МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма: «Розробка програмного забезпечення»

Група: 4РП-08

Дипломний проєкт

здобувача освіти денної форми навчання
РП.08.08.000.ДП

ДЗЕБАНА
ВЛАДИСЛАВА ОЛЕКСАНДРОВИЧА

м. Одеса
2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **121 «Інженерія програмного забезпечення»**
Освітньо-професійна програма: «**Розробка програмного забезпечення»**
Група: **4РП-08**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту на тему:

Розробка web-застосунку для інтелектульного пошуку зображень

Проєктний матеріал складається з пояснювальної записки на 72 сторінках та графічного (презентаційного) матеріалу на 13 аркушах (слайдах)

Дипломник _____ (Дзебан В.О.)
Керівник _____ (Гаджиєв М.М.)

Консультанти:
з економічного розділу _____ (Канський М.Ю.)
з розділу охорони праці та техніки безпеки _____ (Чорновол Н.І.)
з нормоконтролю _____ (Петрашова В.І.)
старший консультант _____ (Кривченко Ю.В.)

До захисту допущений
Голова циклової комісії _____ (Кривченко Ю.В.)
Завідувач відділення _____ (Краснокутська К.Г.)

Захист «21» 06 2025 р. Протокол ЕК № 1
Оцінка ЕК 5/90

Секретар ЕК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *комп'ютерних систем* Комісія *КТ та ПІ*

Спеціальність *121 «Інженерія програмного забезпечення»*

Освітньо-професійна програма *«Розробка програмного забезпечення»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І.В.*

"12" 05 2025 р.

ЗАВДАННЯ

на дипломний проєкт

Здобувачеві освіти *Дзебану Владиславу Олександровичу*

(прізвище, ім'я, по батькові)

1. Тема проєкту *Розробка web-застосунку для інтелектуального пошуку зображень*

затверджена наказом по коледжу від "14" листопада 2024р. № 246

2. Термін здачі закінченого проєкту

3. Вихідні данні до проєкту *Проєктування web-застосунку для інтелектуального пошуку зображень; Використання мови JavaScript для front-end і back-end. Реалізація підказки за допомогою ШІ; Інтеграція Google Custom Search API для пошуку; Реалізація REST API; Використання бази даних MySQL; Використання середовища розробки Visual Studio Code*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

Аналіз ринку; Обрання технологій розробки; Проєктування основних функціональних елементів застосунку; Реалізація серверного коду; Реалізація ендпоінту для генерації підказок; Реалізація логіки для авторизування; Реалізація перевірки токенів користувачів; Огляд розробленого застосунку

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Вибір технологій для розробки front-end та back-end; Взаємозв'язок між front-end та back-end частинами; Порівняння асинхронної моделі вводу-виводу з синхронною; Алгоритм роботи Virtual DOM від React; Реалізація серверу з Express.js; Реалізація ендпоінту для генерації підказок; Реалізація логіки авторизації; Реалізація перевірки токену користувача; Блок схема алгоритму користування web-застосунком; Огляд web-застосунку

6. Консультанти по проєкту, із зазначенням розділів проєкту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний розділ	*Гаджиєв М.М.*		
Економічний розділ	*Канський М.Ю.*		
Розділ охорони праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		
Старший консультант	*Кривченко Ю.В.*		

7. Дата видачі завдання _____

Керівник *Гаджиєв М.М.* _____
(підпис)

Завдання прийняв до виконання *Дзебан В.О.* _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проєкту	Термін виконання етапів дипломного проєкту (роботи)	Відмітка про виконання
1	*Вступ*	*15.05.25*	Виконав
2	*Аналіз Ринку*	*16.05.25*	Виконав
3	*Формулювання технічних вимог*	*18.05.25*	Виконав
4	*Аналіз технологій розробки*	*21.05.25*	Виконав
5	*Проектування web-застосунку*	*25.05.25*	Виконав
6	*Розробка web-застосунку*	*02.06.25*	Виконав
7	*Оформлення пояснювальної записки*	*09.06.25*	Виконав
8	*Підготовка графічної частини презентації*	*10.06.25*	Виконав
9	*Економічний розрахунок*	*11.06.25*	Виконав
10	*Опис аспектів охорони праці та техніки безпеки*	*12.06.25*	Виконав
11	*Підведення висновків*	*13.06.25*	Виконав
12	*Підготовлення доповіді до захисту*	*16.06.25*	Виконав
13			
14			
15			

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			РП 08. 08 000. 00 ДП	Дипломний проєкт		
А4			РП 08. 08 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	РП 08. 08 000. 00 ДП		
Розробив		Дзебан В.О.		13.06	Розробка web-застосунку для інтелектуального пошуку зображень	Літ.	Аркуш 4 / Аркушів 72
Перевірив		Гаджиєв М.М.		13.06		НДП	
Н. контр.		Петрашова В.І.		18.06.25		ВСП "ОТФК ОНТУ" гр.4РП-08	
Затверд.		Кривченко Ю.В.		18.06			

ЗМІСТ

ВСТУП

Штучний інтелект (надалі - ШІ) це область науки, що займається створенням комп'ютерів та машин, які можуть мислити, вчитися і діяти так чи інакше, та як зазвичай вимагає людського інтелекту або включає дані, обсяг яких перевищує можливості людини. Окрім цього область ШІ охоплює дуже багато різних дисциплін в які входять: інформатика, аналітика даних і статистика, розробка апаратного та програмного забезпечення а також лінгвістика.

На операційному рівні для використання в бізнесі ШІ являє собою набір технологій, які були створені в основному на машинному навчанні, які використовуються для аналізу даних, прогнозування, класифікації об'єктів, обробки природної мови, рекомендацій, інтелектуального пошуку даних і багато чого іншого.

Теж окрім ANI моделі яка зараз використовується є ще види штучного інтелекту але вони наразі тільки теоретичні:

- Артифіційний суперінтелект (Artificial Super Intelligence, ASI) Це концептуальна форма ШІ, яка не лише може відтворити розумові здібності людини, але й значніше перевищує їх. Припускається, що ASI зможе повністю розуміти людські емоції та думки, що дозволить йому впливати на волю людини.
- Штучний загальний інтелект (Artificial General Intelligence, AGI)AGI є вершиною штучного інтелекту, що характеризується здатністю виконувати будь-які інтелектуальні завдання, які може виконувати людина. На відміну від вузького ШІ, який призначений для виконання конкретних завдань (таких як розпізнавання зображень або обробка природної мови), AGI має потенціал розуміти, навчатися та застосовувати знання в широкому діапазоні завдань.
- Обмежений штучний інтелект (Artificial Narrow Intelligence, ANI) Цей тип ШІ, що спеціалізується на виконанні однієї або кількох конкретних функцій. Він демонструє лише вузькі прояви розумної поведінки і не здатен розвиватися самостійно чи діяти автономно без участі людини.

1 ОСНОВНИЙ РОЗДІЛ

Створення web-застосунку для інтелектуального пошуку зображень включає кілька ключових етапів та компонентів. Основна мета полягає в розробці інтерфейсу та логіки, яка дозволяє користувачеві знаходити зображення на основі текстового запиту, з використанням ШІ для генерації підказок і уточнення наміру користувача.

Одним з найбільших новаторів у галузі ШІ був Джон Маккарті, який отримав звання «батька штучного інтелекту» за свій внесок у розвиток інформатики та штучного інтелекту. Маккарті вперше вжив термін «штучний інтелект» у 1956 році, коли організував Дартмутську конференцію, яка вважається початком штучного інтелекту як галузі науки, саме тому Маккарті класифікував ШІ таким чином: «Штучний Інтелект - це наука про те, як змусити машини робити те, що потребує інтелекту, якби це робила людина». Саме так зараз більшість сучасних людей описали б Штучний Інтелект.

З плином часу ШІ почали розглядати як сукупність алгоритмів і програмних рішень, здатних вирішувати завдання подібно до того, як це робить людина. Серед ключових характеристик ШІ можна виділити здатність розуміти мову, навчатися, мислити логічно та приймати рішення. Штучний інтелект є комплексом сучасних технологій, які стрімко вдосконалюються і розширюють межі свого застосування. До таких технологій належать:

1 Обробка природної мови дає змогу машинам аналізувати, розпізнавати та інтерпретувати текстову інформацію, враховуючи граматику, зміст та контекст, а також формувати осмислені відповіді.

2 Машинне навчання напрям, що досліджує алгоритми, які дозволяють системам самостійно вчитись на основі вхідних даних і поступово підвищувати ефективність своїх рішень без жорстко запрограмованих інструкцій.

4 Віртуальні помічники та чат-боти інтелектуальні агенти, що взаємодіють із користувачами природною мовою, надають інформацію, відповідають на запити та виконують різноманітні завдання.

5 Рекомендаційні системи - аналітичні механізми, що враховують інтереси,

поведінку та вподобання користувачів, щоб пропонувати товари, сервіси або контент, які можуть бути для них корисними чи цікавими.

1.1 Процес розробки web-застосунку для інтелектуального пошуку зображень

1.1.1 Визначення цільової аудиторії

Для початку треба виявити цільову аудиторію котра буде користуватися даним web-застосунком.

Основними категоріями аудиторії є, по-перше, креативні фахівці:

Дизайнери, ілюстратори, маркетологи та інші представники творчих професій звертаються до таких сервісів в першу чергу за натхненням, їм важливо швидко знаходити візуальні образи, що відповідають певному настрою, стилю або концепції.

По-друге Розробники та веб-дизайнери:

Люди, які створюють інтерфейси та сайти, часто потребують конкретних візуальних елементів: іконок, фонів, патернів або шаблонів. Для них важлива точність вони знають, чого хочуть, але не завжди можуть підібрати правильний пошуковий запит.

По-третє Блогери та власники соцмереж

Творці контенту для соціальних мереж часто шукають зображення, які ілюструють емоції, актуальні теми або тренди. Для них важливо знаходити яскраві, виразні зображення, які швидко привертають увагу аудиторії.

По-четверте Освітня та студентська аудиторія

Студенти, викладачі та автори навчальних матеріалів використовують зображення для ілюстрації складних тем, для підготовки презентацій або оформлення курсів. Часто вони стикаються з труднощами пошуку не перевантажених візуалів.

По-п’яте Звичайні користувачі

Широка аудиторія це маса людей, які просто шукають зображення для особистих потреб: вітальні листівки, ідеї для ремонту, шпалери на телефон або референси.

1.2 Аналіз ринку

До початку реалізації проєкту важливо було проаналізувати ринок в галузі пошуку зображень, зокрема ті, що використовують штучний інтелект для покращення інтерпретації запитів користувачів. Особливу увагу було приділено інструментам, які застосовують ШІ не для створення зображень, а для аналізу введених пошукових фраз і генерації відповідних підказок, що робить результати пошуку більш точними.

Сектор аналізу запитів, підсилений технологіями штучного інтелекту, нині демонструє динамічне зростання. Це відбувається на тлі глобального попиту на автоматизовані рішення та збільшення ролі візуального контенту в онлайн-взаємодії. Бізнеси та користувачі активно шукають ефективні інструменти для пошуку та управління зображеннями, і розумні системи стають для них ключовим інструментом. Серед основних гравців у сфері - технологічні гіганти, такі як Google, Amazon, IBM та Microsoft. Вони пропонують потужні інструменти для розпізнавання об’єктів, класифікації зображень та пошуку за візуальною схожістю. Проте більшість із них орієнтовані переважно на корпоративних клієнтів і вимагають складної технічної інтеграції. До того ж, ці сервіси здебільшого зосереджуються на аналізі завантаженого візуального контенту, а не на інтелектуальній допомозі користувачеві при формулюванні пошукового запиту.

Згідно з даними Fortune Business Insights, глобальний ринок розпізнавання зображень оцінюється в $58,56 млрд у 2025 році і прогнозується досягти $163,75 млрд до 2032 року, при середньому річному темпі зростання 15,8%. У той же час, Statista повідомляє, що ринок розпізнавання зображень досягне $15,37 млрд у 2025 році, з очікуваним CAGR 14,39% до 2031 року. Різниця в оцінках може бути пов'язана з відмінностями в методологіях і охопленні досліджень.

Ключовими факторами зростання є:

– Збільшення обсягу візуальних даних, що вимагають ефективної обробки та аналізу.
– Розвиток технологій глибокого навчання та нейронних мереж, що

підвищують точність розпізнавання.

– Зростання попиту на автоматизацію процесів у різних галузях.
– Впровадження рішень на основі ШІ в мобільні додатки та хмарні сервіси.

Компанії, такі як Google, Amazon, IBM і Mirosoft, активно інвестують у розвиток технологій розпізнавання зображень, пропонуючи рішення для різних секторів економіки. Однак існує потреба в спеціалізовних продуктах, орієнтованих на поліпшення користувацького досвіду та адаптацію до конкретних завдань.

Таким чином, ринок пошуку та розпізнавання зображень на основі ШІ в 2025 році являє собою динамічно розвиваючу сферу з широкими можливостями для інновацій та впровадження нових рішень, спрямованих на задоволення зростаючих потреб користувачів і бізнесу.

1.3 Формулювання технічних вимог

Наразі треба сформулювати технічні вимоги до проєкту для web-застосунку інтелектуального пошуку зображень

– Загальні вимоги:

Інтерфейс користувача: Дуже зрозумілий та простий у використанні (GUI)

Мови програмування: JavaScript (front-end та back-end).

Архітектура: Клієт-серверна архітектура

Серед функціональних вимог важливо визначити внутрішню роботу системи, її поведінку, а саме пошук зображень за запитом користувача, фільтрація за сайтами та форматами файлів зображень, генерація підказок для уточнення пошуку та намірів користувача за допомогою ШІ, швидкий відгук серверу.

Використовуєтсья модель нейронної мережі GPT 4 turbo для достатньо швидкої генерації актуальних підказок (займає близько 2-4 секунд після запиту).

До того ж використовується Google Custom Search API у вигляді пошукової системи котра робить пошук з сайтів з великою кількістю самих різних зображень, з якої ШІ зчитує запит та на його основі генерує підказки для користувача, та швидкість виведення зображень на сайт також достатньо швидка

(приблизно 1-2 секунди).

– Нефункціональні вимоги:

Безпека:

Захист від SQL-ін'єкцій використовується параметризовані запити з mysql2 захист від SQL-ін'єкцій є.

Підтримка API для доступу до функціональності іншими системами: Реалізован REST API (Express), до якого можуть звертатися інші системи та кліенти.

1.4 Проєктування технічної частини web-застосунку для пошуку зображень

1.4.1 Огляд великих мовних моделей ШІ

Станом на 2025 рік великі мовні моделі (LLM - Large Language Models), розроблені ключовими компаніями в галузі штучного інтелекту, стали невід’ємною частиною сучасного цифрового світу. Вони активно застосовуються для створення текстів, обробки зображень, роботи з мультимодальними даними, здійснення пошуку та ведення діалогів з користувачами. Серед найпоширеніших моделей GPT від OpenAI, Gemini від Google, Claude від Anthropic і LLaMA від Meta. Кожна з них вирізняється своїми сильними сторонами, обмеженнями та особливостями реалізації, що визначає їхню ефективність і відповідність тим чи іншим прикладним завданням.

GPT моделі від OpenAI, зокрема GPT 4, GPT 4.5 зарекомендували себе як одні з найпотужніших інструментів у сфері генерації природної мови. Їх сильна сторона - глибоке розуміння контексту, здатність підтримувати тривалі діалоги, відтворювати стиль користувача і навіть генерувати креативні тексти на рівні, що важко відрізнити від людських. Вони здатні адаптуватися до складних сценаріїв, розв’язувати логічні задачі та навіть брати участь у програмуванні. Крім того, GPT-моделі показують чудові результати у тестах на креативність, логіку та мовну побіжність. Але ця ж складність є і недоліком - через глибину моделі і великі розміри вона залишається "чорним ящиком", поведінку якого не завжди

можна передбачити. У практиці це означає, що, попри вражаючу логіку, GPT може надати візуально переконливу, але абсолютно помилкову відповідь. Це явище відоме як "галюцинація моделі". Інша складність полягає у високих обчислювальних витратах - запуск і використання GPT-4 на серверному рівні потребує значних ресурсів, що обмежує її використання на локальних пристроях або в недорогих рішеннях.

Модель Gemini від Google включаючи з останніми поколіннями Gemini 1.5 особливо вирізняється своїм мультимодальним підходом. Вона вміє працювати не лише з текстом, відео, аудіо, а також таблицями і кодом. Завдяки тісній інтеграції з екосистемою Google (Docs, Sheets, Search, YouTube) ця модель є надзвичайно зручною у використанні для тих, хто вже працює у віртуальному середовищі Google. Її особлива сила полягає в розширеній контекстній пам’яті вона здатна опрацьовувати великі обсяги тексту одночасно, що робить її корисною у складних проєктах: підготовці звітів, аналізі даних або складанні резюме. Водночас ця модель часто критикується за більшу закритість, меншу гнучкість у кастомізації для сторонніх розробників і дещо надмірну орієнтацію на бізнес-інтеграції, а не відкриту взаємодію з користувачем. Крім того, як і в випадку GPT, вона може мати труднощі з логічним аналізом, коли стикається з неоднозначними запитами.

Claude, створена компанією Anthropic, є мовною моделлю, яка від початку розроблялася з акцентом на етичність, безпечність і дбайливе поводження з даними. Цей підхід базується на принципах відповідального штучного інтелекту, включаючи зменшення ризику шкоди та максимальне збереження приватності користувача. Claude особливо добре проявляє себе в ролі діалогу як помічник або співрозмовник, вона часто відповідає в делікатному, емоційно інтелігентному стилі, що робить її надзвичайно придатною для освітніх, консультаційних або психологічно чутливих ситуацій. Її відповіді, як правило, звучать більш "людяно" і менш механістично. Проте така обережність має і зворотний бік: модель іноді надає надто обтічні або нерішучі відповіді, особливо в тих випадках, коли очікується конкретна технічна чи чітка оцінка. Крім того, Claude дещо поступається GPT у завданнях, що вимагають глибокої генерації коду або

розв’язання складних математичних задач.

Моделі LLaMA від Meta, які в 2025 році дійшли вже до версії 3.0, фокусуються на відкритості. Вони поширюються переважно з відкритим вихідним кодом, що дає можливість спільноті дослідників, розробників і стартапів вбудовувати їх у власні продукти без залежності від закритих API. Це величезна перевага у сфері наукових досліджень, розробки внутрішніх сервісів у компаніях або створення локальних моделей, які не вимагають відправлення запитів у зовнішнє середовище. Водночас LLaMA моделі часто вимагають додаткового донавчання для досягнення якості, подібної до GPT або Claude. З точки зору UX, ці моделі менш «витончені» і потребують ручного налаштування, що ускладнює їх використання для некваліфікованих користувачів.

Варто згадати ще одну тенденцію - поява спеціалізованих моделей, таких як Mistral, Command R, Falcon та інших, які розробляються з урахуванням вузьких задач (пошук документів, юридичні висновки, робота з науковими статтями). Вони часто мають менший розмір, швидше навчаються, і забезпечують більш стабільну якість у межах своїх ніш. Однак поза межами спеціалізації вони можуть поступатися універсальним моделям. На 2025 рік жодна з LLM не є ідеальною. Кожна з них має сильні та слабкі сторони, пов’язані з дизайном, розмірами, доступністю, комерційною моделлю та метою розробників. Вибір між ними має базуватися на конкретному завданні.У будь-якому разі розвиток великих мовних моделей продовжує змінювати уявлення про автоматизацію, пошук, аналіз і створення контенту, роблячи ШІ дедалі доступнішим і кориснішим.

