріїв:

- Ступінь статичної типізації: 0.25
- Можливості оптимізації швидкості завантаження: 0.30
- Гнучкість та простота розробки: 0.20
- Споживання ресурсів: 0.10
- Легкість інтеграції з бібліотеками: 0.15

9

Рисунок Б.9 – Слайд 9

РЕЗУЛЬТАТИ ОЦІНКИ ТЕХНОЛОГІЙ

- React + TypeScript: 0.327
- **React + Next.js: 0.381**
- Чистий React: 0.292

Висновок:
Найкращим вибором є React + Next.js, що забезпечує високий рівень оптимізації та продуктивності.

10

Рисунок Б.10 – Слайд 10

Рисунок Б.11 – Слайд 11

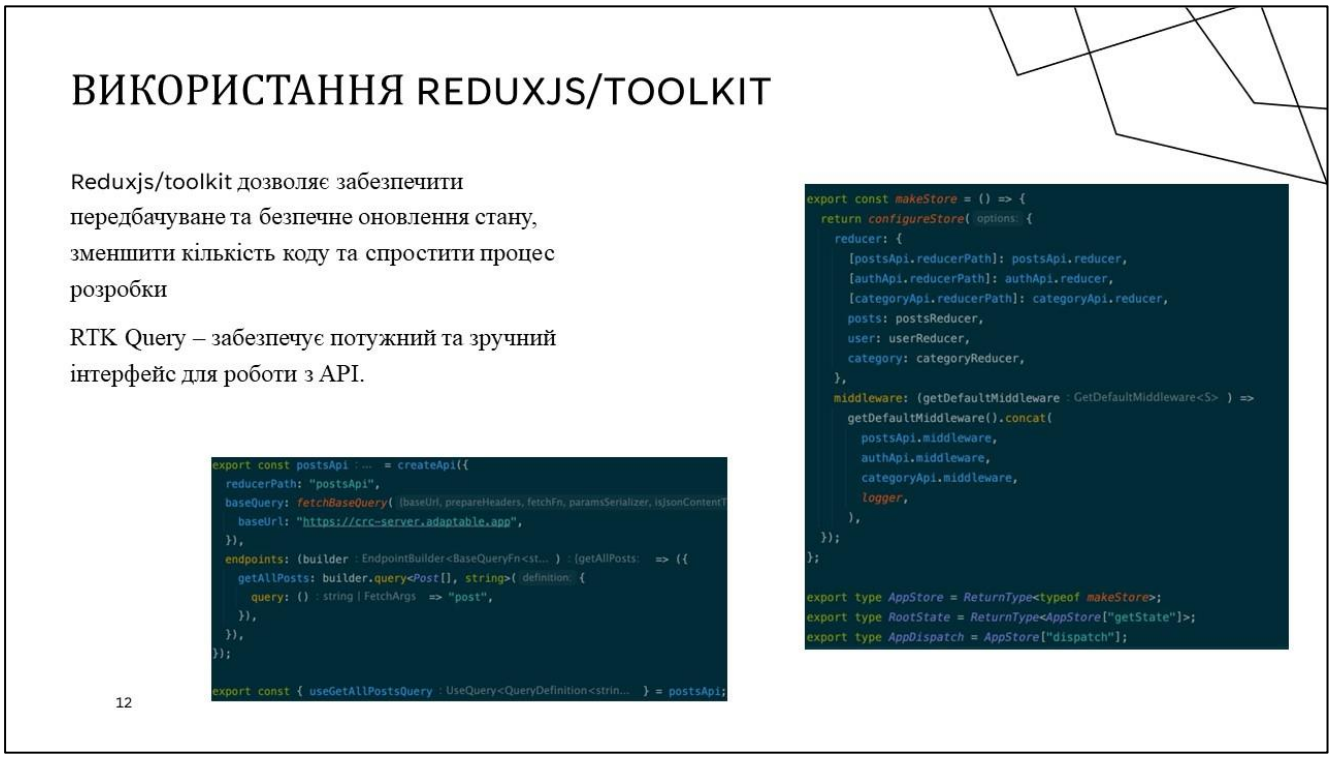


Рисунок Б.12 – Слайд 12

Рисунок Б.13 – Слайд 13

