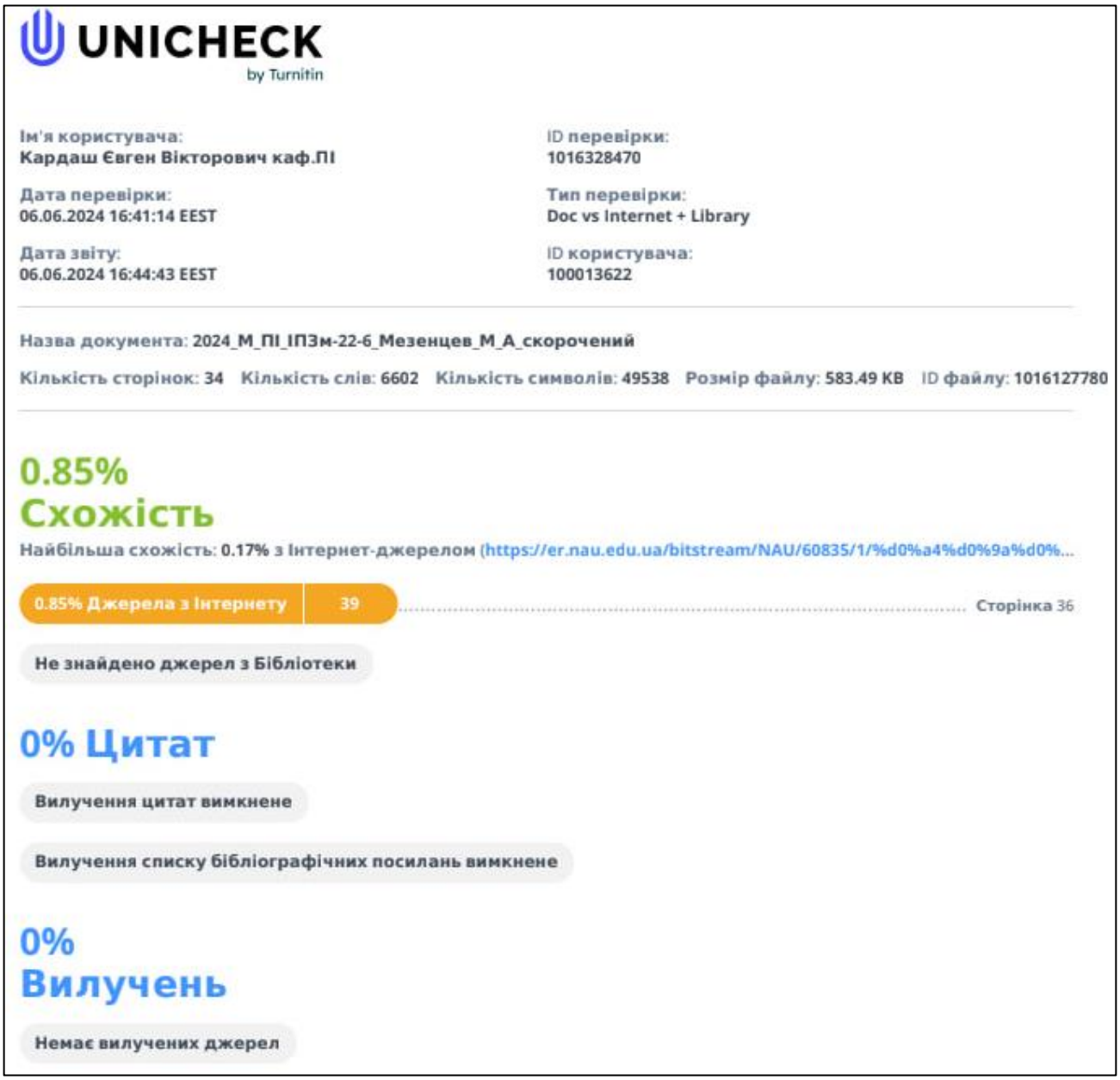
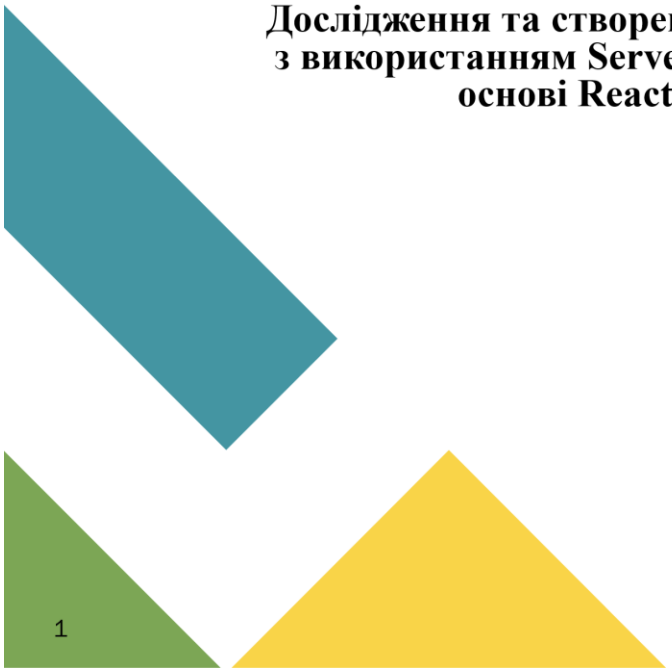
ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