1.4.2 Обрання ШІ моделі для генерації підказок

GPT 4 turbo - це вдосконалена версія моделі GPT 4 від OpenAI, що поєднує найкращі риси попередніх генерацій, водночас вирішуючи деякі з їх ключових недоліків. Саме ця модель підходить для функції для генерування підказок, оскільки вона надає мені ідеальний баланс між потужністю, точністю та ефективністю.

Основною перевагою GPT 4 turbo є її оптимізована архітектура, яка забезпечує значно швидшу обробку запитів порівняно з базовим GPT 4. Для

проєкту web-застосунку інтелектуального пошуку зображень це дуже важливо, адже користувач сподівається на миттєву відповідь. Завдяки GPT 4 turbo можна забезпечити високий рівень продуктивності без втрати якості результатів. Ще один важливий момент це розширена контекстна пам’ять моделі. GPT 4 turbo здатна працювати з тривалішими запитами та зберігати цілісність контексту протягом великої кількості повідомлень. Модель не лише "запам’ ятовує", що вже було сказано, а й враховує це уподальшихвідповідях, що створює відчуття справжньої інтелектуальної розмови. Крім того, GPT 4 turbo є помітно дешевшою в використанні в порівнянні з попередніми версіями при збереженні тієї ж точності. Модель також демонструє високу креативність, гнучкість у роботі з природною мовою і краще справляється з багатомовними запитами. Це відкиває нові можливості для розширення аудиторії, включаючи користувачів з різних країн та мовних середовищ.

Зрештою, під час проєктування було вибрано GPT 4 turbo тому що ця модель надає технологічну основу,здатне забезпечити не просто функціональність, а й справжню "інтелектуальну взаємодію" між системою та користувачем. Саме цей рівень якості дозволяє створювати не просто інструмент пошуку, а зручного, розумного помічника, який розуміє запити та підказує знайти потрібні зображення швидко та досить коректно.

1.4.3 Інтеграція API OpenAI для створення підказок у web-застосунку

API для генерації підказки дозволяє надіслати запит із текстовим вмістом (пошуковий запит користувача), на основі якого модель gpt 4 turbo генерує рекомендацію для покращення пошуку.

Використання API

URL: http://localhost:3001/api/search/google

Метод: POST

Параметри запиту:

query (string): Пошуковий запит користувача.

images (array): Масив об'єктів з описами знайдених картинок (наприклад, { url, description }).

Параметри запиту до OpenAI:

model (string): Вибір моделі, наприклад, "gpt-4-turbo".

messages (array): Список вхідних повідомлень, кожне з яких має роль (user або assistant) та вміст (content).

max_tokens (integer): Максимальна кількість токенів для генерації.

temperature (number): Рівень випадковості у відповідях (опціонально).

stream (boolean): Використовувати потокову передачу відповідей (опціонально).

system (string): Системний промпт для встановлення контексту (опціонально).

tools (array): Визначення інструментів, які може використовувати модель (опціонально)

Код запиту на мові JavaScript:

```
const { OpenAI } = require('openai');
const openai = new OpenAI({ apiKey: process.env.OPENAI_API_KEY });
exports.generalSuggestions = async (req, res) => {
  const { query, images } = req.body;
  if (!query || !images || !Array.isArray(images) || images.length === 0) {
    return res.status(400).json({ message: 'Нема запросу чи картинок' });
  }
  // Промт для підказок
  const prompt = `
Користувач шукав: "${query}".
Ось приклади знайдених картинок (опис):
${images.slice(0, 10).map((img,i)=>`${i + 1}. ${img.description || ''}`).join('\n')}
Запропонуй 5 порад, як користувач може зробити свій пошуковий запит більш точним і отримати кращі результати. Врахуй початковий запит та описи знайдених картинок. Відповідай списком, без пояснень.
  `.trim();
```

```
try {
  const completion = await openai.chat.completions.create({
    model: "gpt-4-turbo",
    messages: [{ role: "user", content: prompt }],
    max_tokens: 100,
    temperature: 0.7
  });
  const text = completion.choices[0].message.content;
  const suggestions = text
    .split('\n')
    .map(s => s.replace(/^\d+[\).\s-]*/, '').trim())
    .filter(Boolean);
```

Код відповіді:

```
{
  "id": "chatcmpl-abc123",
  "object": "chat.completion",
  "created": 1717777777,
  "model": "gpt-4-turbo",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": "Щоб отримати більш релевантні результати, спробуйте вказати стиль кухні або бажаний колір."
      },
      "finish_reason": "stop"
    }
  ],
  "usage": {
```

```
    "prompt_tokens": 20,
    "completion_tokens": 25,
    "total_tokens": 45
  }
}
```


Основні можливості:

1 Пошук зображень за допомогою Google Search API:

Користувачі можуть шукати зображення за допомогою будь-якого запиту, а результати надходять безпосередньо від Google та інших сайтів з великою кількістю зображень різного стилю .

2 Генерація інтелектуальних пропозицій за допомогою OpenAI GPT-4 Turbo: Після пошуку користувачі отримують поради щодо поліпшення свого запиту, згенеровані штучним інтелектом.

3 REST API для взаємодії між фронтендом і бекендом:

Вся логіка реалізована за допомогою HTTP запитів, що спрощує інтеграцію інших клієнтів або сервісів.

4 Масштабована архітектура (клієнт-сервер):

Фронтенд і бекенд розділені, що дозволяє масштабувати та розвивати проєкт незалежно.

5 Фільтрування та обробка результатів пошуку:

Сервер фільтрує та обробляє результати Google, щоб показувати тільки релевантні та правильні зображення.

1.4.4 Головні функції та властивости API OpenAI для генерації підказок

В проєкті реалізовано класичну взаємодію між клієнтом і сервером за допомогою REST API. Всі основні функції побудовані на стандартних HTTP-запитах, що забезпечує простоту інтеграції, надійність і зручність підтримки. Такий підхід дозволяє легко масштабувати застосунок і підключати до нього різних клієнтів без необхідності реалізовувати складні протоколи обміну даними.

Основні властивості API в моєму проєкті:

1 HTTP-запити:

Взаємодія між фронтендом і бекендом відбувається через стандартні HTTP POST запити.

2 Асинхронна обробка:

Клієнт надсилає запит і отримує відповідь після завершення асинхронної обробки на сервері.

3 Простота інтеграції:

Такий підхід простіше реалізувати і підтримувати, не вимагає додаткової обробки подій на клієнті.

4 Підтримка REST API:

Весь функціонал реалізований через REST API, що зручно для інтеграції з будь-якими клієнтами.

5 Стандартні коди помилок:

У разі помилок сервер повертає стандартні HTTP-коди (наприклад, 400, 500) і повідомлення про помилки в JSON-форматі.

1.5 Опис функціоналу web-застосунку для інтелектуального пошуку зображень

Web-застосунок для інтелектуального пошуку зображень - це інструмент для швидкого та зручного пошуку зображень за будь-якою темою з інтелектуальною підтримкою. Інтерфейс програми дозволяє користувачу виконувати такі дії:

Пошуковий запит:

Введення будь-якого текстового запиту для пошуку зображень (наприклад, "дизайн кухні", "літній сад", "модерн інтер'єр").

Відображення результатів:

Після відправки запиту користувач миттєво отримує добірку релевантних зображень, які відображаються у вигляді адаптивної сітки, зручною для перегляду на різних пристроях.

Інтелектуальні підказки:

Після завантаження зображень система автоматично генерує персоналізовану підказку за допомогою штучного інтелекту (OpenAI GPT-4 Turbo). Підказка допомагає користувачу уточнити та покращити свій пошуковий запит для отримання ще більш релевантних результатів (наприклад, "Уточніть стиль або колір", "Додайте розмір або матеріал").

Миттєва взаємодія:

Всі запити обробляються швидко, а результати та підказки з'являються одразу після пошуку, що забезпечує комфортний користувацький досвід.

Адаптивний інтерфейс:

Сервіс коректно працює як на комп'ютерах, так і на мобільних пристроях, зберігаючи зручність перегляду та керування.

Після введення пошукового запиту та перегляду результатів користувач може скористатися підказкою для формування нового, більш точного запиту, що дозволяє знаходити саме ті зображення, які найбільше відповідають його потребам.

1.6 Обрання технологій для написання front-end та back-end частин програми

Було прийнято рішення розділити front-end та back-end частини для простішої роботи між ними. Обидві частини писати на одній мові програмування JavaScript. Для розробки front-end частини застоснуку використовується технологія React , та для back-end частини використовується Node.js. Вважається що таке поєднання технологій дуже зручне як мінімум тому що можна писати на одній мові програмування JavaScript.

Front-end і back-end розробники співпрацюють для створення цілісного web-застосунку. Розробник серверної частини реалізує API набір інтерфейсів, через які можна взаємодіяти з логікою й даними на сервері. Розробник клієнтської частини використовує ці інтерфейси, щоб отримувати або надсилати інформацію.

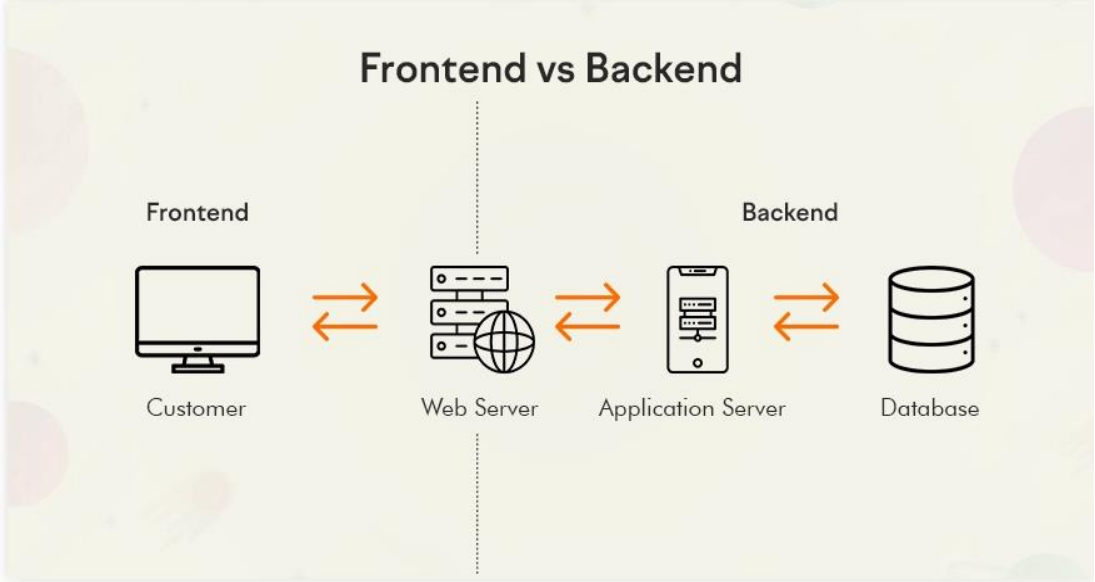


Рисунок 1.1 Взаємодія між front-end та back-end частинами розробки

1.6.1 Опис технологій для розробкии back-end частини

Node.js це середовище виконання JavaScript, побудоване на базі движка V8 від Google Chrome. Завдяки своїм особливостям виконання Node.js набув широкого застосування у веб-розробці. Він використовується для створення масштабованих серверних додатків, REST API, сервісів реального часу, архітектури мікросервісів та інших рішень.

Асинхронна та неблокуюча модель вводу-виводу є однією з голових переваг, Node.js є його асинхронна, керована подіями модель вводу і виводу. Ця архітектура дозволяє обробляти кілька запитів одночасно, не блокуючи потік виконання існуючих запитів. Замість того, щоб чекати завершення однієї операції перед початком наступної, Node.js виконує операції паралельно, що дозволяє обробляти тисячі з'єднань з мінімальним використанням ресурсів. Це робить Node.js ідеальним вибором для розробки високопродуктивних веб-додатків у реальному часі що потребують миттєвого обміну даними. Завдяки подієво-орієнтованій архітектурі, Node.js швидко реагує на вхідні запити, передаючи обробку задач до черги подій, де вони виконуються у фоновому режимі.

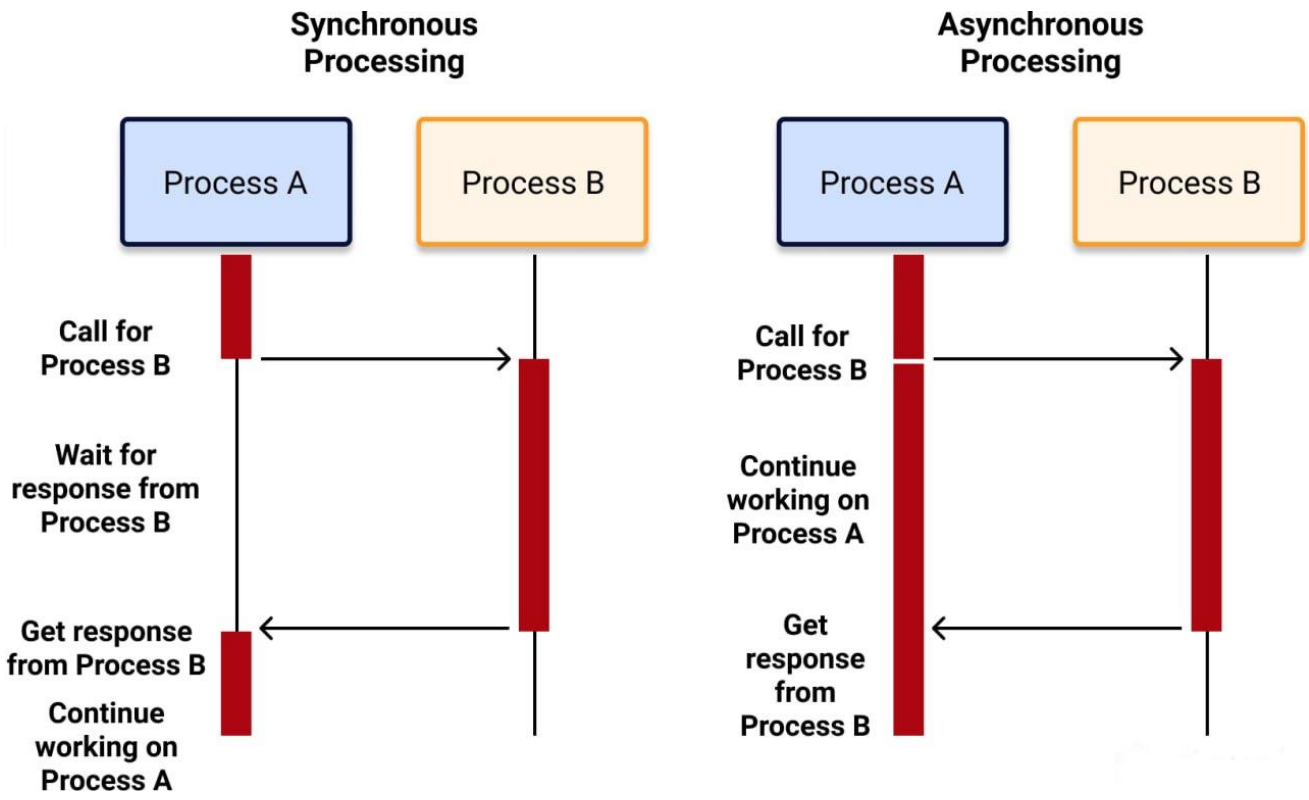


Рисунок 1.2 Порівняння синхронної та асинхронної моделі вводу-виводу

Асинхронна обробка особливо корисна для додатків, які вимагають постійної взаємодії з базами даних, файловими системами або сторонніми API. Це забезпечує високу продуктивність і швидкість реагування навіть при великих навантаженнях.

Використання JavaScript на сервері. Node.js дозволяє використовувати JavaScript не тільки в браузері, але й на стороні сервера. Це дає можливість створювати веб-додатки з єдиним технологічним стеком від фронтенду до бекенду. Це знижує бар'єр входу для нових розробників, спрощує навчання та прискорює розробку. Робота з єдиною мовою на клієнті та сервері спрощує співпрацю в команді, зменшує кількість помилок, пов'язаних з несумісністю логіки, та дозволяє легко обмінюватися кодом між рівнями додатків.

З приводу продуктивності треба зазначити що Node.js працює на движку V8, який компілює JavaScript у машинний код перед виконанням. Це забезпечує високу швикість виконаня коду. Разом з асинхронною обробкою та циклом подій досягається висока ефективність, особливо в роботі з великою кількістю паралельних з'єднань.

Node.js ідеально підходить для додатків, які обробляють дані в режимі реального часу. Висока швидкість обробки запитів і швидка реакція сервера роблять його ефективним рішенням для завдань, де важлива мінімальна затримка. Пакетний менеджер npm є однією з найбільших переваг Node.js є величезна екосистема пакетів npm (Node Package Manager). Це найбільший реєстр відкритих бібліотек JavaScript, що включає мільйони модулів і інструментів для найрізноманітніших завдань.Для того щоб керувати залежностями через npm у нього є відповідні команди:

- npm install - встановлює всі залежності з файлу package.json
- npm install <package> - встановлює конкретний пакет та добавляє його в залежності.
- npm start - запускає основний сервер.
- npm uninstall <package> - видаляє пакет та прибирає його з залежностей.
- npm update - оновлює всі встановлені пакети.
- npm audit -перевіряє проєкт на наявність вразливостей у залежностях.

Існують готові рішення для роботи з базами даних, реалізації автентифікації, створення REST API, обробки файлів, логування, тестування та багато іншого.

Завдяки своїй архітектурі Node.js добре підходить для побудови масштабованих розподілених систем. Він може використовуватися як основа для мікросервісів, де кожна частина системи реалізована як окремий модуль, який взаємодіє через API. Такий підхід спрощує підтримку, оновлення та масштабування окремих компонентів без зупинки всієї системи.

Також є вбудована можливість запускати кілька екземплярів процесу Node.js (через модуль cluster) для використання всіх ядер процесора та підвищення продуктивності.

Підтримка WebSocket та додатків реального часу в Node.js прекрасне рішення для додатків, які вимагають двостороннього зв'язку між клієнтом і сервером у режимі реального часу. Підтримка WebSocket і архітектура яка керується подіями роблять його ідеальною платформою для створення чатів,

онлайн-ігор, систем сповіщень, панелей моніторингу та інших рішень де важлива швидка передача даних.

Завдяки високій продуктивності та здатності обробляти велику кількість одночасних підключень, такі додатки працюють безперебійно навіть під великим навантаженням.В проєкті використовується фреймворк Express.js, цей фреймворк надає широкий набір інструментів для створення серверних застосунків і REST API. Він є одним із найпопулярніших фреймворків в екосистемі Node.js і використовується в більшості серверних застосунків, написаних на цій платформі.