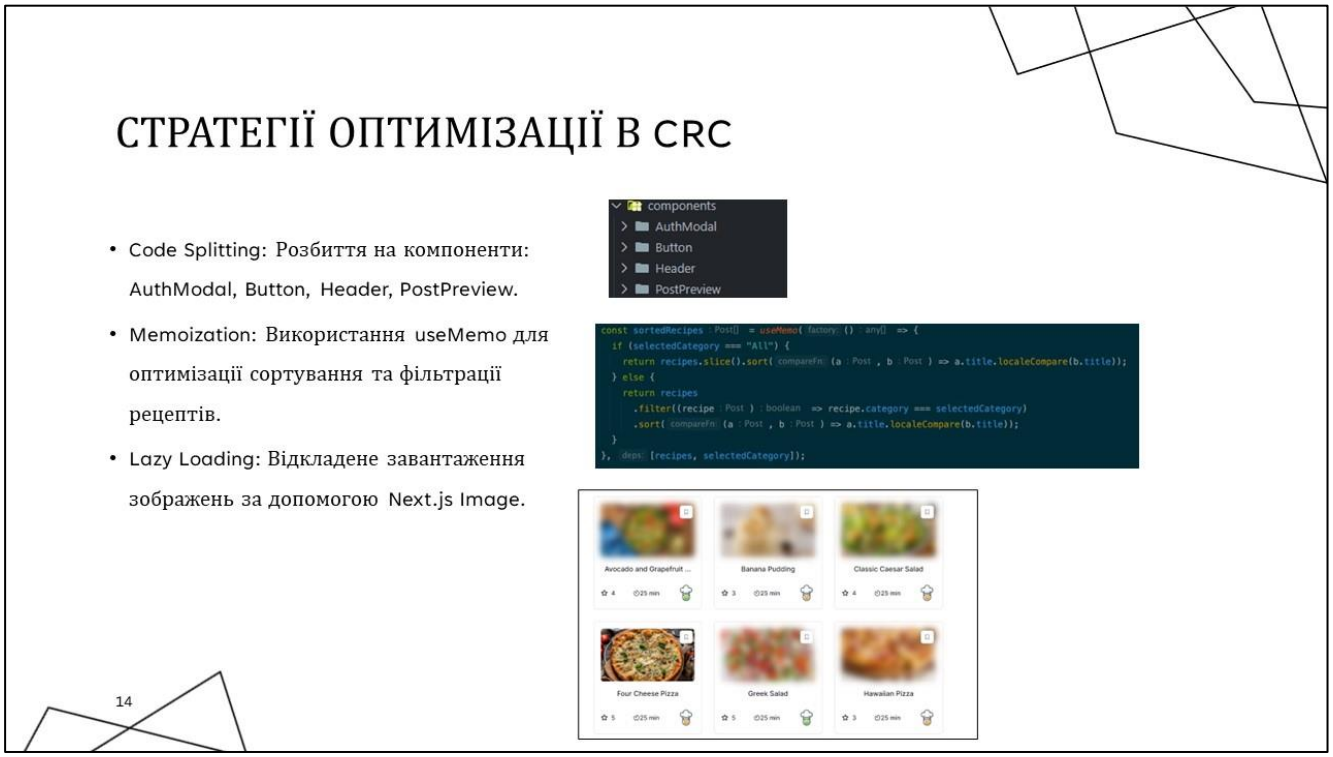


Рисунок Б.14 – Слайд 14

ПОРІВНЯННЯ ТЕХНОЛОГІЙ

Показник	Чистий React	Next.js 14 + reduxjs/toolkit + shadcn/ui
Час завантаження сторінки	2-3 секунди	< 1 секунди
Продуктивність рендерингу	Середня	Висока
Зручність розробки	Низька	Висока
Гнучкість та розширюваність	Висока	Висока
Загальна ефективність	Середня	Висока

15

Рисунок Б.15 – Слайд 15

ВИСНОВКИ

Досягнення:
- Впроваджені стратегії оптимізації значно покращили продуктивність веб-додатків.
- Використання Next.js, Reduxjs/toolkit та Shadcn/ui забезпечило високу продуктивність, зручність розробки та масштабованість.