ДОДАТОК Б

Слайди презентації



Аналіз предметної галузі

- Обираючи вид рендерингу, ми визначаємо, як саме сторінки будуть відображатися для користувачів та як вони будуть генеруватися на сервері чи клієнті. Це має великий вплив на продуктивність, швидкодію, індексацію пошуковими системами, безпеку та інші аспекти веб-додатку.
- Вибір підходу до рендерингу (CSR, SSR, SSG тощо) визначається різними факторами, такими як вимоги бізнесу, потреби користувачів, характеристики проекту та його масштаби, а також стратегії SEO.

Постановка задачі

- Провести аналіз різних методів рендерингу.
- Визначити, який метод є кращим в залежності від задач.
- Проаналізувати сучасні технології для створення веб-застосунків, такі як React та Next.js.
- Провести експереминтальне дослідження та визначити який метод рендерінгу є швидшим за інші.

Роль React та Next.js в SSR

- Автоматичний SSR: Next.js автоматично рендерить сторінки на сервері, якщо це необхідно. Він визначає, які сторінки повинні бути відрендерені на сервері, а які можна рендерити на клієнті.
- Статична генерація: Окрім SSR, Next.js підтримує статичну генерацію сторінок (SSG), що дозволяє попередньо створювати сторінки під час збірки застосунку.
- React надає базову інфраструктуру для створення компонентів, які можуть бути відрендерені на сервері та гідровані на клієнті. Next.js, у свою чергу, доповнює React, пропонуючи потужні інструменти для автоматизації та оптимізації процесу серверного рендерингу, спрощуючи розробку високопродуктивних та SEO-оптимізованих веб-застосунків.

5

Різниця між відами рендерінгу

- Рендеринг на стороні клієнта (CSR) відбувається у браузері, після отримання сторінки, що дозволяє побудувати динамічні інтерфейси, але може призвести до більшого часу завантаження сторінки.
- Рендеринг на стороні сервера (SSR) генерує HTML на сервері перед відправленням його клієнту, що поліпшує швидкодію завантаження сторінок та забезпечує кращу індексацію пошуковими системами.
- Статичне попереднє рендеринг (SSG) генерує статичні файли під час збірки, що дозволяє швидко завантажувати сторінки та знижує навантаження на сервер.

