Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА ВЕБЗАСТОСУНКУ З РОЗПІЗНАВАННЯ ТЕКСТУ ІЗ ЗОБРАЖЕНЬ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-19-2
Мартиненко М.В.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Шафроненко А.Ю.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Мартиненку Максиму Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка вебзастосунку з розпізнавання тексту із зображень

затверджена наказом університету від 15 травня 2023 року № 474 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 27 травня 2023 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, бібліотека комп'ютерного зору з відкритим кодом Tesseract OSR, середовище розробки Visual Studio.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд предметної області та постанова задачі.
2. Моделювання вебзастосунку.
3. Розробка вебзастосунку та тестування.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Аналіз предметної області, постановка задачі, моделювання вебзастосунку, аналіз роботи розробленого вебзастосунку.

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

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант з дотримання діючих стандартів та норм	Доцент Творошенко І.С.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	10.04.2023	
2	Аналіз завдання, підбір літератури	11.04.23-17.04.23	
3	Аналіз літератури з досліджуваної проблеми	18.04.23-20.04.23	
4	Аналіз технічних засобів	21.04.23-30.04.23	
5	Розробка вебзастосунку	01.05.23-14.05.23	
6	Програмна реалізація	15.05.23-23.05.23	
7	Оформлення пояснювальної записки	24.05.23-26.05.23	
8	Перевірка на плагіат	27.05.23	
9	Рецензування	28.05.23	
10	Підготовка презентації та доповіді	29.05.23-30.05.23	
11	Занесення роботи в електронний архів	31.05.23	
12	Попередній захист кваліфікаційної роботи	05.06.23	

Дата видачі завдання 10 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ _____доц. Шафроненко А.Ю._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 55 с., 27 рис., 31 джерело.

РОЗПІЗНАВАННЯ ТЕКСТУ, ВЕБЗАСТОСУНОК, JAVASCRIPT, TESSERACT, REACT, BOOTSTRAP.

Об’єктом роботи є дослідження тексту із зображень.

Метою роботи є розробка вебзастосунку з розпізнавання тексту із зображень.

Використано бібліотеку стилів Bootstrap, фреймворк React.js та бібліотеку для обробки зображень Tesseract.js. Досліджено та протестовано роботу вебзастосунку.

У результаті виконання кваліфікаційної роботи розроблено вебзастосунок із розпізнавання тексту з зображень.

TEXT RECOGNITION, WEB APPLICATION, JAVASCRIPT, TESSERACT, REACT, BOOTSTRAP.

The object of the work is the study of text from images.

The aim of the work is to develop a web application for recognising text from images.

The Bootstrap style library, the React.js framework, and the Tesseract.js image processing library were used. The web application has been investigated and tested.

As a result of the qualification work, a web application for recognising text from images has been developed.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 6
Вступ 7
1 Огляд предметної області та постанова задачі 8
1.1 Що таке інтернет та для чого він був створений 8
1.2 Робота інтернету 9
1.3 Структура сучасного вебсайту 11
1.4 Технології розпізнавання тексту із зображень 14
1.5 Постановка задачі 16
2 Моделювання вебзастосунку 18
2.1 Архітектура вебзастосунків 18
2.1.1 Моноліт 18
2.1.2 Мікросервіси 20
2.1.3 Серверлес 22
2.2 Шаблони проєктування 24
2.2.1 Шаблон проєктування MVC 24
2.2.2 Шаблон проєктування MVP 26
2.2.3 Шаблон проєктування MVVM 27
2.3 Технології обробки зображень в вебзастосунках 28
2.4 Розробка моделі вебзастосунку 29
3 Розробка вебзастосунку та тестування 32
3.1 Обґрунтування вибору середовища програмної реалізації 32
3.2 Програмна реалізація 35
3.3 Інструкція користувача 39
3.4 Тестування розробленої моделі 44
3.4.1 Тестування зображень з різною якістю зображення 44
3.4.2 Тестування зображень з маленьким текстом 46
3.4.3 Тестування зображення з текстом під нахилом 47
3.5 Перспективи подальшої роботи 48
Висновки 51
Перелік джерел посилання 52

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

DNS – Domain Name System (система доменних імен)

HTTP – Hypertext Transfer Protocol (протокол передачі гіпертексту)

SLL – Secure Sockets Layer (рівень захищених сокетів)

TLS – Transport Layer Security (безпека транспортного рівня)

OCR – Optical Character Recognition (оптичне розпізнавання символів)

CSS – Cascading Style Sheets (каскадні таблиці стилів)

ВСТУП

Сьогодні людство швидко розвивається, і цей розвиток відбувається щодня. Недавно комп’ютери займали цілі кімнати для простих математичних обчислень, але зараз їх можна легко носити з собою і створювати різноманітні картинки, відеоролики та програми.

Із стрімким розвитком технологій люди все більше покладалися та довіряли роботу комп’ютеру. Все більше інформації створювалось і зберігалось в електронному вигляді. Для легкої передачі та доступу був створений інтернет. Він став місцем, де можна знайти будь-яку інформацію з будь-якої галузі. Сьогодні все більше документообіг переходить в мережу, цю можливість впроваджують від малих компаній до державних установ. Не рідко документи зберігаються у вигляді фотографій і отримання тесту з цих файлів є нагальною проблемою.