Рекомендації:
- Використання результатів дослідження для подальшого розвитку веб-додатків.
- Впровадження сучасних технологій та підходів до оптимізації.

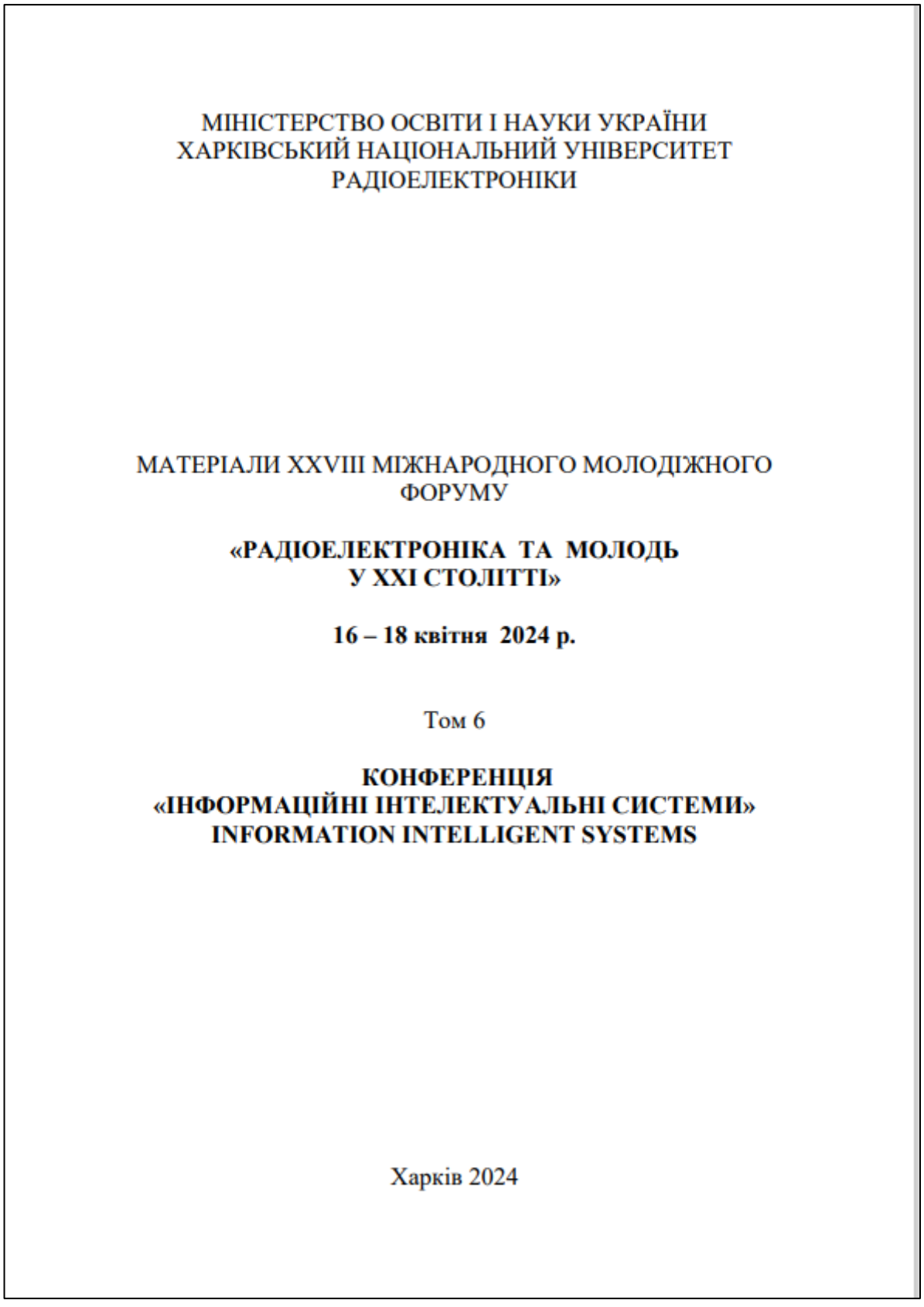
16

Рисунок Б.16 – Слайд 16

Рисунок Б.17 – Слайд 17

ДОДАТОК В

Апробація результатів роботи



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
РАДІОЕЛЕКТРОНІКИ

МАТЕРІАЛИ XXVIII МІЖНАРОДНОГО МОЛОДІЖНОГО ФОРУМУ

«РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ У XXI СТОЛІТТІ»

16 – 18 квітня 2024 р.

Том 6

КОНФЕРЕНЦІЯ
«ІНФОРМАЦІЙНІ ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ»
INFORMATION INTELLIGENT SYSTEMS

Харків 2024

Рисунок В.1 – Титульна сторінка

УДК 004.89 DOI: https://doi.org/10.30837/IYF.IIS.2024.421

ДОСЛІДЖЕННЯ ТА РОЗРОБКА ЕФЕКТИВНИХ СТРАТЕГІЙ ОПТИМІЗАЦІЇ ВЕБ-ДОДАТКІВ НА ОСНОВІ ТЕХНОЛОГІЇ REACT

Загнойко І. Ю.
Науковий керівник – професор Галуза О. А.
Харківський національний університет радіоелектроніки, каф. ПІ
м. Харків, Україна
e-mail: ihor.zahnoiko@nure.ua

This work addresses optimizing React web app performance by reducing unnecessary re-renders, optimizing data caching/loading, minimizing API requests, and decreasing bundle sizes. Key techniques like React.memo, useMemo, client-side caching, code splitting, and lazy loading are utilized. The proposed methods aim to decrease render times, improve responsiveness, reduce API requests, and accelerate load times for an enhanced user experience. Implementation leverages tools like Webpack, React Profiler, and Lighthouse. This comprehensive approach delivers more efficient, high-performing React apps.