6

Метод рендерінгу в залежності від проєкту

- Прості проекти, такі як односторінкові інтернет -магазини або особисті блоги, часто використовують **CSR (Client-Side Rendering)**. Цей підхід дозволяє створювати динамічний та інтерактивний вміст без перезавантаження сторінки, що дуже зручно для користувачів, особливо при взаємодії з продуктами або контентом.
- Більш складні проекти, такі як фінансові платформи або біржі акцій, найчастіше використовують **SSR (Server-Side Rendering)**. Цей метод забезпечує швидке завантаження сторінок та забезпечує кращу безпеку, оскільки HTML генерується на сервері перед відправкою клієнту. Це особливо важливо для фінансових додатків, де точність та безпека даних є пріоритетними.
- Для статичних сайтів або лендінгових сторінок, які не потребують динамічної зміни вмісту, ідеально підходить **SSG (Static Site Generation)**. Цей метод дозволяє швидко завантажувати сторінки та знижує навантаження на сервер, що особливо важливо для сайтів з великим обсягом відвідувачів.

7

Розробка веб -сайту з використанням сучасних технологій

React - це бібліотека JavaScript, створена компанією Facebook. Вона призначена для створення інтерфейсів користувача, які відображаються на веб-сторінках.

Основна ідея React полягає в тому, щоб розбити інтерфейс користувача на невеликі компоненти, які можна перевикористовувати та легко управляти.

React використовує концепцію віртуального DOM (Document Object Model), що дозволяє ефективно оновлювати тільки ті частини сторінки, які змінилися, замість повного перерендерингу всього дереваDOM.

Next.js - це фреймворк для розробки веб-додатків на основі React.

Він надає розширений набір функцій та інструментів, що полегшують створення як статичних, так і динамічних вебдодатків.

Next.js додає до React додаткові можливості, такі як попереднє рендеринг (SSR та SSG), маршрутизація на основі файлів, оптимізоване завантаження сторінок та інші

8

Різниця між відами рендерінгу

- Рендеринг на стороні клієнта (CSR) відбувається у браузері, після отримання сторінки, що дозволяє побудувати динамічні інтерфейси, але може призвести до більшого часу завантаження сторінки.
- Рендеринг на стороні сервера (SSR) генерує HTML на сервері перед відправленням його клієнту, що поліпшує швидкодію завантаження сторінок та забезпечує кращу індексацію пошуковими системами.
- Статичне попереднє рендеринг (SSG) генерує статичні файли під час збірки, що дозволяє швидко завантажувати сторінки та знижує навантаження на сервер.

9

Приклади програмної реалізації компонентів з різними видами рендерінгу

```
export default function Csr() {
  const [serverTime, setServerTime] = useState("");

  useEffect(() => {
    const fetchServerTime = async () => {
      const response = await fetch("/api/time");
      const data = await response.json();
      setServerTime(data.serverTime);
    };

    fetchServerTime();
  }, []);

  return (
    <>
      <Head>
        <title>Create Next App</title>
        <meta name="description" content="Generated by create next app" />
        <meta name="viewport" content="width=device-width, initial-scale=1" />
        <link rel="icon" href="/favicon.ico" />
      </Head>
      <main className={`${styles.main} ${inter.className}`}>
        <div className={styles.description}>
          <p>
            This is CSR page
          </p>
          <div>
```

```
export async function getServerSideProps() {
  // Fetch server time
  const serverTime = new Date().toString();


  return {
    props: {
      serverTime,
    },
  };
}

export default function Home({ serverTime }) {
  return (
    <>
      <Head>
        <title>Create Next App</title>
        <meta name="description" content="Generated by create next app" />
        <meta name="viewport" content="width=device-width, initial-scale=1" />
        <link rel="icon" href="/favicon.ico" />
      </Head>
      <main className={`${styles.main} ${inter.className}`}>
        <div className={styles.description}>
          <p>
            this is SSR page
          </p>
          <div>
            <a
              href="https://vercel.com?utm_source=create-next-app&utm_medium=default-template&utm_campaign=create-next-app"
              target="_blank"
              rel="noopener noreferrer"
            >
              By{" "}
              <Image
```

10

Результат швидкості рендерингу клієнтського компоненту (CSR)