Ключові особливості Express:

– Маршрутизація

Express надає простий та потужний механізм маршрутизації, що дозволяє обробляти HTTP запити різних методів (GET, POST, PUT, DELETE тощо) та прив’язувати їх до певних шляхів URL.

– Middleware

Express використовує концепцію middleware - функцій, які послідовно обробляють запити. За допомогою middleware реалізується автентифікація, логування, валідація даних, обробка помилок та інші задачі.

– Підтримка шаблонізаторів

Можливість використання шаблонізаторів які дозволяють генерувати html-сторінки на стороні сервера.

1.6.2 Обрання середовища для написання та редагування коду

Авжеж потрібно ще й якесь середовище в якому я буду розробляти свій проєкт, та я обираю Visual Studio Code тому ще це актуальний редактор коду, який поєднує в собі швидку роботу та простоту інтерфейсу. Він підтримує безліч мов програмування з комплекту, а за допомогою розширень можна додати підтримку майже будь-якої мови або інструменту. Завдяки вбудованому терміналу розробник може запускати команди, не виходячи з редактора, що прискорює робочий процес.

У Visual Studio Code реалізовано розумне автозаповнення коду,

підсвічування синтаксису і вбудована система налагодження, що робить написання і налагодження коду комфортним і швидким. Інтеграція з системами контролю версій, такими як Git, дозволяє відстежувати зміни і керувати репозиторіями безпосередньо з редактора.

Редактор легко налаштовується під особисті уподобання: можна змінювати теми, комбінації клавіш, встановлювати плагіни і створювати власні шаблони коду. Все це робить VS Code потужним інструментом для розробки, який підходить як новачкам, так і професіоналам. До того ж, редактор безкоштовний і має відкриту архітектуру, що сприяє його безперервному розвитку і появі нових можливостей.

1.6.3 Опис використаних технологій для розробки front-end частини

Front-end це частина програмного продукту, з якою безпосередньо контактує користувач. Він відповідає за зовнішній вигляд, структуру та функціонування інтерфейсу додатка або сайту. Всі елементи, які бачить і використовує людина - кнопки, меню, форми, зображення, анімації належать до front-end частини.

Основне завдання фронтенду це зробити роботу з програмою комфортною, зрозумілою і приємною для користувача. Для цього розробляються візуальні компоненти, реалізуються різні сценарії взаємодії, забезпечується адаптація під різні пристрої та екрани. Фронтенд відповідає за обробку дій користувача, відображення даних, отриманих з сервера, та зворотний зв'язок з користувачем.

Сучасний фронтенд тісно пов'язаний з дизайном і користувацьким досвідом, а також з технологіями, які дозволяють створювати динамічні, інтерактивні та адаптивні інтерфейси. Цей напрямок безперервно розвивається, пропонуючи нові інструменти та підходи для створення ефективних і красивих користувацьких рішень.

Для проєкту були вибрані сучасні, перевірені та ефективні технології, котрі дозволяють створювати зручний, швидкий і масштабований інтерфейс користувача. Основні інструменти, які використовуються: React, React Router, Axios, та ще засоби для стилізації інтерфейсу.

React це сучасна бібліотека для створення інтерфейсів користувача, розроблена компанією Facebook. React був обран тому що він дозволяє будувати складні додатки з великою кількістю інтерактивних елементів, використовуючи компонентний підхід. У React кожен елемент інтерфейсу це окремий компонент, який можна багаторазово використовувати, комбінувати з іншими та легко підтримувати.

Головна перевага React його гнучкість і масштабованість. Завдяки розділенню на компоненти, можна розробляти великі проєкти, не боячись, що код стане заплутаним. Кожен компонент відповідає за свою частину інтерфейсу, має власну логіку та стан. Це спрощує тестування та подальший розвиток застосунку.

Ще однією важливою специфікою React є використання віртуального DOM. Це означає, що при зміні стану компонентів React спочатку оновлює віртуальне дерево елементів, а потім мінімізує кількість змін у реальному DOM. Завдяки цьому додатки на React працюють дуже швидко, навіть якщо на сторінці багато динамічних елементів. Virtual DOM дозволяє швидко визначати, які частини інтерфейсу треба оновити, не чіпаючи весь документ.

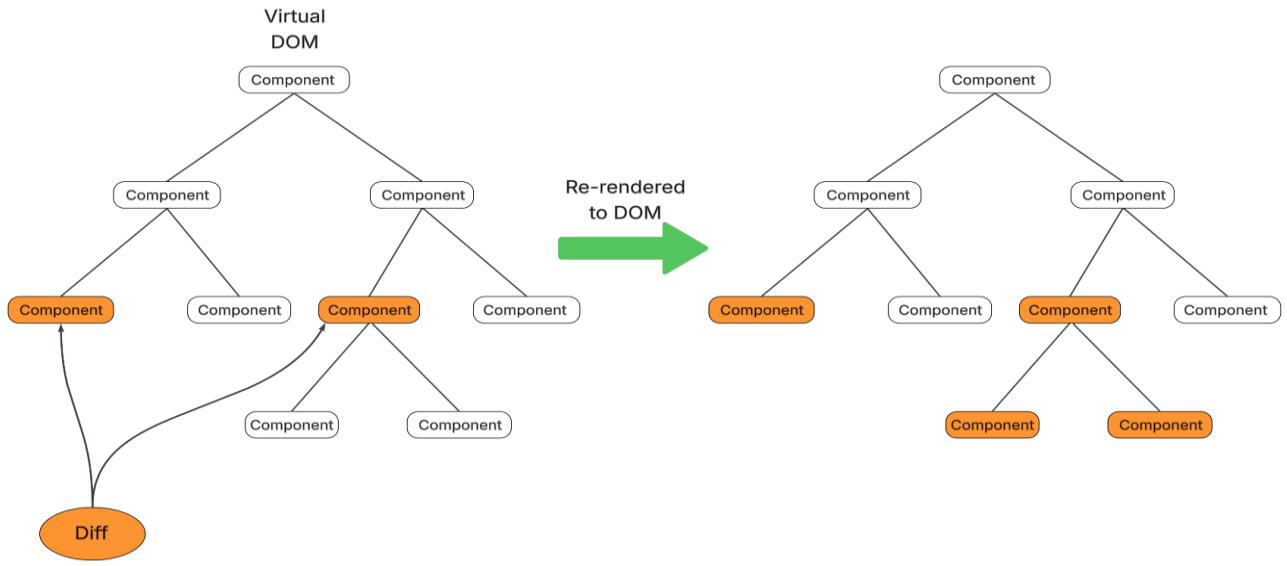


Рисунок 1.3 Структурна схема, яка пояснює працює virtual DOM

React має величезну спільноту розробників, що забезпечує наявність великої кількості бібліотек, інструментів та навчальних матеріалів.

Ще одна корисна перевага React це можливість створювати односторінкові додатки (SPA), де всі переходи між сторінками відбуваються без

перезавантаження сторінки. Це значно покращує користувацький досвід, робить роботу з додатком швидкою та плавною.

Для організації навігації у застосунку я використовую React Router, це спеціальна бібліотека, яка дозволяє створювати багатосторінкові додатки без перезавантаження сторінки. Завдяки React Router можна легко реалізувати переходи між різними розділами, зберігати історію переходів, працювати з параметрами у URL.

React Router інтегрується з React дуже органічно, дозволяючи описувати маршрути як окремі компоненти. Це робить код зрозумілим і структурованим. Наприклад, можна створити окремі сторінки для реєстрації, входу, пошуку, перегляду профілю тощо, і кожна з них буде окремим компонентом, який відображається залежно від поточного маршруту.

Використання React Router значно покращує користувацький досвід, оскільки переходи між сторінками відбуваються миттєво, без повного перезавантаження застосунку. Це особливо важливо для сучасних веб-додатків, де швидкість і плавність роботи мають велике значення.

Ще одна важлива особливість React Router підтримка вкладених маршрутів. Це означає, що можна створювати складні структури сторінок, де одна сторінка містить у собі інші, і кожна з них має власний маршрут. Це дуже зручно для організації великих додатків із багатьма розділами.

Для взаємодії з бекендом було обрано бібліотеку Axios. Axios - це популярний HTTP-клієнт, який дозволяє легко надсилати запити до серверу, отримувати дані, обробляти помилки та працювати з асинхронними операціями. Я використовую Axios для пошуку зображень, отримання порад та інших операцій, які потребують обміну даними з сервером.

Axios має простий і зрозумілий API, підтримує проміси, що дозволяє зручно працювати з асинхронним кодом у React. Крім того, Axios дозволяє налаштовувати заголовки запитів, обробляти токени авторизації, перехоплювати запити та відповіді, що дуже корисно для реалізації захищених додатків.

Ще однією перевагою Axios є його універсальність він працює як у браузері, так і

на сервері, що дозволяє використовувати його у різних частинах проєкту.

У моєму проєкті Axios використовується для всіх основних операцій, пов’язаних із сервером: отримання результатів пошуку, завантаження зображень, отримання порад від штучного інтелекту тощо. Це дозволяє централізовано обробляти всі запити, легко додавати обробку помилок і забезпечувати безпеку передачі даних.

Для створення привабливого та зручного інтерфейсу я використовую стандартний CSS. Це дозволяє повністю контролювати зовнішній вигляд застосунку, робити його адаптивним і сучасним. CSS використовується для визначення кольорів, шрифтів, відступів, розташування елементів, а також для створення анімацій та ефектів.

Особливо велику увагу приділяється адаптивності інтерфейсу, щоб застосунок коректно відображався на різних пристроях комп’ютерах, планшетах, смартфонах. Для цього використовуються медіа-запити, сучасні CSS властивості та тестування інтерфейсу на різних розмірах екранів.

Ще одна важлива складова це кросбраузерність. Треба стежити за тим, щоб застосунок однаково добре працював у різних браузерах, використовуючи сучасні, але підтримувані властивості CSS. CSS дозволяє створювати унікальний дизайн, не обмежуючись готовими шаблонами. Це дає змогу реалізувати будь які ідеї та зробити інтерфейс максимально зручним для користувача.

Всі ці технології органічно поєднуються у сучасній фронтенд розробці. React відповідає за побудову інтерфейсу та управління станом компонентів. React Router забезпечує навігацію між сторінками без перезавантаження. Axios дозволяє взаємодіяти з сервером, отримувати та надсилати дані. CSS відповідає за зовнішній вигляд і адаптивність застосунку.

Коли користувач взаємодіє з додатком (наприклад, натискає кнопку пошуку), React компонент обробляє цю подію, формує запит до серверу за допомогою Axios, отримує відповідь і оновлює інтерфейс. CSS забезпечує привабливий вигляд і зручність використання на будь-якому пристрої.

1.7 Реалізація web-застосунку для інтелектуального пошуку

зображень

1.7.1 Встановлення основних технологій розробки та їх залежностей

Щоб почати новий проєкт з використанням сучасних інструментів, спершу потрібно встановити Node.js і npm. Після цього ви можна буде скористатися командним рядком для створення нового проєкту.

Ось основна послідовність подій:

Встановлення Node.js

Покроково треба встановити Node.js через їх офіційний сайт

1 Переходимо на офіційний сайт Node.js:

Посилання на сайт https://nodejs.org

2 На головній сторінці сайту можна завантажити рекомендовану версію LTS, в залежності від встановленої на комп'ютері операційної системи.

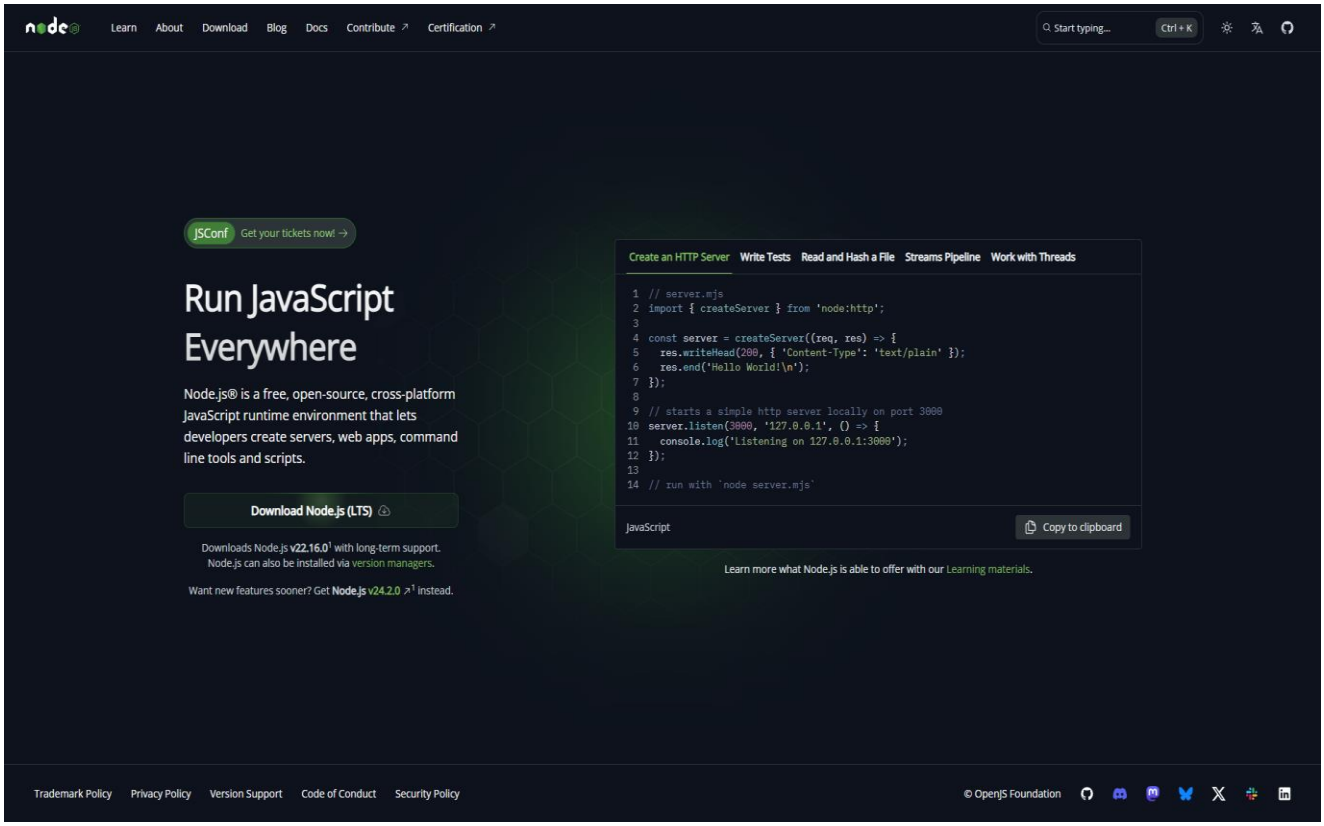


Рисунок 1.4 Офіційний сайт Node.js для завантаження LTS версії

3 Завантаження інсталятору:

Треба натиснути на посилання інсталятора для Windows або macOS в залежності від операційної системи. Це завантажить файл, що запускається (.msi

для Windows і .pkg для macOS).

4 Запуск інсталятору:

Після завершення завантаження треба виконати інсталятор. Це розпочне майстер налаштування.

Для Windows: слідування інструкціям у майстру налаштування. Прийняття угоду про ліцензію, обрання теку для встановлення і потім треба натиснути кнопку "Далі", доки не з’явиться кнопка "Встановити".

5 Перевірка встановлення:

Відкрийте командний рядок (Windows)

Треба ввести *node -v* і натиснути Enter. Це відобразить версію Node.js, яка була встановлена, яка підтвердить встановлення.

Перевірка npm (node package manager) за допомогою введення npm -v та натискання Enter.

Окрім встановлення Node.js з офіційного сайту ще можна завантажити його за допомогою NVM (Node Version Manager):

1 Для встановлення NVM, для Windows потрібно завантажити та інсталювати nvm-windows із сайту https://github.com/coreybutler/nvm-windows/releases

Для macOS або Linux потрібно відкрити термінал і виконати команду:

curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh | bash

Після завершення встановлення треба закрити та знову відкрити термінал щоб почати користуватися nvm.

2 Для перевірки встановлення NVM треба ввести у терміналі команду:

nvm -version

Якщо з’явиться номер версії це означає, що NVM встановлено правильно.

3 Щоб встановити Node.js через NVM, скористайтеся командою:

nvm install -lts

або, якщо потрібно встановити якусь конкретну версію Node.js, можна використати ось таку команду:

nvm install 20.14.0

4 Для перемикання між різними встановленими версіями Node.js використовується команда:

nvm use 18.13.0

5 Щоб переконатися, що Node.js працює, вводиться наступна команда:

node -v

Ці два варіанти встановлення Node.js забезпечують гнучкість, в залежності від того яка версія потрібна для розробки, фіксована або мати можливість перемикатися між декількома версіями за допомогою використання nvm.

Далі потрібно встановити React, ось

- Встановлюється Create React App глобально на комп'ютер з використанням npm:

npm install -g create-react-app

- Для того щоб створити новий проєкт React використовується наступна команда:

  npx create-react-app my-react-project

  Ця команда створить папку з вашим проєктом, у якій вже будуть усі необхідні файли для старту роботи з React.

- Наступним кроком треба зробити перехід в створену директорію:

  cd my-react-project

- Встановлення додаткових залежностей для маршрутизації та HTTP запитів:

npm install react-router-dom axios.

- Наступним кроком є налаштування проєкту:

Створення нової директорії для проєкту, під назвою "backend". Потім треба ініціалізувати проєкт виконавши в даній директорії npm init -y, який створить файл package.json для регулювання залежностями проєкту.

Встановлення залежностей.