Актуальність та постановка проблеми. Оптимізація продуктивності веб-додатків, створених із використанням фреймворку React, є актуальним завданням, оскільки швидкість роботи безпосередньо впливає на залученість і задоволеність користувачів. Основними проблемами, які потребують вирішення, є перерендеринги компонентів, неефективне кешування та завантаження даних, що призводить до зайвих запитів до серверного API, а також великі розміри JS та CSS бандлів, які уповільнюють завантаження та обробку сторінки. У цій роботі ставиться за мету дослідити та запропонувати ефективні стратегії оптимізації продуктивності React додатків. Розглядаються основні проблеми та шляхи їх вирішення на різних рівнях – від архітектури до окремих компонентів.

Методи та засоби вирішення проблеми. Оптимізація на рівні компонентів, а саме React.memo застосовується для запобігання непотрібних перерендерингів шляхом меморизації результатів обчислень пропсів. useMemo та useCallback використовуються для меморизації результатів важких обчислень та функцій, які передаються як пропси дочірнім компонентам. Перевірка на зміну пропсів та стану в shouldComponentUpdate дозволяє запобігти непотрібних перерендерингів класових компонентів.

Оптимізація запитів до серверного API, декілька запитів об'єднуються в один для зменшення кількості HTTP-запитів. GraphQL використовується для ефективного отримання лише необхідних даних. Відповіді API кешуються на клієнтській стороні для уникнення повторних запитів. При швидкому скролі запити до API відкладаються для уникнення затримок.