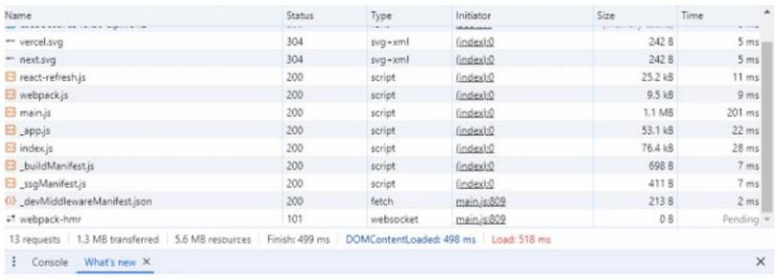
Name	Status	Type	Initiator	Size	Time
next.svg	304	svg+xml	ssg:0	242 B	6 ms
react-refresh.js	200	script	ssg:0	25.2 kB	11 ms
webpack.js	200	script	ssg:0	9.5 kB	13 ms
main.js	200	script	ssg:0	1.1 MB	292 ms
_app.js	200	script	ssg:0	53.1 kB	26 ms
ssg.js	200	script	ssg:0	76.8 kB	38 ms
_buildManifest.js	200	script	ssg:0	698 B	14 ms
_ssgManifest.js	200	script	ssg:0	411 B	14 ms
_devMiddlewareManifest.json	200	fetch	main.js:809	213 B	2 ms
webpack-hmr	101	websocket	main.js:809	0 B	Pending
favicon.ico	304	x-icon	Other	243 B	4 ms

14 requests | 1.3 MB transferred | 5.7 MB resources | Finish: 736 ms | DOMContentLoaded: 662 ms | Load: 686 ms

Console | What's new

11

Результат швидкості рендерингу серверного компоненту (SSR)



Name	Status	Type	Initiator	Size	Time
vercel.svg	304	svg+xml	(index):0	242 B	5 ms
next.svg	304	svg+xml	(index):0	242 B	5 ms
react-refresh.js	200	script	(index):0	25.2 kB	11 ms
webpack.js	200	script	(index):0	9.5 kB	9 ms
main.js	200	script	(index):0	1.1 MB	201 ms
_app.js	200	script	(index):0	53.1 kB	22 ms
index.js	200	script	(index):0	76.4 kB	28 ms
_buildManifest.js	200	script	(index):0	698 B	7 ms
_ssgManifest.js	200	script	(index):0	411 B	7 ms
_devMiddlewareManifest.json	200	fetch	main.js:809	213 B	2 ms
webpack-hmr	101	websocket	main.js:809	0 B	Pending

13 requests | 1.3 MB transferred | 5.6 MB resources | Finish: 499 ms | DOMContentLoaded: 498 ms | Load: 518 ms

Console | What's new

12

Опис сформованих рекомендацій

- Метод рендерінгу необхідно обирати в залежності від потреб та можливостей бізнесу.
- Сучасні технології React та Next.js суттєво спрощують розробку веб - застосунків та додають значну кількість функціонала.
- Бібліотека Next.js гарно підходить як для серверного, так і для клієнтського рендерингу.
- Метод серверного рендерінгу швидше, за клієнтський.

Висновки

- Проведено аналіз серверного та клієнтських видів рендерингу.
- Проведено дослідження в ході якого був зроблений висновок, що серверний вид рендерингу є швидшим за клієнтський.
- На основі проведених експериментів були сформовані рекомендації про вибір технологій та рендерингу.
- Подано тези доповіді на двадцять восьмий міжнародний форум «РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ У XXI СТОЛІТТІ».
- Подано статтю в збірник Intelligent Systems Workshop на восьму інтернаціональну конференцію з теми “Комп’ютерна лінгвистика та Інтелектуальні системи” (CoLInS-2024).

ДОДАТОК Г

Апробація результатів роботи

Intelligent Systems Workshop on 8th International Conference on Computational Linguistics and Intelligent Systems (CoLInS-2024), сторінки 122-133.

ДОДАТОК Д

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-22-6 (група)

Мезенцев Максим Андрійович

(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 позначки	
	7.15 Додатки	

зауважень немає

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

08.06.2024