Пакети з залежностями, які використовуються в проєкті, пишуться у форматі JSON у файлі package.json "dependencies":

"axios": "^1.9.0",

"bcrypt": "^6.0.0",

"bcryptjs": "^3.0.2",

"cors": "^2.8.5",

"dotenv": "^16.5.0",

"express": "^5.1.0",

"jsonwebtoken": "^9.0.2",

"mysql2": "^3.14.1",

"openai": "^5.0.1",

"sharp": "^0.34.2",

Кожен із цих пакетів з залежностями виконує певну роль у створенні та підтримці веб-сервера на Node.js який використовує Express.js. Короткий опис кожної залежності:

axios - інструмент для виконання асинхронних HTTP запитів до вашого сервера або сторонніх API. Спрощує відправку запитів різних типів і обробку отриманих даних або помилок.

bcrypt - модуль для надійного шифрування паролів за допомогою сучасного алгоритму. Забезпечує захист даних користувачів від несанкціонованого доступу.

bcryptjs - реалізація bcrypt на чистому JavaScript, що не вимагає нативних залежностей. Застосовується для генерації та перевірки хешів паролів, зручна для різних платформ.

cors - проміжне ПЗ для Express, що дозволяє керувати доступом до ресурсів з інших доменів. Важливо для коректної взаємодії frontend і backend, розміщених на різних адресах.

dotenv - модуль для завантаження змінних середовища з файлу .env в застосунок. Дозволяє безпечно зберігати і використовувати секретні дані поза основним кодом.

express - широко використовуваний фреймворк для створення серверних додатків на Node.js. Надає засоби для маршрутизації, підключення middleware і обробки HTTP запитів.

jsonwebtoken - бібліотека для генерації та перевірки токенів JWT

застосовується для реалізації механізмів автентифікації та авторизації користувачів.

mysql2 - драйвер для роботи з базою даних MySQL в Node.js. Підтримує асинхронні операції і сучасні можливості MySQL.

openai - офіційний SDK для інтеграції з API OpenAI. Дає можливість використовувати функції штучного інтелекту, такі як генерація тексту, у вашому застосунку.

sharp - інструмент для серверної обробки зображень: зміни розміру, формату, обрізки та оптимізації. Дозволяє швидко та ефективно працювати з графікою.

server.js

backend > server.js > ...

```
1  const express = require('express');
2  const cors = require('cors');
3  const authRoutes = require('./routes/auth');
4  const searchRoutes = require('./routes/search');
5  const generalSuggestRoutes = require('./routes/generalSuggest');
6  const app = express();
7
8  app.use(cors());
9  app.use(express.json());
10
11 app.use('/api/auth', authRoutes);
12 app.use('/api/search', searchRoutes);
13 app.use('/api/suggest', generalSuggestRoutes);
14
15 const PORT = process.env.PORT || 3001;
16 app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

Рисунок 1.5 Скриншот коду з файлу server.js

У папці backend у файлі *server.js* відбувається основне налаштування серверної частини застосунку на базі Express.js. У цьому файлі підключаються всі необхідні middleware, такі як cors для забезпечення взаємодії між frontend і backend, а також express.json() для обробки JSON-даних у запитах. Далі підключаються основні маршрути, які відповідають за різні функції застосунку: */api/auth* - для аутентифікації користувачів, */api/search* - для пошуку зображень, */api/suggest* - для генерації порад. Кожен із цих маршрутів реалізований у окремому файлі, що забезпечує зручність підтримки та масштабування проєкту. У цьому файлі сервер запускається на вказаному у змінних середовища або стандартному порту, після чого він готовий приймати та обробляти запити від клієнтів.

Інтеграція з API OpenAI

Ця функція відповідає за надсилання даних клієнта до API OpenAI з використанням таких параметрів, як вибір моделі та обмежена кількість токенів у відповіді. Для авторизації застосовується API ключ, який зберігається у безпечному вигляді за допомогою змінних оточення.

```
1   const { OpenAI } = require('openai');
2   const openai = new OpenAI({ apiKey: process.env.OPENAI_API_KEY });
3
4   exports.generalSuggestions = async (req, res) => {
5     const { query, images } = req.body;
6     if (!query || !images || !Array.isArray(images) || images.length === 0) {
7       return res.status(400).json({ message: 'Нема запросу чи картинок' });
8     }
9
```

Рисунок 1.6 Фрагмент коду з файлу generalSuggestController.js в якому перевіряються дані запиту для підказок

OpenAI({ apiKey: process.env.OPENAI_API_KEY })

Створює екземпляр клієнта OpenAI з використанням API-ключа, збереженого у змінних оточення. Це дозволяє безпечно автентифікувати запити до OpenAI.

exports.generalSuggestions = async (req, res) => {...}

Експортує асинхронну функцію, яка приймає HTTP-запит та відповідь. Функція обробляє дані користувача, формує промпт і повертає результат у відповідь клієнту.

const { query, images } = req.body;

Отримує з тіла запиту текст пошукового запиту та масив зображень. Перевіряє, чи ці дані присутні та коректні.

```
  // Промт для підказок
  const prompt = `
Користувач шукав: "${query}".
Ось приклади знайдених картинок (опис):
${images.slice(0, 10).map((img, i) => `${i + 1}. ${img.description || ''}`).join('\n')}

Запропонуй 5 порад, як користувач може зробити свій пошуковий запит більш точним і отримати кращі результати.
Врахуй початковий запит та описи знайдених картинок.Дай відповіді тією ж мовою, якою поставлено запит.Відповідай списком, без пояснень.
  `.trim();
```

Рисунок 1.7 Фрагмент коду з файлу generalSuggestController.js в якому формується промт для генерації підказок

const prompt = ...

Формує текстовий шаблон (промпт) для OpenAI, який містить запит користувача та описи знайдених зображень. Цей промпт використовується для генерації порад.

```
19   try {
20     const completion = await openai.chat.completions.create({
21       model: "gpt-4-turbo",
22       messages: [{ role: "user", content: prompt }],
23       max_tokens: 100,
24       temperature: 0.7
25     });
26
27     const text = completion.choices[0].message.content;
28     const suggestions = text
29       .split('\n')
30       .map(s => s.replace(/^\d+[\).\s-]*/, '').trim())
31       .filter(Boolean);
32
33     res.json({ suggestions });
34   } catch (e) {
35     res.status(500).json({ message: 'Помилка генерації підказок', error: e.message });
36   }
37 };
```

Рисунок 1.8 Фрагмент коду з файлу generalSuggestController.js в якому обробляються відповіді та повертаються підказки

await openai.chat.completions.create({...})

Відправляє запит до OpenAI з промптом, моделлю, лімітом токенів та іншими параметрами. Отримує відповідь із згенерованим текстом порад.

const suggestions = ...

Обробляє отриманий текст: розбиває його на окремі поради, очищає від зайвих символів і формує масив рядків.

res.json({ suggestions });

Відправляє клієнту масив згенерованих порад у форматі JSON.

catch (e) { ... }

Обробляє помилки, які можуть виникнути під час генерації порад, і повертає повідомлення про помилку клієнту.

Запуск серверу

Щоб запустити сервер для цього потрібно використати команду *npm start* у

директорії *backend*. А для того щоб запустити сервер у режимі розробки використовується команда *npm run dev* котра працює завдяки пакету nodemon, який автоматично перезапускає сервер при зміні коду, що зручно під час розробки

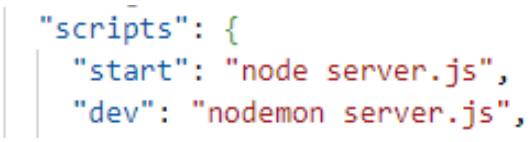


Рисунок 1.9 Скриншот команди для запуску серверної частини проєкту

Для того щоб запустити сервер в режимі розробки треба використати команду “npm run dev”.Після виконання цієї команди в термінали висвітиться ось таке повідомлення

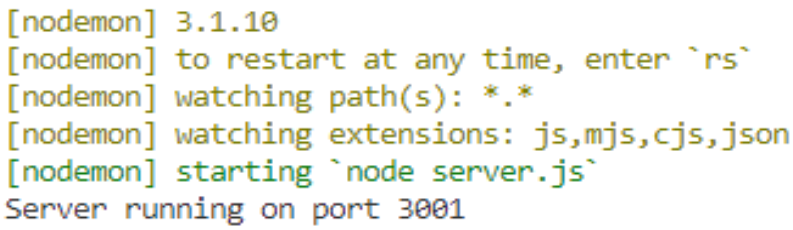


Рисунок 1.10 Скриншот з терміналу на якому показано запуск серверу з nodemon

За для безпеки проєкту у файлі .env будуть зберігатися всі дані які користувачу не потрібно знати

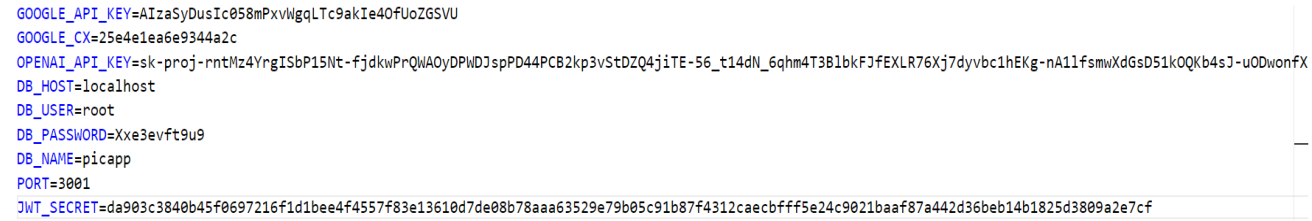


Рисунок 1.11 Скриншот з файлу .env

Для того щоб розпочати працювати над front-end частиною на React, перш за все, потрібно інсталювати Node.js і npm, якщо вони ще не встановлені. Це забезпечить необхідне середовище для роботи з сучасними веб-технологіями.

1 Для створення front-end частини з базовою структурою react використовуйте офіційний інструмент create-react-app. Введіть наступну команду в терміналі:

npx create-react-app my-react-project

За допомогою цієї команди можна автоматично створити нову папку з

назвою my-react-project, яка включатиме всі стандартні файли та конфігурації для проєкту.

Наразі треба перейти до директорії проєкту, для цього використоується наступна команда: *cd my-react-project*

2 Інсталювання залежностей: Після створення стандартної структури на react та переходу в необхідну директорію треба інсталювати всі залежності для проєкту з використанням команди:

npm install <назви залежностей>

3 Запуск сервера для розробки. Щоб запустити локальний сервер розробки та переглянути застосунок у браузері, треба ввести:

npm start

Після введеня цієї команди проєкт буде доступний у браузері за адресою http://localhost:3000

Застережується що ця інструкція є вихідню точкою. Вам можливо знадобиться доповнити свій проєкт ще деякими бібліотеками, плагінами та залежностями (в залежності від вподобань та потреб для проєкту) для того щоб проєкт коректно працював.

Структура проєкту: показана на рисунку в документації, в якій легко можна зорієнтуватись в розподілені файлів та компонентів.

Огляд структури:

Front-end частина на React має структуру з додатковими інструментами та налаштуваннями, які потрібні для покращення процесу розробки.

Опис структури проєкту:

backend:

node_modules: Каталог із залежностями backend, встановленими через npm.

controllers: Містить файли з логікою обробки запитів, наприклад, authController.js, searchController.js, generalSuggestController.js.

routes: Містить файли з описом маршрутів API, наприклад, auth.js, search.js, generalSuggest.js.

server.js: Файл запуску Express сервера, де підключаються middleware та

маршрути.

.env: Файл для зберігання секретних налаштувань (API-ключі, паролі).

package.json: Файл із залежностями, скриптами запуску (start, dev), та іншими метаданими проєкту.

package-lock.json: Файл для фіксації точних версій встановлених залежностей.

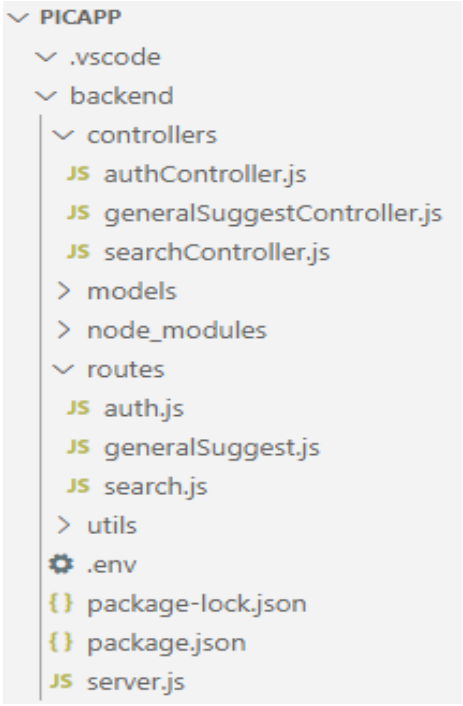


Рисунок 1.12 Скриншот структури back-end частини

frontend:

node_modules: Каталог із залежностями frontend, встановленими через npm.

public:

favicon.ico: Іконка сайту для вкладки браузера.

index.html: Головний HTML-шаблон для React-застосунку.

robots.txt: Файл для налаштування індексації сайту пошуковими системами.

src:

components: Папка з React-компонентами.

App.js: Кореневий компонент React-застосунку.

index.js: Точка входу для ініціалізації React застосунку.

окремі CSS файли: Файли для стилізації інтерфейсу.

package.json: Файл із залежностями (React, React Router, Axios), скриптами запуску (start, build, test), та іншими метаданими.

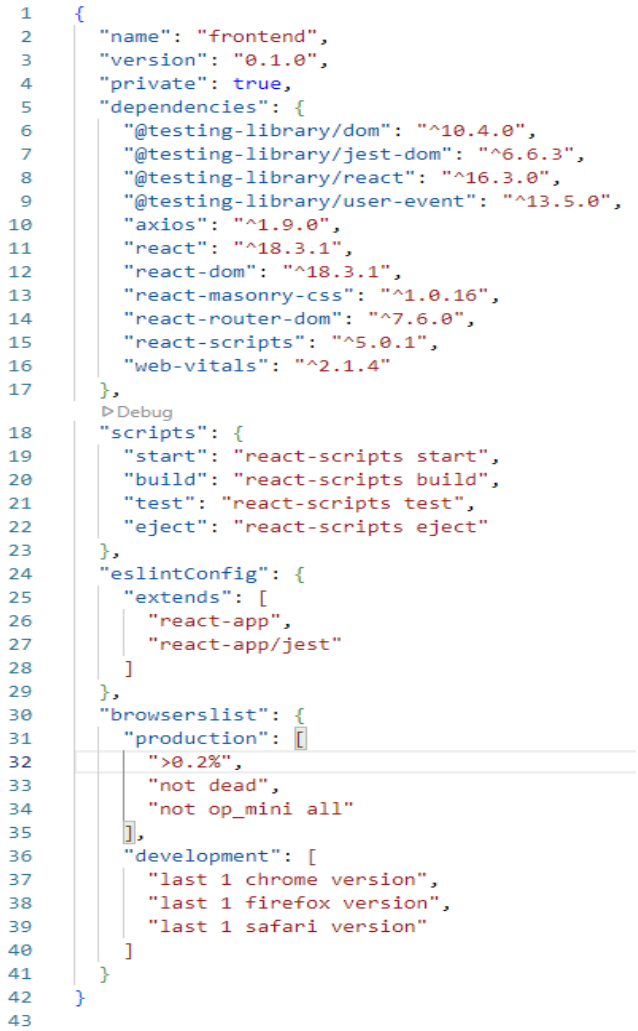
package-lock.json: Файл для фіксації версій залежностей.

Рисунок 1.13 Скриншот структури front-end частини

Розділ "dependencies" у файлі package.json front-end частини містить бібліотеки, які використовуються безпосередньо у проєкті. Для кожної залежності вказується версія, а символ ^ перед номером версії означає, що npm може оновлювати пакет до будь-якої версії, де не змінюється перша значуща цифра (наприклад, ^18.3.1 дозволяє встановлювати до 19.0.0).

Розділ "devDependencies" містить бібліотеки, які потрібні лише під час розробки, інструменти для запуску локального сервера, тестування або збірки. Ці пакети не потрапляють у фінальну збірку для користувача.

У цьому конкретному файлі package.json front-end частини використовується React для створення інтерфейсу, React Router для організації маршрутизації, Axios для виконання HTTP запитів, а також стандартний CSS для стилізації.



```
{
  "name": "frontend",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/dom": "^10.4.0",
    "@testing-library/jest-dom": "^6.6.3",
    "@testing-library/react": "^16.3.0",
    "@testing-library/user-event": "^13.5.0",
    "axios": "^1.9.0",
    "react": "^18.3.1",
    "react-dom": "^18.3.1",
    "react-masonry-css": "^1.0.16",
    "react-router-dom": "^7.6.0",
    "react-scripts": "^5.0.1",
    "web-vitals": "^2.1.4"
  },
  Debug
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

Рисунок 1.14 Скриншот з файлу package.json в папці frontend із залежностями

Також, у проєкті є файл, який використовується як шаблон index.html у папці public front-end частини. Під час запуску React застосунку цей файл слугує основою для всієї сторінки: у нього автоматично підставляється згенерований JavaScript-код, а ще і інші ресурси.

Весь інтерфейс React вшивається у спеціальний елемент * *.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <link href="https://fonts.googleapis.com/css?family=Inter:400,600&display=swap" rel="stylesheet">
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <title>Picapp</title>
  </head>
  <body>
    <div id="root"></div>
  </body>
</html>
```

Рисунок 1.15 Скриншот файлу index.html

Для більшої ясності треба описати структуру данного html документу:
Це базова структура HTML документа, яка використовується як вхідна точка для веб-додатка, створеного за допомогою React.

Декларація *<!DOCTYPE html>*, цей рядок повідомляє браузеру, що сторінка написана з використанням HTML5 актуальної версії мови розмітки для створення веб-сторінок.

Тег <*html*> є коренем HTML документа. Атрибут *lang="en"* визначає,що основна мова сторінки англійська.

Всередині тегу <*head*> знаходяться різноманітні теги meta, link та інші службові елементи, що забезпечують належну роботу та зовнішній вигляд веб-застосунку:

Тег <*meta charset="utf-8"*> відповідає за встановлення кодування символів UTF Це кодування яке охоплює усі символи Unicode, гарантуючи коректне відображення тексту незалежно від обраної мови.

Тег <*link rel="icon" href="%PUBLIC_URL%/favicon.ico"*> призначений для визначення іконки сайту що відображається у вкладці браузера поряд із назвою веб-сторінки.

Тег <linkhref="https://fonts.googleapis.com/css?family=Inter:400,600&display=swap" rel="stylesheet"> виконує підключення зовнішнього шрифту Inter, який використовується для оформлення інтерфейсу. Це дозволяє застосовувати стильний та сучасний шрифт без потреби завантажувати локальні файли.

Тег <*meta name="viewport" content="width=device-width, initial-scale=1"*> використовується для забезпечення адаптивного дизайну. Він забезпечує адаптацію сторінки до ширини екрану пристрою та задає початковий масштаб при завантаженні, що особливо критично для мобільних пристроїв.

Тег <*meta name="theme-color" content="#000000"*> визначає головний колір теми браузера для мобільних пристроїв. В даному випадку чорний. Цей атрибут впливає, зокрема на колір верхньої панелі браузера на пристроях Android.

Тег <*meta name="description" content="Web site created using create-react-app"*> містить короткий опис сайту. Цей опис може відображатися у результатах пошуку або при поширенні посилання на сторінку.

Тег *<link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png">* використовується для встановлення іконки веб-застосунку на головному екрані iPhone чи iPad.

Тег *<link rel="manifest" href="%PUBLIC_URL%/manifest.json">* відповідає за підключення маніфесту PWA (Progressive Web App), де прописані назва застосунку,іконки,колір теми та інші параметри, необхідні для встановлення сайту як застосунку на пристрої.

Тег *<title>Picapp</title>* задає назву веб-сторінки, що відображається у заголовку вкладки браузера або при додаванні сайту в закладки.

У середині тіла документа *<body>:*

Тег * * є контейнером для розміщення усього React застосунку. Після запуску, весь інтерфейс монтується саме в цей елемент за допомогою JavaScript.Також у проекі є файл під назвою index.js котрий являє собою точку входу для проєкту

```
1   import React from "react";
2   import ReactDOM from "react-dom/client";
3   import App from "./App";
4   import "./index.css";
5
6   const root = ReactDOM.createRoot(document.getElementById("root"));
7   root.render(<App />);
```

Рисунок 1.16 Скриншот з файлу index.js котрий є точкою входу

Цей JavaScript-файл слугує точкою входу до React застосунку. Його основне завдання первинне налаштування застосунку, підключення стилів та монтаж головного компонента в HTML документі. У першому рядку відбувається імпорт бібліотеки React, яка необхідна для створення інтерфейсу користувача.
Потім імпортується ReactDOM з пакета react-dom/client. Ця частина бібліотеки відповідає за "прикріплення" React-компонентів до елементів HTML. Тут застосовується метод createRoot, який доступний у версіях React 18 та новіших.

Наступним кроком є імпортування головного компонента застосунку App. Цей компонент є центральним у всьому інтерфейсі, містить ключову логіку, маршрути та інші компоненти. Після цього відбувається імпорт index.css

глобального CSS файлу, де зберігаються стилі, що застосовуються до всього застосунку. Далі формується кореневий вузол React за допомогою команди *createRoot(document.getElementById("root")).* Це вказує, що React працюватиме в межах елемента з id="root", який знаходиться у HTML файлі index.html.

Та на прикінці, виконується рендеринг компонента App а саме застосунок вставляється в DOM та стає видимим для користувача. Цей файл є необхідним для кожного React проєкту, оскільки саме з нього запускається виконання всього коду.

Також у проєкті наявний файл, який можна назвати серцем клієнтської частини застосунку. Саме з нього стартує базова логіка взаємодії користувача з додатком, визначення його автентифікації та відображення відповідного інтерфейсу. Він розроблений як функціональний React компонент, який використовує хук useState для збереження стану користувача, і useEffect для перевірки його автентифікації при запуску застосунку.