Оптимізація розміру JS та CSS бандлів: застосовується розбиття коду на чанки (code splitting) за допомогою динамічного імпорту для

Рисунок В.2 – Перша сторінка тез

завантаження лише необхідних частин додатку. Динамічне підвантаження компонентів відбувається за допомогою React.lazy для завантаження компонентів лише при необхідності. Невикористовуваний код виключається (tree shaking) за допомогою сучасних інструментів збірки. Мініфікація та стиснення коду зменшують розмір файлів.

Очікувані результати. Застосування запропонованих методів та засобів оптимізації продуктивності React додатків дозволить значно зменшити кількість непотрібних перерендерингів компонентів та обчислень, скоротивши час рендерингу та підвищивши відгук інтерфейсу. Завдяки оптимізації завантаження та кешування даних буде зменшена кількість зайвих запитів до серверного API, що призведе до прискорення відображення даних. Розмір JS та CSS бандлів зменшиться, що позитивно вплине на час першого рендерингу та прискорить завантаження та обробку сторінок.

Як результат, покращення швидкодії та відгуку додатку сприятиме підвищенню задоволеності користувачів. Загалом, застосування цих методів та засобів оптимізації дозволить створювати більш ефективні та продуктивні React додатки, орієнтовані на забезпечення кращого користувацького досвіду.

Висновок. Оптимізація продуктивності React додатків вимагає комплексного підходу на всіх рівнях – від архітектури до окремих компонентів. Застосування запропонованих методів та засобів, таких як React.memo, useMemo, useCallback, кешування даних, оптимізація запитів до API, розбиття коду на чанки, динамічне підвантаження компонентів, дозволить суттєво підвищити швидкодію та продуктивність як нових, так і вже існуючих проектів на React. Це, в свою чергу, позитивно вплине на залученість та задоволеність користувачів, забезпечивши плавний та відгукний користувацький досвід, що є надзвичайно важливим для успіху будь-якого веб-додатку.

Список використаних джерел:

1. Optimize Your React App: A Practical Guide to Better React Performance. URL: https://www.copycat.dev/blog/react-performance/ (дата звернення: 02.03.2024).

2. Gackenheimer C. Introduction to React. Berkeley: Apress, 2015. 188 p.

3. Research of Ways to Increase the Efficiency of Functioning Between Firewalls in the Protection of Information Web-Portals in Telecommunications Networks / Z. Dudar та ін. Current Trends in Communication and Information Technologies. Cham, 2021. С. 272–292. URL: https://doi.org/10.1007/978-3-030-76343-5_14 (дата звернення: 10.03.2024).

Рисунок В.3 –Друга сторінка тез

Рисунок В.3 – Сертифікат учасника

ДОДАТОК Г

Експертний висновок результатів перевірки кваліфікаційної роботи на відповідність оформлення вимогам ДСТУ 3008: 2015

Експертний висновок результатів перевірки кваліфікаційної роботи

студент (посада) | програмної інженерії (кафедра) | ІПЗм-22-3 (група)

Загнойко Ігор Юрійович
(прізвище, ім'я, по батькові)

Зауваження

Пункт ДСТУ 3008-2015	Зміст пункту	Сторінка кваліфікаційної роботи
1	2	3
	7.1 Загальні положення	
	7.3 Нумерація сторінок звіту	
	7.4 Нумерація розділів, підрозділів, пунктів, підпунктів	
	7.5 Рисунки	
	7.6 Таблиці	
	7.7 Переліки	
	7.8 Примітки	
	7.9 Виноски	
	7.10 Формули та рівняння	
	7.11 Посилання	
	7.13 Список авторів	
	7.14 Скорочення та умовні познаки	
	7.15 Додатки	

зауважень немає

Експерт _____ (підпис) Олена ОЛІЙНИК (прізвище, ініціали)

11.06.2024

Рисунок Г.1 – Результат перевірки записки на відповідність оформлення вимогам ДСТУ 3008:2015