```
const db = require('../models/db');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');

const SECRET = process.env.JWT_SECRET || 'dev_secret';

exports.register = (req, res) => {
  const { email, password } = req.body;
  if (!email.endsWith('@gmail.com')) {
    return res.status(400).json({ message: 'Реєстрація дозволена лише для Gmail' });
  }
  db.query('SELECT * FROM users WHERE email = ?', [email], (err, results) => {
    if (err) return res.status(500).json({ message: 'Помилка серверу' });
    if (results.length > 0) return res.status(400).json({ message: 'Користувач вже існує' });
    const hash = bcrypt.hashSync(password, 8);
    db.query('INSERT INTO users (email, password) VALUES (?, ?)', [email, hash], (err, result) => {
      if (err) return res.status(500).json({ message: 'Помилка реєстрації' });
      const token = jwt.sign({ id: result.insertId, email }, SECRET, { expiresIn: '7d' });
      res.json({ token, email });
    });
  });
};

exports.login = (req, res) => {
  const { email, password } = req.body;
  db.query('SELECT * FROM users WHERE email = ?', [email], (err, results) => {
    if (err) return res.status(500).json({ message: 'Помилка серверу' });
    if (results.length === 0) return res.status(400).json({ message: 'Користувач не знайден' });
    const user = results[0];
    if (!bcrypt.compareSync(password, user.password)) return res.status(400).json({ message: 'Невірный пароль' });
    const token = jwt.sign({ id: user.id, email }, SECRET, { expiresIn: '7d' });
    res.json({ token, email });
  });
};

exports.authMiddleware = (req, res, next) => {
  const auth = req.headers.authorization;
  if (!auth) return res.status(401).json({ message: 'Нема токена' });
  const token = auth.split(' ')[1];
  try {
    req.user = jwt.verify(token, SECRET);
    next();
  } catch {
    res.status(401).json({ message: 'Невірный токен' });
  }
};
```

Рисунок 1.17 Скриншот коду з файлу authController.js який реалізує логіку авторизації користувачів

Цей код реалізує функціональність реєстрації, аутентифікації та перевірки авторизації користувачів в застосунку, розробленому на Node.js. Він працює з базою даних MySQL для зберігання інформації про користувачів, застосовує bcrypt для хешування паролів і використовує JWT для формування токенів авторизації.

Процес реєстрації передбачає отримання email та паролю від користувача. Спочатку система перевіряє, чи присутній вказаний email у базі даних. У разі відсутності такого email, пароль хешується, а новий користувач додається до бази даних. Після цього для нього створюється JWT токен, який передається клієнту разом з email. Під час процедури входу здійснюється пошук користувача за вказаним email. Якщо користувача не знайдено або пароль не відповідає (порівнюється з хешем з бази даних), повертається повідомлення про помилку.

При успішному вході також генерується JWT-токен, який відправляється клієнту. Воно перевіряє наявність токена авторизації в заголовку запиту. У разі наявності токену він розшифровується, з нього отримується інформація про користувача та додається до запиту. Якщо токен відсутній або недійсний, генерується помилка доступу.

```
import React, { useEffect, useState } from "react";
import Feed from "./pages/Feed";
import Login from "./pages/Login";
import Register from "./pages/Register";
import axios from "axios";

function App() {
  const [user, setUser] = useState(() => localStorage.getItem("token"));
  const [isLogin, setIsLogin] = useState(true);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    const checkToken = async () => {
      if (!user) {
        setLoading(false);
        return;
      }
      try {
        await axios.get("http://localhost:3001/api/auth/check", {
          headers: { Authorization: `Bearer ${user}` }
        });
        setLoading(false);
      } catch {
        localStorage.removeItem("token");
        setUser(null);
        setLoading(false);
      }
    };
    checkToken();
  }, [user]);

  if (loading) return <div>Завантаження...</div>;

  if (!user) {
    return isLogin
      ? <Login setUser={setUser} setIsLogin={setIsLogin} />
      : <Register setUser={setUser} setIsLogin={setIsLogin} />;
  }

  return <Feed setUser={setUser} />;
}

export default App;
```

Рисунок 1.18 Скриншот коду перевірки токену користувача

На самому початку файлу здійснюється імпортування потрібних модулів: Імпортується React та два базові хуки: *useEffect і useState*. Хуки це спеціальні функції в React, які дозволяють застосовувати стан та інші можливості без потреби створення класів. Імпортуються три компоненти: Feed, Login, Register. Це окремі сторінки чи розділи застосунку, які відображатимуться відповідно до стану аутентифікації користувача.

Імпортується бібліотека axios, яка використовується для здійснення HTTP запитів до серверу.

Після імпортів оголошується функція App, яка є головним компонентом. Всередині цієї функції визначаються три змінні стану за допомогою useState:

1 *user* - відповідає за збереження токена користувача, який зберігається у localStorage. Якщо токен існує, користувач вважається авторизованим.

2 *isLogin* - булева змінна, що визначає, яку форму показувати: вхід чи реєстрацію.

3 *loading* - булева змінна, що контролює, чи завершилася перевірка токена. Доки триває перевірка, на екрані відображається повідомлення про завантаження.

Далі йде блок *useEffect*, що виконується одразу після монтування компонента. Його задача - перевірити, чи є в *localStorage* токен та чи є він дійсним. Якщо токена немає, негайно завершується завантаження *(setLoading(false))*. Якщо токен є, відбувається запит на сервер на маршрут */api/auth/check*, щоб перевірити його дійсність. У випадку помилки токен видаляється з localStorage, і користувач вважається неавторизованим.

Після виконання useEffect, компонент перевіряє, чи ще триває завантаження *(if (loading))*. Якщо так - повертається простий текст "Завантаження...".

Якщо користувач не авторизований, тоді компонент вирішує, що саме показати: форму входу <*Login* /> або форму реєстрації <*Register* />. Це визначається за значенням змінної *isLogin*. Обидва ці компоненти отримують два пропси - *setUser* для встановлення нового токена користувача після успішного входу або реєстрації, і *setIsLogin* для перемикання між формами.

Якщо користувач авторизований компонент повертає головну частину

інтерфейсу компонент Feed, який є основним функціоналом застосунку. Він також отримує функцію setUser, яка наприклад може застосовуватися для виходу користувача з системи.

Ну і в кінці функція App експортується за допомогою export default App, що дозволяє застосовувати її в інших частинах проєкту, зокрема в index.js, де вона монтується у HTML документ.

1.8 Опис роботи проєкту

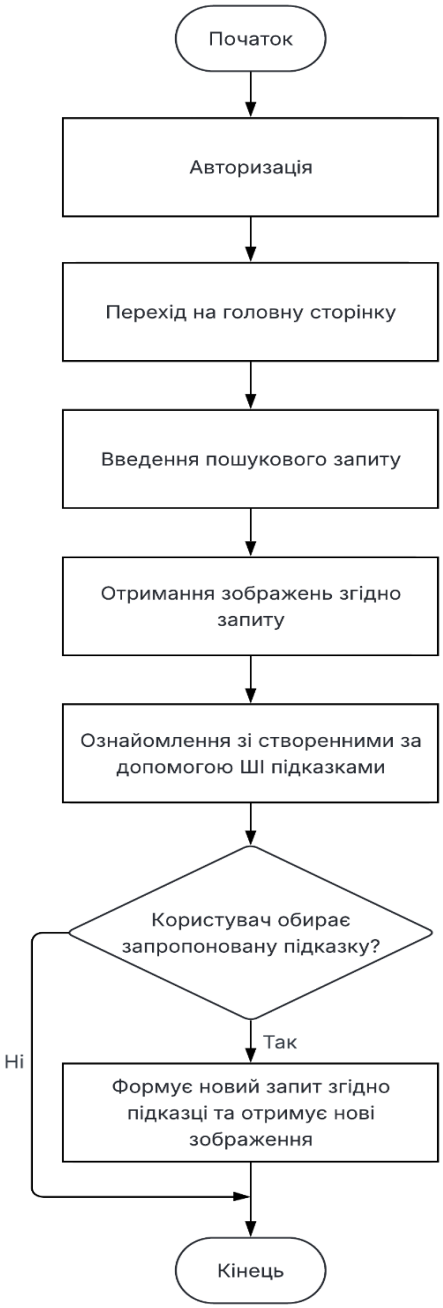


Рисунок 1.19 Блок схема алгоритма для користування web-застосунком

1 Сторінка реєстрації

На цій сторінці новий користувач може створити обліковий запис. Форма містить поля для введення пошти та пароля. Після заповнення форми і натискання кнопки "Зареєструватися" дані надсилаються на сервер. Якщо реєстрація проходить успішно, користувач автоматично авторизується і переходить до основного інтерфейсу. Якщо виникає помилка (наприклад, пошта вже використовується), відображається відповідне повідомлення.

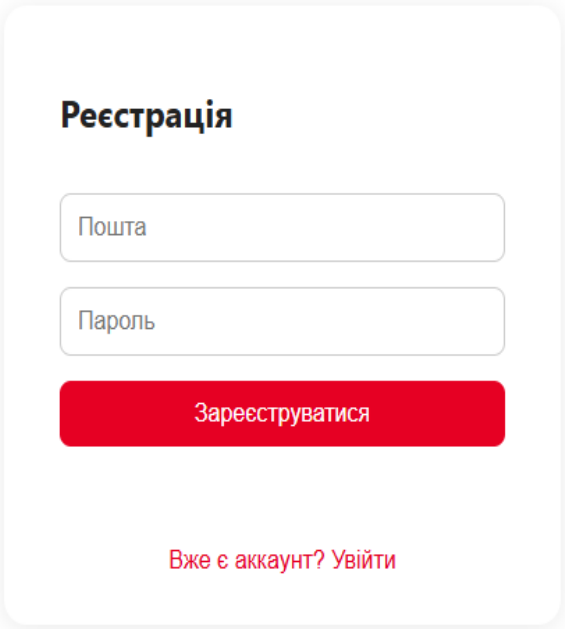


Рисунок 1.20 Скриншот сторінки реєстрації

2 Сторінка входу в аккаунт

Ця сторінка призначена для авторизації вже зареєстрованих користувачів. Форма містить поля для введення пошти та пароля. Після натискання кнопки "Увійти" дані перевіряються на сервері. У разі успішної авторизації користувач отримує доступ до основної сторінки сайту але якщо дані не вірні з’являється помилка

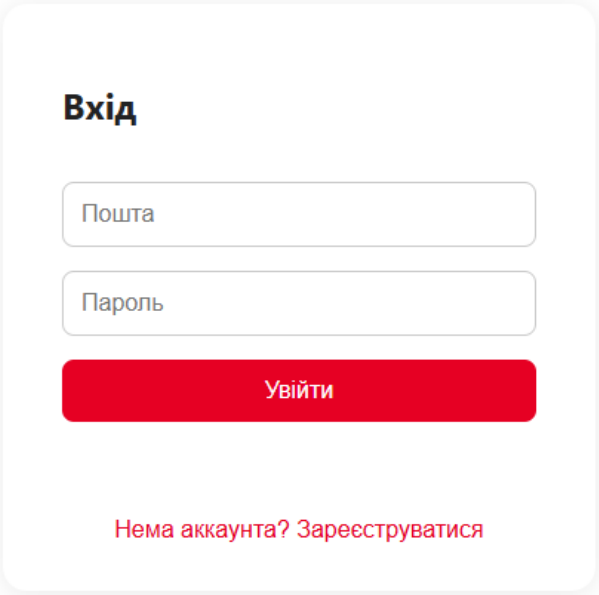


Рисунок 1.21 Скриншот зі сторінки для входу в аккаунт

3 Основна сторінка

Після входу користувач потрапляє на головну сторінку сайту. Тут доступна панель пошуку зображень, стрічка знайдених картинок (після вводу запиту), а також підказки для пошуку. Є можливість вийти з облікового запису через відповідну кнопку. Інтерфейс організований для зручного перегляду та взаємодії з контентом.

Основні елементи інтерфейсу:

Пошуковий рядок - текстове поле, в яке користувач вводить свій запит для пошуку зображень.

Кнопка пошуку: Активує процес пошуку зображень за введеним запитом.

Підказка: Після виконання пошуку може з’являтися блок із порадою щодо покращення пошукового запиту, згенерованою на основі результатів.

Стрічка зображень: Основна частина інтерфейсу, де у вигляді сітки відображаються знайдені зображення, що відповідають запиту користувача.

Кнопка виходу: Дозволяє користувачу завершити сесію та вийти з облікового запису.

Відправлення даних:

Після введення запиту та натискання кнопки пошуку, дані надсилаються на сервер, який обробляє запит, отримує зображення та повертає їх для відображення у стрічці.

Якщо пошук успішний, тоді у стрічці з’являються відповідні зображення. Якщо зображення не знайдено, то тоді відображається відповідне повідомлення. Підказка допомагає користувачу сформулювати більш точний запит для отримання кращих результатів.

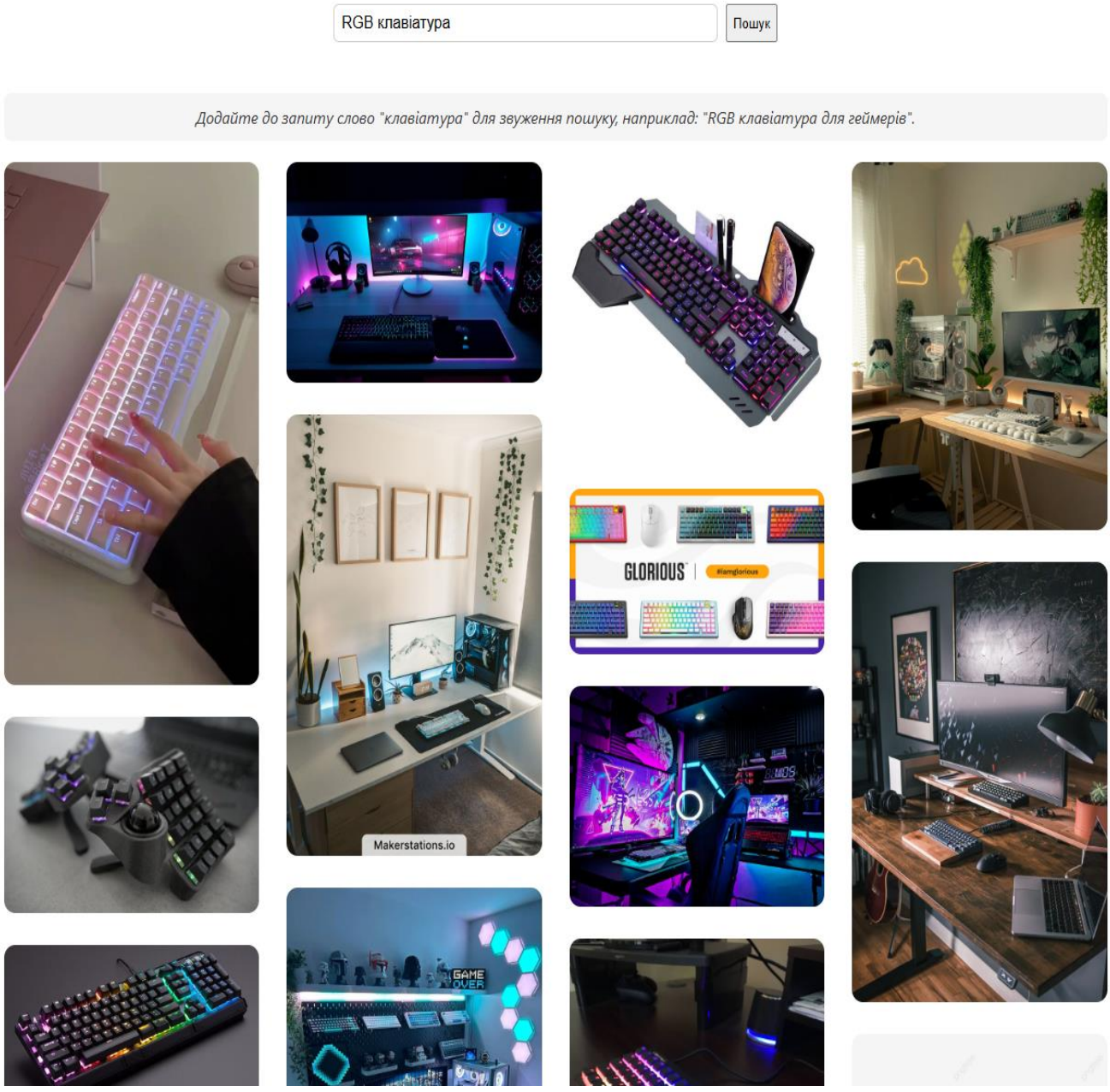


Рисунок1.22 Скриншот відповіді запиту користувача

2 ЕКОНОМІЧНИЙ РОЗДІЛ

В дипломному проєкті створений web-застосунок для інтелектуального пошуку зображень.

Особливістю застосунку є можливість отримувати зображення за допомогою запиту, та на його основі клієнт отримує підказки в залежності від запиту згенеровані штучним інтелектом, що робить роботу з сервісом більш ефективною та зручною.

Розрахунок економічної ефективності розробки створеного web-застосунку не розраховуємо в зв'язку з тим що дана розробка не є комерційною.

Загальні витрати ($В_з$) на створення застосунку складаються з декількох параметрів:

$$В_з = В_р + В_в + В_е \ ,$$

де $В_р$ - витрати на розробку застосунку;

$В_в$ - витрати на впровадження застосунку;

$В_е$ - витрати на експлуатацію застосунку;

Витрати на розробку застосунку ($В_р$) є одноразовими та складаються з вартості наступних видів робіт зі створення застосунку :

1 Розробка дизайну застосунку: розробка макетів дизайну для головної та внутрішньої сторінок застосунку; розробка логотипу

2 Наповнення застосунку інформацією: наповнення та форматування web-сторінок; обробка малюнків для публікації на web-сторінках, верстка (переклад в HTML-формат) web-сторінок

3 Програмна розробка застосунку: створення програмного коду застосунку

4 Налаштування модулів: модуля e-mail форм, модуля фотогалереї, модуля на багато користувачів доступу до системи

Для визначення витрат на розробку застосунку ($В_р$) розраховуємо оплату праці виконавців, безпосередньо притягнених до її виконання. Для реалізації проєкту Web-системи використовуються наступні спеціалісти: Розробник-Дипломник

Для визначення трудомісткості розробки застосунку (B_p) складено план-графік по розробці web-застосунку і тривалості виконання робіт. Розподіл робіт по етапах і видах виконавців наведено в таблиці 2.1.

Таблиця 2.1 - План-графік по розробці Web-застосунку

№	Назва етапу	Час виконання (годин)	Посада виконавця
1	Аналіз ринку	8	Розробник-Дипломник
2	Формулювання технічних вимог	8	Розробник-Дипломник
3	Проєктування технічної частини	16	Розробник-Дипломник
4	Вибір моделі та розробка функіоналу API ШІ для генерації підказок	24	Розробник-Дипломник
5	Тестування та виправлення помилок генерації підказок	40	Розробник-Дипломник
6	Розробка функціоналу web-застосунку пошуку зображень	36	Розробник-Дипломник
7	Розробка інтерфейсу та серверної частини web-застосунку	56	Розробник-Дипломник
8	Тестування web-застосунку	16	Розробник-Дипломник
9	Виправлення недоліків	16	Розробник-Дипломник
	ВСЬОГО:	220	

Розрахунок трудомісткості здійснений в наступній послідовності:

1 Складений перелік всіх етапів і видів робіт, які необхідно виконати в ході даної розробки. Після узгодження з керівником проєкту допущено виключення, доповнення, об'єднання окремих етапів і видів робіт;

2 По кожному виду робіт визначений кваліфікаційний рівень виконавців. В разі виконання однієї роботи виконавцями різної кваліфікації, робота розподілена на ряд паралельних конкретних робіт для кожної категорії виконавця.

В умовах відсутності нормативної бази тривалість виконання окремих робіт розраховуємо на основі вірогідних оцінок робіт, що задані розробником.

Розмір заробітної плати розраховується виходячи з чисельності різних категорій виконавців, трудомісткості, що витрачається ними на виконання різних видів робіт, а також їх середньої заробітної плати (ставки) за годину (або один робочий день).

При визначенні вартості виконуваних робіт орієнтуємося на мінімальну заробітну плату, встановлену Відповідно до «Закону про Державний бюджет України» (станом на 01 січня 2025 року), враховуючи кваліфікацію виконавців, Витрати на заробітну плату приведені в таблиці 2.2 (мінімальна заробітна плата в місяць - 8000 грн; в годину - 48 грн).

Таблиця 2.2 - Витрати на заробітну плату

№	Персонал	Етапи розробки	Кількість робочих годин (або днів)	Погодинна ставка (або денна ставка), грн.	Заробітна плата, грн.
1	Дипломник	Аналіз ринку	8	48	384
2	Дипломник	Формулювання технічних вимог	8	48	384
3	Дипломник	Проєктування технічної частини	16	48	768
4	Дипломник	Вибір моделі та тестування API ШІ для генерації підказок	24	48	1152

Кінець таблиці 2.2

5	Дипломник	Розробка функціоналу генерації підказок	40	48	1920
6	Дипломник	Розробка функціоналу web-застосунку пошуку зображень	36	48	1728
7	Дипломник	Розробка інтерфейсу та серверної частини web-застосунку	56	48	2688
8	Дипломник	Тестування web-застосунку	16	48	768
9	Дипломник	Виправлення недоліків	16	48	768
				ВСЬОГО:	$В_{зп}$ =10560

До складу витрат на оплату праці також включаються податки, збори і інші обов'язкові платежі, встановлені системою оподаткування що діє. Розмір єдиного соціального внеску складає 22% від заробітної плати, розраховується за наступною формулою:

$$В_{єсв} = 10560 \times 0,22 = 2323,2 \text{ грн.}$$

Загальні витрати ($В_р$) на розробку веб-застосунку розраховуються як сума витрат на заробітну плату праці персоналу ($В_{зп}$) та єдиного соціального внеску ($В_{єсв}$):

$$В_р = 10560 + 2323,2 = 12883,2 \text{ грн.}$$

Витрати на впровадження застосунку ($В_в$) складаються з двох складових:

- витрати на реєстрацію доменного імені на 1 рік ($462_{в1}$);

так як реєстрація буде в пошуковій системі Google тому витрати на впровадження застосунку у пошуковій системи не розраховуємо;

$$В_в = 462 \text{ грн.}$$

Витрати на експлуатацію застосунку (Ве) включають вартість робіт з підтримки застосунку в робочому стані і вартість послуг по продовженню доменного імені на 1 рік.

Роботи по підтримці застосунку в робочому стані включають в себе:

1 Оновлення даних на сайті;

2 Налаштування параметрів сервера хостингу;

3 Забезпечення щомісячного захисту застосунку;

Підтримка застосунку в робочому стані буде здійснювати сама організація яка виконує ці види робіт за договором. У таблиці 2.3 визначаються постійні витрати як сума витрат на впровадження та експлуатацію застосунку протягом року.

Таблиця 2.3 - Постійні витрати

№	Стаття витрат	Вартість за рік, грн.
1	Оновлення даних у застосунку	18000
2	Налаштування параметрів сервера хостингу	2842
3	Забезпечення щомісячного захисту застосунку	15000
	Всього:	$В_{пост}$ = 35842

Загальні витрати ($В_з$) на розробку, впровадження та експлуатацію веб-застосунку розраховуються за наступною формулою:

$$В_з = В_р + (В_в + В_е) = 12883,2 + 462 + 35842 = 49187,2$$

Економічна ефективність так як застосунок не є комерційним економічну ефективність не розраховується

Функціональна ефективність: покращення релевантності пошуку, економія часу, розумна автоматизація, краща UX навігація

Соціальна ефективність, доступ до знань, розвиток творчості, зниження бар'єрів у використанні сучасних технологій

3 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХІКИ БЕЗПЕКИ

Охорона праці на виробництві завжди мала першорядне значення, саме завдяки рекомендаціям з охорони праці, персонал, який працює на підприємстві, формує алгоритм виконання робочих задач з чітким дотриманням інструкцій.

Основне завдання охорони праці це створення та реалізація заходів, спрямованих на забезпечення безпеки життя, збереження працездатності та здоров'я людини під час трудової діяльності.

При роботі з комп'ютером, як і в багатьох інших сферах, необхідно враховувати нормативи освітлення, температури, відносної вологості та інтенсивності вібрації. Але при роботі в приміщенні з комп'ютерами найважливішим є дотримання правил пожежної безпеки, включаючи вогнестійкість приміщення, а ще й рівень звукового шуму та характеристики електромагнітних, ультрафіолетових і інфрачервоних полів.

Для аналізу охорони праці в дипломному проєкті досліджується безпека праці розробника веб-сторінок в офісному приміщенні.

3.1 Аналіз та безпека умов праці працівника на робочому місці

Під час виконання будь-яких видів роботи за комп'ютером, на працівника можуть впливати небезпечні фактори виробничого середовища, зокрема: фізичні та психофізіологічні небезпечні та шкідливі виробничі фактори.

Серед фізичних небезпечних факторів, найпоширенішими є підвищена температура повітря робочої зони, підвищений рівень шуму, знижена вологість повітря це типові фактори, які виникають при роботі в приміщеннях з комп'ютерами, через їхню роботу на робочому місці підіймається температура та знижується вологість повітря. Крім цього, комп'ютер випромінює електростатичні та електромагнітні поля в діапазоні від 5 Гц до 2 кГц та від 2 до 400 кГц, тому робота за комп'ютером включає підвищені показники електромагнітного випромінювання та статичної електрики. У службових приміщеннях не завжди є достатня кількість природного сонячного світла, тому присутня велика кількість штучного освітлення, яке в свою чергу не завжди

належно відрегульоване, з цього виникає, що світло може бути недостатньо інтенсивним або надмірно яскравим.

Важливим аспектом охорони праці в ІТ сфері є профілактика професійних захворювань. Тривала робота за монітором може викликати зорове напруження, головний біль, порушення осанки, тунельний синдром (ущемлення нерва зап'ястя) тощо. Тому вкрай важливим є правильне організаційне планування робочого місця, чергування праці та відпочинку (регламентовані перерви), застосування ергономічних меблів і достатнє освітлення (500 люкс при роботі з текстами та 300 люкс при роботі з графікою.

3.2 Розробка заходів з охорони праці

3.2.1 Виробниче освітлення

Штучне освітлення в приміщеннях з робочими місцями, обладнаними відеодисплейними терміналами, має здійснюватися системою загального рівномірного освітлення. У виробничих та адміністративно-громадських приміщеннях, у разі переважної роботи з документами, дозволяється застосування системи комбінованого освітлення (додатково до системи загального освітлення, встановлюються світильники місцевого освітлення).

3.2.2 Мікроклімат

При роботі в приміщеннях з великою кількістю комп’ютерів, приміщення яких класифікуються як приміщення з підвищеною небезпекою електротравм, температура повітря влітку може перевищувати 35°С, що негативно впливає на здоров’я людини, тому в таких приміщеннях повітря повинно охолоджуватися, а знижена вологість повітря повинна регулюватися спеціальним обладнанням.

Відповідно до норм ДСН3.3.6.042-99 температура повітря в офісі повинна становити 22-25°С, вологість повітря 40-60%, швидкість руху повітря не більше 0,1 м/с. Якщо ці норми перевищено, робочий день працівника повинен бути скорочений на 10%.

3.2.3 Шум

Під час інтелектуальної роботи, котра потребує концентрації, допустимий рівень шуму дорівнює 50дБ. Щоб зменшити шум і вібрацію в кімнаті, обладнання, апарати та прилади розміщують на спеціальних амортизуючих прокладках. Якщо стіни в приміщенні генерують шум, їх необхідно облицювати звукопоглинальними матеріалами.

3.3 Організація робочого місця користувача ПК

Конструкція робочого місця користувача ПК та взаємне розташування всіх його елементів (сидіння, органи керування, засоби відображення інформації) відповідають антропометричним, фізіологічним і психологічним вимогам, а також характеру роботи. Конструкція робочих меблів повинна забезпечувати можливість індивідуального регулювання відповідно до росту працюючих для підтримки зручної пози. Робочий стіл повинен бути пофарбований матовою фарбою. Дисплей розташований так, що його верхній край перебуває на рівні очей на відстані близько 70 см, що укладається в припустимі рамки від 60 до 90 см. Частота мерехтіння екрана fмер = 100 Гц, що відповідає умові fмер > 70 Гц.

Робоче місце розташоване перпендикулярно віконним прорізам, це зроблено з метою виключити пряму й відбиту мерехтливість екрана від вікон і приладів штучного освітлення.

Згідно з темою дипломного проєкту, робоче місце програміста укомплектовано пристроями з електромагнітним випромінюванням.

3.4 Пожежна безпека

Забезпечення пожежної безпеки на об’єкті праці є важливою частиною роботи зі створення безпечних та здорових умов праці.

Прохід до аварійних виходів повинен бути вільний, шириною не менше 1 метра, у разі великої кількості горючих відходів необхідно використовувати відведені сміттєзбірники. Електроприлади повинні використовуватися тільки за

їхнім прямим призначенням, а у разі пошкодження приладів, слід вимкнути їхнє живлення та привести до пожежнобезпечного стану.

Первинні засоби пожежогасіння застосовуються для боротьби з пожежами на початковій стадії. До них належать: пожежні кран-комплекти, вогнегасники, пожежний інвентар (резервуари з водою, ящики з піском, пожежні відра, лопати), а також різний переносний пожежний інструмент (кирки, сокири, багри, ломи тощо).

Для гасіння невеликих осередків пожеж використовуються порошкові вогнегасники, які вважаються за ефективністю пожежогасіння, економічністю та іншими показниками більш перспективними. Засоби для первинного пожежогасіння представлено на рисунку 3.1:

Рисунок 3.1 Засоби пожежогасіння

На кожному поверсі будинку адміністративного призначення розміщено менше двох вогнегасників з масою заряду вогнегасної речовини 5 кг і більше. Експлуатація вогнегасників без призначення відповідального за організацію цієї роботи не допускається.

Первинні засоби пожежогасіння розміщують на пожежних щитах, які встановлюються на виробничій території з розрахунку один щит на 5000 м2. Вони фарбуються у червоний колір, вимоги до червоного кольору визначено в державних стандартах з пожежної безпеки. Це потрібно для того, щоб робітники будь-якого підприємства мали змогу без зусиль впізнати пожежний щит.

Забороняється палити на підприємстві, окрім спеціально відведених для цього місць, забороняється зберігати легко займисті матеріали такі як папір, ближче ніж 1 метр від електрощитів, 0,15 метрів від приладів центрального

водяного опалення та, від сповіщувачів автоматичної пожежної сигналізації, також документація повинна зберігатися у спеціально відведених для цього шафах.

Для запобігання поширенню пожежі встановлюють протипожежні системи, які складаються з датчиків, звукових сповіщувачів, аварійних кнопок, приймально-контрольної панелі, яка виступає як аналізатор інформації, яку отримують датчики, і відправляє ці дані на пульт пожежної охорони.

Протипожежна сигналізація призначена для виявлення пожежі на початковому етапі.

У разі, якщо пожежі не вдалося уникнути, необхідно:

1 терміново повідомити пожежну охорону по телефону 101, вказати при цьому адресу, кількість поверхів, місце виникнення пожежі, наявність людей, своє прізвище;

2 організувати евакуацію людей та матеріальних цінностей;

3 повідомити про виникнення пожежі адміністрацію та чергового (за його наявності);

4 вимкнути, у разі необхідності, струмоприймачі та вентиляцію;

5 розпочати гасіння пожежі наявними первинними засобами пожежогасіння;

6 організувати зустріч підрозділів пожежної охорони й надати їм консультаційну та іншу допомогу в процесі гасіння пожежі.

Навчання з охорони праці працівників також є обов'язковим. Відповідно до Постанови Кабінету Міністрів України від 26.06.2013 № 333, всі працівники при прийомі на роботу, а також не рідше одного разу на три роки (або щороку - для робіт підвищеної небезпеки) повинні проходити навчання та перевірку знань з питань охорони праці.

У випадку виявлення порушень у сфері охорони праці, роботодавець несе відповідальність згідно з Кодексом України про адміністративні правопорушення (стаття 41, 188-4) або навіть кримінальну відповідальність у разі тяжких наслідків.

ВИСНОВКИ

В процесі роботи над дипломним проектом розроблено веб-застосунок для інтелектуального пошуку зображень, який базується на сучасних веб-технологіях та штучному інтелекті.

Під час реалізації проведено аналіз існуючих рішень у галузі пошуку візуального контенту, визначено їхні сильні та слабкі сторони. Це дозволило створити продукт, що відповідає вимогам для створення конкуренції на ринку та інноваційності.

Застосунок реалізован за клієнт-серверною архітектурою з використанням Node.js для бекенду та React для фронтенду. Такий вибір забезпечує гнучкість, легкість масштабування та зручність в підтримці. Інтеграція API OpenAI та GPT 4 Turbo дозволила реалізувати механізм генерування інтелектуальних підказок для пошукових запитів користувачів, що покращує точність і відповідність результатів.

Розроблено REST API для взаємодії клієнта та сервера, що дозволяє інтегрувати систему з іншими сервісами. Впроваджено Google Search API для швидкого пошуку та обробки зображень із застосуванням штучного інтелекту.

Результати роботи над проектом підтверджують ефективність використання обраних технологій, зокрема, GPT-моделей для генерації підказок. Проект демонструє перспективи розвитку таких систем з урахуванням зростання важливості візуального контенту.

Отримані знання та розроблені рішення можуть стати основою в майбутньому для подальшого вдосконалення web-застосунку або розробки нових застосунків із використанням штучного інтелекту.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1 Тарасов О.І. Основи веб-розробки: навчальний посібник. – Київ: КНЕУ, 2021. – 224 с.;

2 Олійник А.В. Програмування вебзастосунків з використанням JavaScript. – Львів: ЛНУ ім. Івана Франка, 2020. – 180 с.;

3 Громов В.В. Web-програмування: HTML, CSS, JavaScript. – Київ: Центр учбової літератури, 2019. – 368 с.;

4 Джонсон Б. React. Up & Running: Building Web Applications. – O'Reilly Media, 2018. – 258 p. – ISBN 978-1-4919-5328-9;

5 Banks A. Learning React: Modern Patterns for Developing React Apps. – 2nd Edition. – O'Reilly Media, 2020. – 350 p. – ISBN 978-1-4920-1832-2.;

6 Valerio De Sanctis. ASP.NET Core and React. – Packt Publishing, 2021. – 712 p. – ISBN 978-1-80056-116-8.;

7 Mernagh B. Node.js Web Development: Server-side development with Node 18 using modern JavaScript techniques. – Packt Publishing, 2022. – 712 p. – ISBN 978-1-80324-570-6.;

8 Батченко А.С. Системи штучного інтелекту: підручник. – Київ: КНУ, 2021. – 298 с.;

9 Russell S., Norvig P. Artificial Intelligence: A Modern Approach. – 4th Edition. – Pearson, 2021. – 1136 p. – ISBN 978-0-13-461099-3.;

10 Documentation – OpenAI API [Електронний ресурс]. – Режим доступу: https://platform.openai.com/docs

11 Google Custom Search JSON API [Електронний ресурс]. – Режим доступу: https://developers.google.com/custom-search

12 React Documentation [Електронний ресурс]. – Режим доступу: https://react.dev/

13 Node.js Documentation [Електронний ресурс]. – Режим доступу: https://nodejs.org/en/docs

ДОДАТОК А. Лістинг коду основних функцій web-додатку на мові JavaScript

Код authController.js

```
const db = require('../models/db');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');

const SECRET = process.env.JWT_SECRET || 'dev_secret';

exports.register = (req, res) => {
  const { email, password } = req.body;
  if (!email.endsWith('@gmail.com')) {
    return res.status(400).json({ message: 'Реєстрація дозволена лише для Gmail' });
  }
  db.query('SELECT * FROM users WHERE email = ?', [email], (err, results) => {
    if (err) return res.status(500).json({ message: 'Помилка серверу' });
    if (results.length > 0) return res.status(400).json({ message: 'Користувач вже існує' });
    const hash = bcrypt.hashSync(password, 8);
    db.query('INSERT INTO users (email, password) VALUES (?, ?)', [email, hash], (err, result) => {
      if (err) return res.status(500).json({ message: 'Помилка реєстрації' });
      const token = jwt.sign({ id: result.insertId, email }, SECRET, { expiresIn: '7d' });
      res.json({ token, email });
    });
  });
};

exports.login = (req, res) => {
  const { email, password } = req.body;
  db.query('SELECT * FROM users WHERE email = ?', [email], (err, results) => {
    if (err) return res.status(500).json({ message: 'Помилка серверу' });
    if (results.length === 0) return res.status(400).json({ message: 'Користувач не знайден' });
    const user = results[0];
    if (!bcrypt.compareSync(password, user.password)) return res.status(400).json({ message: 'Невірный пароль'
});
    const token = jwt.sign({ id: user.id, email }, SECRET, { expiresIn: '7d' });
    res.json({ token, email });
  });
};
```

```
exports.authMiddleware = (req, res, next) => {
  const auth = req.headers.authorization;
  if (!auth) return res.status(401).json({ message: 'Нема токена' });
  const token = auth.split(' ')[1];
  try {
    req.user = jwt.verify(token, SECRET);
    next();
  } catch {
    res.status(401).json({ message: 'Невірный токен' });
  }
};
```

Код generalSuggestController.js

```
const { OpenAI } = require('openai');
const openai = new OpenAI({ apiKey: process.env.OPENAI_API_KEY });

exports.generalSuggestions = async (req, res) => {
  const { query, images } = req.body;
  if (!query || !images || !Array.isArray(images) || images.length === 0) {
    return res.status(400).json({ message: 'Нема запросу чи картинок' });
  }

  // Промт для підказок
  const prompt = `
Користувач шукав: "${query}".
Ось приклади знайдених картинок (опис):
${images.slice(0, 10).map((img, i) => `${i + 1}. ${img.description || "}`).join('\n')}
```

Запропонуй 5 порад, як користувач може зробити свій пошуковий запит більш точним і отримати кращі результати. Врахуй початковий запит та описи знайдених картинок.Дай відповіді тією ж мовою, якою поставлено запит.Відповідай списком, без пояснень.

```
  `.trim();

  try {
    const completion = await openai.chat.completions.create({
      model: "gpt-4-turbo",
      messages: [{ role: "user", content: prompt }],
      max_tokens: 100,
      temperature: 0.7
    });

    const text = completion.choices[0].message.content;
    const suggestions = text
```

```
    .split('\n')
    .map(s => s.replace(/^\d+[\).\s-]*/, '').trim())
    .filter(Boolean);

  res.json({ suggestions });
 } catch (e) {
  res.status(500).json({ message: 'Помилка генерації підказок', error: e.message });
 }
};
```

Код searchController.js

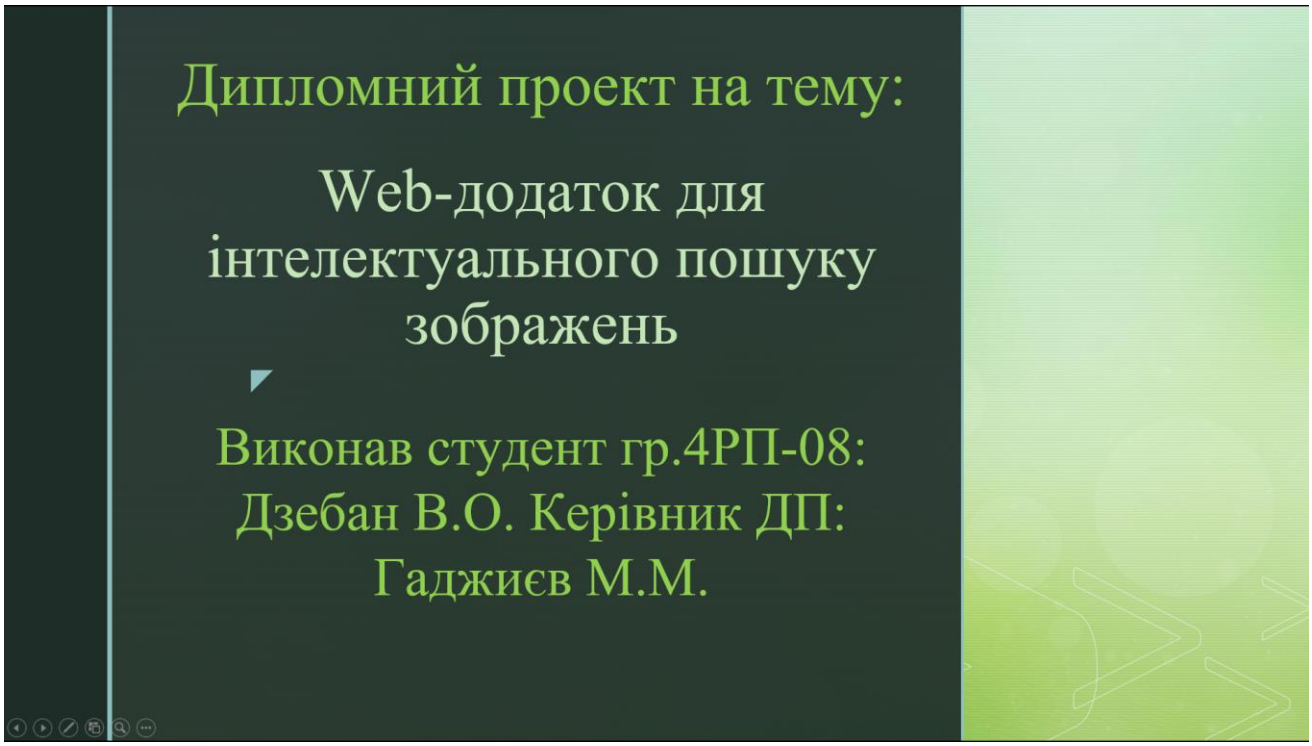
```
const axios = require('axios');

exports.searchImagesGoogle = async (req, res) => {
 const { query, total = 30 } = req.body;
 if (!query) return res.status(400).json({ message: 'Нема запиту' });
 try {
  let images = [];
  for (let start = 1; images.length < total; start += 10) {
   const resGoogle = await axios.get('https://www.googleapis.com/customsearch/v1', {
    params: {
     key: process.env.GOOGLE_API_KEY,
     cx: process.env.GOOGLE_CX,
     q: query,
     searchType: 'image',
     num: Math.min(10, total - images.length),
     start
    }
   });
   if (!resGoogle.data.items || resGoogle.data.items.length === 0) break;
   images.push(...resGoogle.data.items.map(item => ({
    url: item.link,
    description: item.title
   })));
  }
  //фільтрація по доменам
  const allowedDomains = [
   'unsplash.com',
   'images.unsplash.com',
   'pinimg.com',
   'pinterest.com'
```

```
    ];

    images = images.filter(img => {
      try {
        const url = new URL(img.url);
        const hostname = url.hostname;
        return (
          typeof img.url === 'string' &&
          img.url.startsWith('http') &&
          /\.(jpg|jpeg|png)$/i.test(img.url) &&
          allowedDomains.some(domain => hostname.includes(domain))
        );
      } catch {
        return false;
      }
    });

    res.json(images);
  } catch (e) {
    res.status(500).json({ message: 'Помилка пошуку', error: e.message });
  }
};
```

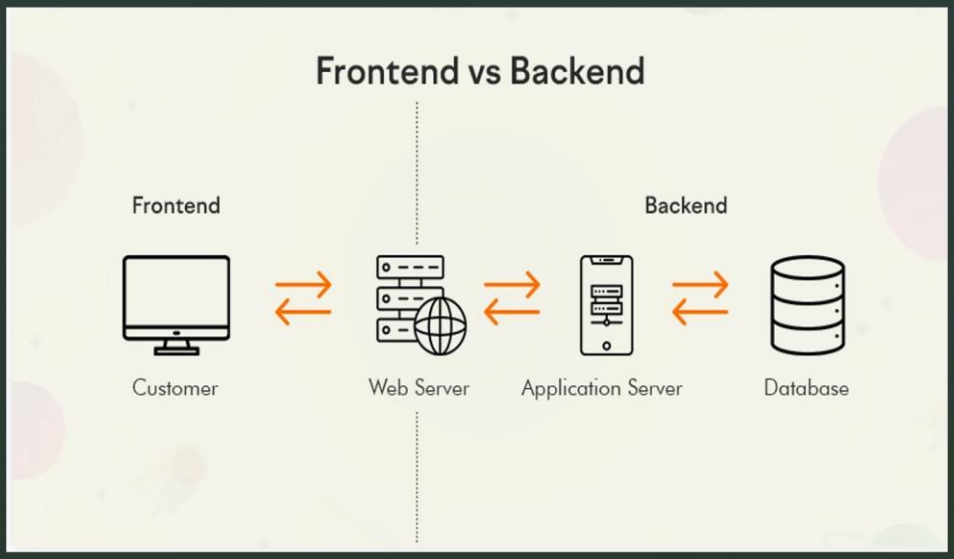
Код server.js

```
const express = require('express');
const cors = require('cors');
const authRoutes = require('./routes/auth');
const searchRoutes = require('./routes/search');
const generalSuggestRoutes = require('./routes/generalSuggest');
const app = express();

app.use(cors());
app.use(express.json());

app.use('/api/auth', authRoutes);
app.use('/api/search', searchRoutes);
app.use('/api/suggest', generalSuggestRoutes);

const PORT = process.env.PORT || 3001;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

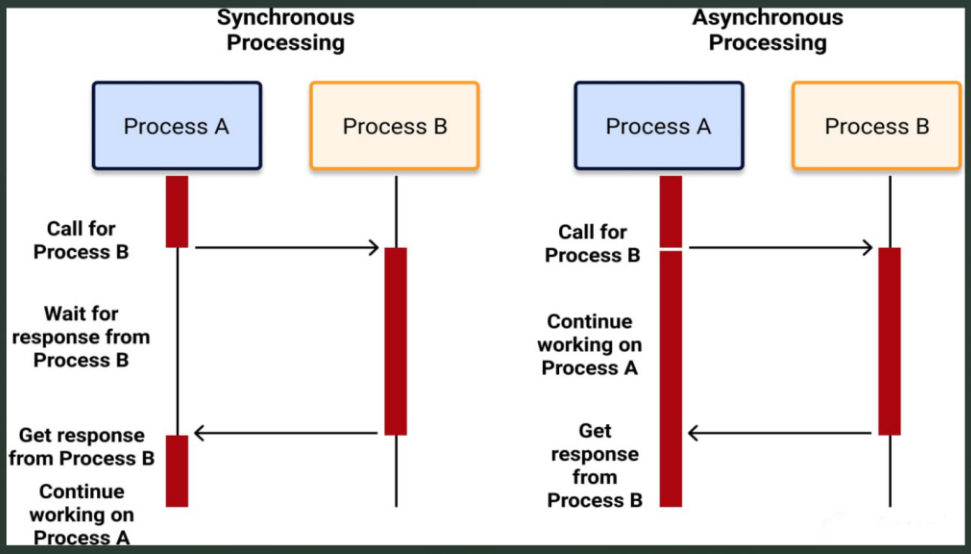
ДОДАТОК Б. Слайди мультимедійної презентації



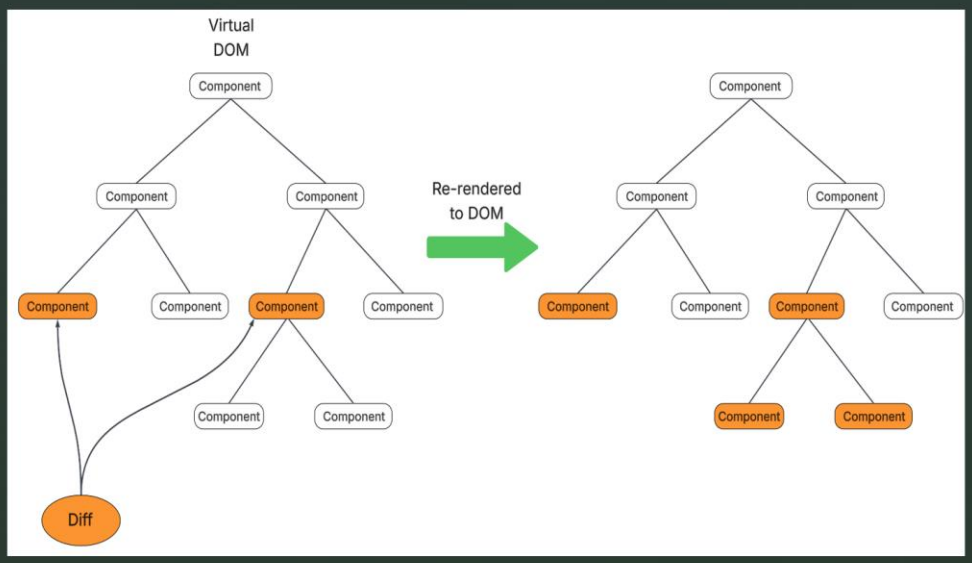
Взаємозв’язок між front-end та back-end частинами



Порівняння асинхронноїмоделі вводу-виводу з синхронною



Алгоритм роботи Virtual DOM від React



Реалізація серверу з Express.js

server.js

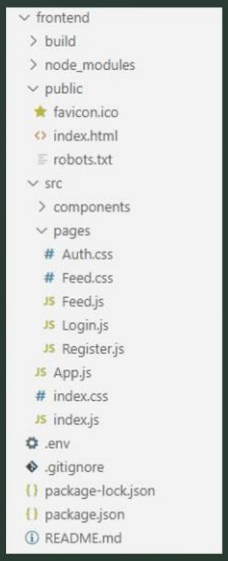
backend > server.js > ...

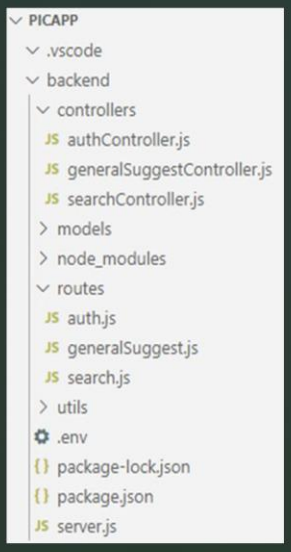
```
1   const express = require('express');
2   const cors = require('cors');
3   const authRoutes = require('./routes/auth');
4   const searchRoutes = require('./routes/search');
5   const generalSuggestRoutes = require('./routes/generalSuggest');
6   const app = express();
7
8   app.use(cors());
9   app.use(express.json());
10
11  app.use('/api/auth', authRoutes);
12  app.use('/api/search', searchRoutes);
13  app.use('/api/suggest', generalSuggestRoutes);
14
15  const PORT = process.env.PORT || 3001;
16  app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

Реалізація ендпоінту для генерації підказок

```
const { OpenAI } = require('openai');

const openai = new OpenAI({ apiKey: process.env.OPENAI_API_KEY });

exports.generalSuggestions = async (req, res) => {
  const { query, images } = req.body;
  if (!query || !images || !Array.isArray(images) || images.length === 0) {
    return res.status(400).json({ message: 'Нема запросу чи картинок' });
  }

  // Промт для підказок
  const prompt = `
Користувач шукав: "${query}".
Ось приклади знайдених картинок (опис):
${images.slice(0, 10).map((img, i) => `${i + 1}. ${img.description || ''}`).join('\n')}

Запропонуй 5 порад, як користувач може зробити свій пошуковий запит більш точним і отримати кращі результати.
Врахуй початковий запит та описи знайдених картинок.Також дай відповідь на тій мові на якій зробили запит.Відповідай списком, без пояснень.
  `.trim();

  try {
    const completion = await openai.chat.completions.create({
      model: "gpt-4-turbo",
      messages: [{ role: "user", content: prompt }],
      max_tokens: 100,
      temperature: 0.7
    });

    const text = completion.choices[0].message.content;
    const suggestions = text
      .split('\n')
      .map(s => s.replace(/^\d+[\).\s-]*/, '').trim())
      .filter(Boolean);

    res.json({ suggestions });
  } catch (e) {
    res.status(500).json({ message: 'Помилка генерації підказок', error: e.message });
  }
};
```

Структура front-end та back-end частин





Реалізація логіки авторизації

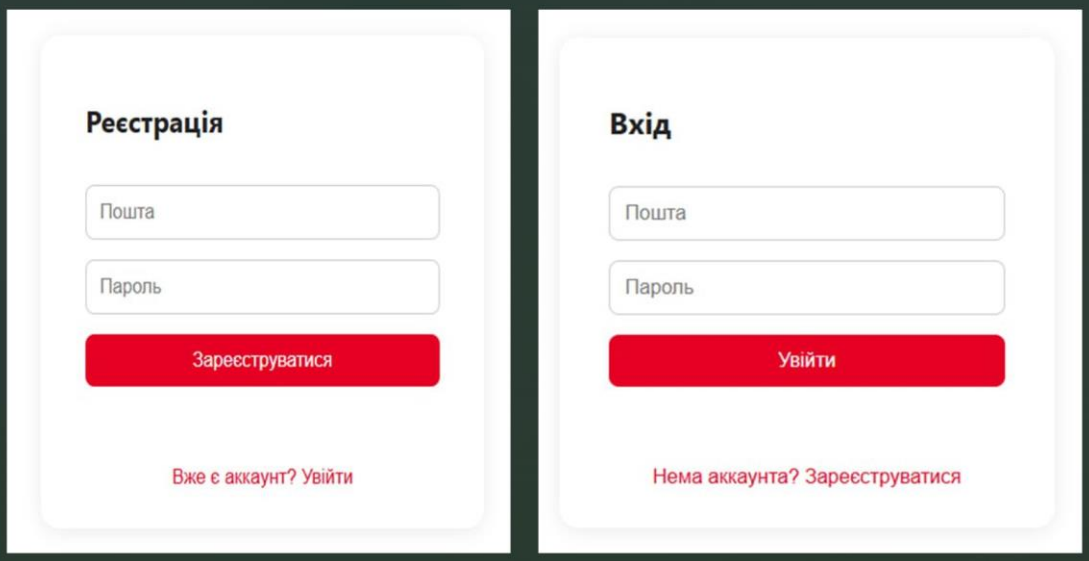
```js
const db = require('../models/db');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');

const SECRET = process.env.JWT_SECRET || 'dev_secret';

exports.register = (req, res) => {
  const { email, password } = req.body;
  if (!email.endsWith('@gmail.com')) {
    return res.status(400).json({ message: 'Реєстрація дозволена лише для Gmail' });
  }
  db.query('SELECT * FROM users WHERE email = ?', [email], (err, results) => {
    if (err) return res.status(500).json({ message: 'Помилка серверу' });
    if (results.length > 0) return res.status(400).json({ message: 'Користувач вже існує' });
    const hash = bcrypt.hashSync(password, 8);
    db.query('INSERT INTO users (email, password) VALUES (?, ?)', [email, hash], (err, result) => {
      if (err) return res.status(500).json({ message: 'Помилка реєстрації' });
      const token = jwt.sign({ id: result.insertId, email }, SECRET, { expiresIn: '7d' });
      res.json({ token, email });
    });
  });
};

exports.login = (req, res) => {
  const { email, password } = req.body;
  db.query('SELECT * FROM users WHERE email = ?', [email], (err, results) => {
    if (err) return res.status(500).json({ message: 'Помилка серверу' });
    if (results.length === 0) return res.status(400).json({ message: 'Користувач не знайден' });
    const user = results[0];
    if (!bcrypt.compareSync(password, user.password)) return res.status(400).json({ message: 'Невірний пароль' });
    const token = jwt.sign({ id: user.id, email }, SECRET, { expiresIn: '7d' });
    res.json({ token, email });
  });
};

exports.authMiddleware = (req, res, next) => {
  const auth = req.headers.authorization;
  if (!auth) return res.status(401).json({ message: 'Нема токена' });
  const token = auth.split(' ')[1];
  try {
    req.user = jwt.verify(token, SECRET);
    next();
  } catch {
    res.status(401).json({ message: 'Невірний токен' });
  }
};
```

Реалізація перевірки токену користувача

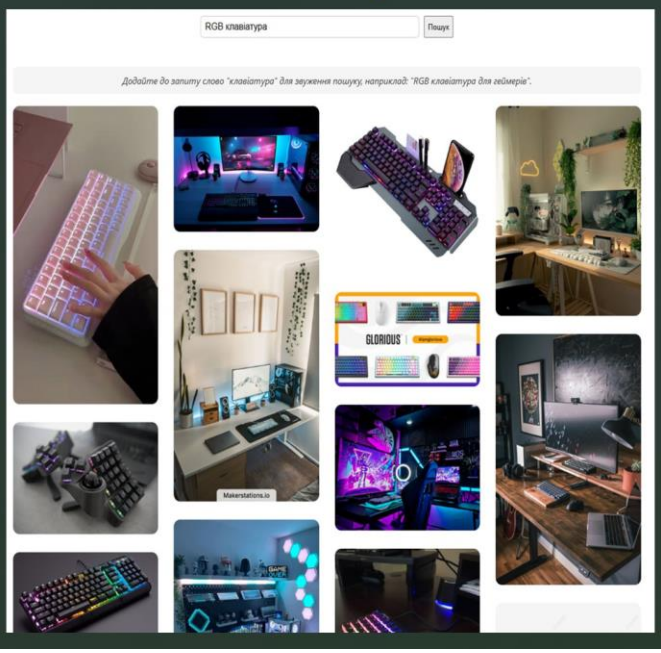
```jsx
import React, { useEffect, useState } from "react";
import Feed from "./pages/Feed";
import Login from "./pages/Login";
import Register from "./pages/Register";
import axios from "axios";

function App() {
  const [user, setUser] = useState(() => localStorage.getItem("token"));
  const [isLogin, setIsLogin] = useState(true);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    const checkToken = async () => {
      if (!user) {
        setLoading(false);
        return;
      }
      try {
        await axios.get("http://localhost:3001/api/auth/check", {
          headers: { Authorization: `Bearer ${user}` }
        });
        setLoading(false);
      } catch {
        localStorage.removeItem("token");
        setUser(null);
        setLoading(false);
      }
    };
    checkToken();
  }, [user]);

  if (loading) return <div>Завантаження...</div>;

  if (!user) {
    return isLogin
      ? <Login setUser={setUser} setIsLogin={setIsLogin} />
      : <Register setUser={setUser} setIsLogin={setIsLogin} />;
  }

  return <Feed setUser={setUser} />;
}

export default App;
```

Блок схема алгоритму користування web-застосунком

Скриншот сторінки реєстрації та входу



Скриншот головної сторінки web-застосунку



ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Дзебана Владислава Олександровича

(прізвище, ім`я та по батькові)

Спеціальність *121 «Інженерія програмного забезпечення»*

Освітня програма *«розробка програмного забезпечення»*

Керівник дипломного проекту (роботи) *Гаджиєв Матін Магсудович*

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) *Розробка web-застосунку для інтелектуального пошуку зображень*

Обсяг розрахунково-пояснювальної записки *72* сторінок

Обсяг графічної (презентаційної) частини *13* аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту завданню

Представлений на рецензію дипломний проект відповідає затвердженій темі та виконаний відповідно технічному завданню. Дипломний проект присвячений проблемі інтелектуального пошуку зображень та складається з пояснювальної записки, додатку з програмним кодом та мультимедійної презентації, що містить приклади роботи програми.

б) характеристика виконання кожного розділу дипломного проекту

Пояснювальна записка складається з основного розділу, економічного розділу, розділу охорони праці та додатків. Перелічені розділи поетапно охоплюють розробку, виконані достатньо обґрунтовано. Розділ охорони праці містить загальну інформацію та вимоги до техніки безпеки оператора КТ. Економічний розділ проекту містить розрахунок витрат на НДР та реалізацію проекту.

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту

Графічна частина складається з 13 слайдів мультимедійної презентації, виконаної у програмному продукті MS PowerPoint, які містять ілюстративні схеми, скріншоти роботи програмного застосунку, передбачені технічним завданням. Пояснювальна записка виконана у відповідності до норм. Якість виконання графічної частини проекту та пояснювальної записки добра, розробку виконано у повному обсязі.

г) перелік позитивних якостей дипломного проекту *Сучасна технологічна база та клієнт-серверна архітектура. Інтеграція штучного інтелекту для генерації підказок. Аналіз ринку та обґрунтування вибору технологій. Наявність детальних фрагментів коду та структурованої документації.*

д) основні недоліки дипломного проекту *Залежність від зовнішніх API. Не розглянуто, як застосунок буде масштабуватись для обробки великих обсягів пошукових запитів. Недостатньо описано заходи для захисту від потенційних вразливостей (наприклад, витоків токенів або SQL-ін'єкцій). Відсутність детальної інформації про проведення unit-тестування чи інтеграційних тестів. Наявні деякі помилки оформлення ПЗ*

Оцінка розрахункової частини *Добре*

Оцінка графічної частини *Добре*

Загальна оцінка *Добре*

Прізвище, ім'я, по батькові рецензента *к.т.н. Рудніченко Микола Дмитрович*

Місце роботи і посада рецензента *Національний університет «Одеська політехніка», доцент кафедри інформаційних технологій*

Підпис

« *20* » *червня* 2025 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проєкт здобувача (здобувачки) освіти

відділення комп'ютерних систем

Дзебана Владислава Олександровича

(прізвище, ім`я та по батькові)

Спеціальність: *121 "Інженерія програмного забезпечення"*

Освітньо-професійна програма: *«Розробка програмного забезпечення»*

Тема дипломного проєкту: *Розробка web-застосунку для інтелектуального пошуку зображень*

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЄКТУ

а) обсяг і якість виконання проєкту (графічного матеріалу і розрахунково-пояснювальної записки) *Дипломний проєкт виконано відповідно технічному завданню. Пояснювальна записка містить 72 сторінки. У пояснювальній записці виконано опис етапів розробки web-застосунку для інтелектуального пошуку зображень, а також його програмного забезпечення. Графічна частина складається з 13 слайдів мультимедійної презентації, які також містять креслення, передбачені технічним завданням. Якість виконання пояснювальної записки та графічної частини добра, розробку виконано в повному обсязі.*

б) самостійність роботи над проєктом: *Протягом всього строку дипломного проєктування та переддипломної практики здобувач освіти Дзебан Владислав Олександрович поступово та послідовно виконував всі етапи розробки. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника*

в) теоретична підготовка випускника (випускниці): *Здобувач освіти Дзебан Владислав Олександрович під час роботи над дипломним проєктом вивчив достатню кількість літературних джерел та матеріалів за даною тематикою.*

Вважаю, що теоретична підготовка дипломника добра і він готовий до

г) вміння розв'язувати виробничі та конструкторські питання _____

Під час дипломного проєктування здобувач освіти Дзебан Владислав Олександрович продемонстрував навички ефективно розв'язувати виробничі питання, що виникали під час створення web-застосунку для інтелектуального пошуку зображень

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Відмінно*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові керівника дипломного проєкту _____

Гаджиєв Матін Магсудович

Місце роботи і посада керівника дипломного проєкту _____

ВСП "Одеський технічний фаховий коледж ОНТУ", викладач спецдисциплін комісії комп'ютерних технологій та програмної інженерії

Підпис _____

«18» 06 2025 р.

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
(ДИПЛОМНОГО ПРОЕКТУ)
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Дзебан В.О.
здобувач освіти гр. 4РП-08, та

Гаджиєв М.М.,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до дипломного проекту фахового молодшого бакалавра на тему:

«Розробка web-застосунку для інтелектуального пошуку зображень» (автор роботи – Дзебан В.О., керівник роботи – Гаджиєв М.М.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2025 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи і даємо згоду на обробку персональних даних.

Виконавець _____ / Дзебан В.О. /

Керівник _____ / Гаджиєв М.М. /

«16» червня 2025 р.

Д О В І Д К А

циклової комісії КТ та ПІ
про допуск до захисту дипломного проєкту
здобувача (здобувачки) освіти IV курсу
відділення комп'ютерних систем групи 4РП-08

Дзебана Владислава Олександровича

на тему *Розробка web-застосунку для інтелектуального пошуку зображень*

Висновок відповідальної особи за проведення нормоконтролю:
пояснювальна записка до дипломного проєкту виконана з некритичними порушеннями ДСТУ та оформлена відповідно до вимог Положення про дипломне проєктування

(підпис) *18.06.2025* (дата) *Петрашова В.І.* (П.І.Б.)

Висновок відповідальної особи за перевірку роботи на наявність академічного плагіату *згідно звіту про перевірку від 18.06.2025 р. значення коефіцієнту подібності в роботі становить 19,61%, коефіцієнт цитування – 1,74%.*

(підпис) *18.06.2025* (дата) *Краснокутська К.Г.* (П.І.Б.)

Попередня експертиза (малий захист) дипломного проєкту

здобувача (здобувачки) освіти *Дзебана В.О.* (П.І.Б.)

проведена « *18* » *червня* 20*25* р.

Висновки *Пояснювальна записка до дипломного проєкту виконана у повному обсязі. Випускна кваліфікаційна робота (дипломний проєкт) відповідає вимогам Положення про дипломне проєктування та рекомендована до захисту.*

Голова ЦК КТ та ПІ (підпис) *Кривченко Ю.В.* (П.І.Б.)

StrikePlagiarism.com

Дата звіту 6/18/2025
Дата редагування 6/18/2025

Документ прийнятий

Звіт подібності

метадані

Назва організації
Odesa Technical Professional College of Odesa National University of Technology
Заголовок
Розробка web-застосунку для інтелектуального пошуку зображень

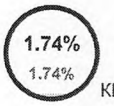
Автор
Дзебан Владислав Олександрович
Науковий керівник / Експерт
Гаджиєв Матин Магсудович

підрозділ
Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету"

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

КП 1



КЦ

25	12862	102178
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	R	2
Інтервали	A⟶	0
Мікропробіли	·	0
Білі знаки	R	0
Парафрази (SmartMarks)	a	76

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download	57 0.44 %
2	https://card-file.ontu.edu.ua/bitstreams/c58b0ff5-46e0-49f8-8cbe-65c32256665d/download	53 0.41 %
3	https://ztu.edu.ua/site/graduation-works-get-file?id=7433761	53 0.41 %
4	https://card-file.ontu.edu.ua/bitstreams/55e2b8f2-7d3c-4235-99fc-2be51199b96d/download	51 0.40 %
5	https://card-file.ontu.edu.ua/bitstreams/34a6756b-592f-4b77-a805-183aa03a6a26/download	45 0.35 %

6	https://card-file.ontu.edu.ua/server/api/core/bitstreams/8da72e29-656f-4ee4-9b22-716dedf53ff5/content	43 0.33 %
7	https://card-file.ontu.edu.ua/server/api/core/bitstreams/8da72e29-656f-4ee4-9b22-716dedf53ff5/content	41 0.32 %
8	https://tr.nmc.dsns.gov.ua/files/%D1%81%D0%B0%D0%B9%D1%82/PDF%20%D1%84%D0%B0%D0%B9%D0%BB%D0%B8/I%D0%94%D0%B8%D1%81%D1%82%D0%9D%D0%B0%D0%B2%D1%87/%D0%9F%D0%91/tema%208.pdf	39 0.30 %
9	https://card-file.ontu.edu.ua/bitstreams/e4afae26-0a7e-4a4d-afc2-94341838de2a/download	38 0.30 %
10	Створення web-застосунку цифрового помічника з використанням Open AI [redacted] 5/28/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	38 0.30 %

з домашньої бази даних (2.58 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Розробка web-застосунку для генерації повідомлень із використанням технологій штучного інтелекту [redacted] 6/14/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	136 (11) 1.06 %
2	Створення web-застосунку цифрового помічника з використанням Open AI [redacted] 5/28/2025 **Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")**	**107 (5) 0.83 %**
3	Розробка програмного застосунку сканування мережевих портів з метою виявлення вразливостей [redacted] 6/17/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	39 (2) 0.30 %
4	Розробка програмного забезпечення для відстеження ступеня готовності виконання проєкту [redacted] 6/17/2025 **Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")**	**38 (1) 0.30 %**
5	Розробка web-застосунку для аутентифікації користувачів з використанням методів криптографії [redacted] 6/17/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	6 (1) 0.05 %
6	Розробка 3D-гри у жанрі survival-horror з налаштуваннями рівнів складності [redacted] 6/12/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	6 (1) 0.05 %

з програми обміну базами даних (0.36 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

1	Система прогнозування розвитку серцево судинних захворювань на основі методів машинного навчання [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	12 (1) 0.09 %
2	Інтернет-магазин «Handmade» [redacted] 5/26/2024 State University of Infrastructure and technology (State University of Infrastructure and technology)	11 (1) 0.09 %
3	Розроблення модуля розумного помічника для створення резюме з використанням мови TypeScript [redacted] 5/29/2025 Kharkiv National University of Economics named after S.Kuznets (KNUE) (KNUE)	10 (1) 0.08 %
4	Чумаченко А.О..docx [redacted] 12/16/2024 East Ukrainian National University named after Volodymyr Dahl (East Ukrainian National University named after Volodymyr Dahl)	7 (1) 0.05 %
5	Магістерська робота [redacted] 2/14/2025 King Danylo University (King Danylo University)	6 (1) 0.05 %

з Інтернету (16.67 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://card-file.ontu.edu.ua/server/api/core/bitstreams/44c16132-5f53-48e2-b6c0-61e9a2f0fd75/content	511 (42) 3.97 %
2	**https://card-file.ontu.edu.ua/bitstreams/e4afae26-0a7e-4a4d-afc2-94341838de2a/download**	**223 (10) 1.73 %**
3	**https://card-file.ontu.edu.ua/bitstreams/538ada8a-2c79-4b1e-b7d2-b0c97f68bc1c/download**	**210 (12) 1.63 %**
4	**https://card-file.ontu.edu.ua/server/api/core/bitstreams/8da72e29-656f-4ee4-9b22-716dedf53ff5/content**	**115 (4) 0.89 %**
5	**https://card-file.ontu.edu.ua/bitstreams/55e2b8f2-7d3c-4235-99fc-2be51199b96d/download**	**105 (4) 0.82 %**
6	**https://card-file.ontu.edu.ua/bitstreams/34a6756b-592f-4b77-a805-183aa03a6a26/download**	**90 (4) 0.70 %**
7	**https://ztu.edu.ua/site/graduation-works-get-file?id=7433761**	**88 (4) 0.68 %**
8	https://card-file.ontu.edu.ua/bitstreams/62baa43e-b968-4993-bb54-8cf8761a89b2/download	85 (8) 0.66 %
9	**https://card-file.ontu.edu.ua/bitstreams/c58b0ff5-46e0-49f8-8cbe-65c32256665d/download**	**71 (3) 0.55 %**
10	**https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download**	**67 (3) 0.52 %**
11	**https://tr.nmc.dsns.gov.ua/files/%D1%81%D0%B0%D0%B9%D1%82/PDF%20%D1%84%D0%B0%D0%B9%D0%BB%D0%B8/I%D0%94%D0%B8%D1%81%D1%82%D0%9D%D0%B0%D0%B2%D1%87/%D0%9F%D0%91/tema%208.pdf**	**60 (3) 0.47 %**
12	https://stackoverflow.com/questions/78052860/token-has-expired-in-node-js	60 (6) 0.47 %
13	https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download	59 (6) 0.46 %
14	**https://card-file.ontu.edu.ua/bitstreams/29489599-0581-4ce6-8890-c3b13d9f2e0e/download**	**58 (5) 0.45 %**
15	**https://card-file.ontu.edu.ua/bitstreams/9908b7a9-6b3e-46f5-a46e-84d83787cfd4/download**	**57 (4) 0.44 %**
16	https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content	39 (2) 0.30 %
17	https://gi.edu.ua/koledzh/pidrozdily/biblioteka/repozytorii/%D1%84%D0%B0%D0%B9%D0%BB/%D0%B7%D0%B0%D0%B2%D0%B0%D0%BD%D1%82%D0%B0%D0%B6%D0%B8%D1%82%D0%B8/51c51db83db4752d1e928dba86d35f15	36 (4) 0.28 %

18	https://essuir.sumdu.edu.ua/bitstream/123456789/91029/1/Pokutniy_mag_rob.pdf	36 (4) 0.28 %
19	https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download	27 (3) 0.21 %
20	https://dev.to/kosa12/creating-and-securing-jwt-authentication-in-a-web-app-5cni	26 (4) 0.20 %
21	https://card-file.ontu.edu.ua/bitstreams/6cf43324-8f08-4031-ba42-f80b18efbbc8/download	14 (1) 0.11 %
22	https://openarchive.nure.ua/server/api/core/bitstreams/ad58c324-0967-4cf4-ab94-f6ab94087c97/content	12 (1) 0.09 %
23	https://card-file.ontu.edu.ua/server/api/core/bitstreams/6eb6bf1c-5813-45e6-93c5-25539b4709d3/content	12 (1) 0.09 %
24	https://bibliotekanauki.pl/articles/2063968.pdf	12 (1) 0.09 %
25	http://dspace.puet.edu.ua/bitstream/123456789/13481/1/Lazarenko_Vladyslav_Oleksandrovych_KNm21-1.pdf	11 (1) 0.09 %
26	https://ela.kpi.ua/bitstreams/c03b51bf-a39f-4c2f-a957-710ccc210c6f/download	10 (1) 0.08 %
27	https://card-file.ontu.edu.ua/bitstreams/12d5c0ab-e979-48f2-a8ec-d5fc31f71fd5/download	9 (1) 0.07 %
28	https://card-file.ontu.edu.ua/server/api/core/bitstreams/c63b91ba-d04f-4715-890d-b16277695c7e/content	9 (1) 0.07 %
29	http://rep.knlu.edu.ua/xmlui/bitstream/handle/787878787/1969/%D0%90%D0%BD%D1%82%D0%BE%D0%BD%D0%BE%D0%B2%D0%B0.pdf?sequence=1	8 (1) 0.06 %
30	https://card-file.ontu.edu.ua/bitstreams/bbaf3f38-16a8-4070-bead-5562769b7c71/download	7 (1) 0.05 %
31	https://card-file.ontu.edu.ua/bitstreams/11562741-24e6-4201-bc41-a00c8013fca1/download	6 (1) 0.05 %
32	https://card-file.ontu.edu.ua/bitstreams/0e72a3b9-bdd7-4711-a3c6-dedc1d4287cc/download	6 (1) 0.05 %
33	https://ela.kpi.ua/bitstream/123456789/49584/1/Vasyliev_bakalavr.pdf	5 (1) 0.04 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма: «Розробка програмного забезпечення»
Група: 4РП- 08

Дипломний проєкт
здобувача освіти денної форми навчання
РП. 08.08.000.ДП

ДЗЕБАНА
ВЛАДИСЛАВА ОЛЕКСАНДРОВИЧА

м. Одеса
2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма: «Розробка програмного забезпечення» Група: 4РП- 08 **ПОЯСНЮВАЛЬНА ЗАПИСКА до дипломного проєкту на тему:** _____
_____ **Проєктний матеріал складається**