Актуальність роботи полягає у тому, що з розвитком технологій та збільшенням обсягу електронної інформації зростає потреба в швидкому та ефективному збереженні, передачі та обробці цієї інформації. Одним із важливих аспектів є отримання тексту з фотографій та інших неструктурованих даних, які містяться в електронних документах. Це може бути корисним для бізнесу, державних установ, наукових досліджень та інших галузей.

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВА ЗАДАЧІ

1.1 Що таке інтернет та для чого він був створений

Наприкінці 1960-х років перший Інтернет, відомий як ARPANET, був створений на основі програмної мережі пакетної комутації Агентства перспективних дослідницьких проєктів Міноборони США. 1 вересня 1969 року був встановлений перший сервер у Каліфорнійському університеті в Лос-Анджелесі. 22 жовтня 1969 року вважається днем народження інтернет з'єднання, адже саме цього дня було встановлене з'єднання та передача даних між Каліфорнійським університетом, який очолював Чарлі Клайн, та Стенфордом, керівником якого був Білл Давалл [1].

У 1984 році Національним науковим фондом США була створена ще одна велика мережа, NSFNet. Ця мережа включала багато менших мереж, в тому числі популярні на той час Bitnet і Usenet, і мала набагато більшу пропускну здатність, ніж мережа ARPANet. Це створило серйозну конкуренцію ARPANet: менш ніж за рік до NSFNet було підключено понад 10 000 комп'ютерів, і термін «Інтернет» поступово став асоціюватися з цією мережею.

У березні 1989 року Тім Бернерс-Лі та Роберт Кайо представили керівництву ЦЕРН проєкт створення мережі гіпертекстових документів під назвою Всесвітня павутина. Це була колекція гіпертекстових документів, яка пов'язувала документи в інформаційну систему, доступ до якої можна було отримати з будь-якого вузла мережі для полегшення роботи в ЦЕРН У грудні 1990 року Тім Бернерс-Лі створив основи мови HTML, і був створений перший вебсайт. Лише в 1995 році Всесвітня павутина стала головним гравцем у сфері даних, що призвело до створення Консорціуму Всесвітньої павутини (W3C). Після 1996 року терміни «Інтернет» і «Всесвітня павутина» стали майже повністю взаємозамінними [2].

Підсумовуючи, Інтернет – це глобальна комунікаційна мережа, що об’єднує мільйони комп’ютерів та інших пристроїв по всьому світу. Ця мережа дає змогу людям обмінюватися інформацією, спілкуватися один з одним і виконувати різні види діяльності за допомогою вебсайтів і застосунків. Інтернет був створений з метою забезпечення швидкого та зручного доступу до інформації та комунікації між комп’ютерами [3].

1.2 Робота інтернету

Сьогодні життя немислиме без інтернету. Він став невід’ємною частиною нашого життя і має значний вплив на багато аспектів повсякденного життя. Ось кілька способів показати важливість інтернету в житті:

– спілкування. Інтернет дозволяє нам спілкуватися з родиною, друзями та колегами по всьому світу. Такі інструменти, як електронна пошта, соціальні мережі та миттєві повідомлення, дозволяють нам спілкуватися в режимі реального часу;

– розваги та розвиток. Інтернет містить багато ресурсів для розваг та освіти. Ви можете дивитися фільми, слухати музику, грати в ігри і навіть вчитися новому за допомогою онлайн-курсів і вебінарів;

– бізнес і комерція. Інтернет дозволяє людям працювати з будь-якої точки світу, що дає можливість бізнесу швидко розширюватися і досліджувати нові ринки. Багато компаній також використовують Інтернет для організації онлайн-конференцій та відеоконференцій;

– пошук інформації. Інтернет – це найбільша база знань, доступна людству. У ньому можна швидко знайти відповіді на всі питання, які виникають у нашому житті;

– покупки та банківські операції. Інтернет дозволяє нам швидко і зручно робити покупки онлайн і отримувати доступ до банківських послуг за допомогою онлайн-банкінгу.

Для обміну інформацією та забезпечення доступу до всіх цих ресурсів Інтернет використовує мережу комп’ютерів, які спілкуються між собою за допомогою спеціальних протоколів передачі даних. DNS є одним з таких протоколів, який перетворює доменні імена на IP-адреси, що потрібні для з’єднання зі сторінкою вебсайту або сервером.

Комп’ютер, підключений до Інтернету, називається клієнтом. Клієнти – це звичайні користувачі, які підключаються до Інтернету за допомогою пристрою (наприклад, комп’ютера, підключеного до Wi-Fi, або мобільного телефону, підключеного до мобільної мережі) і програмного забезпечення, доступного на цьому пристрої (зазвичай браузера, такого як Firefox або Chrome). Клієнти можуть надсилати запити на інформацію до серверів. Сервери – це комп’ютери, на яких зберігаються вебсторінки, сайти або застосунки. Коли клієнт намагається отримати доступ до вебсторінки, копія сторінки завантажується з сервера на клієнтський комп’ютер і відображається в браузері користувача (рис. 1.1) [4].

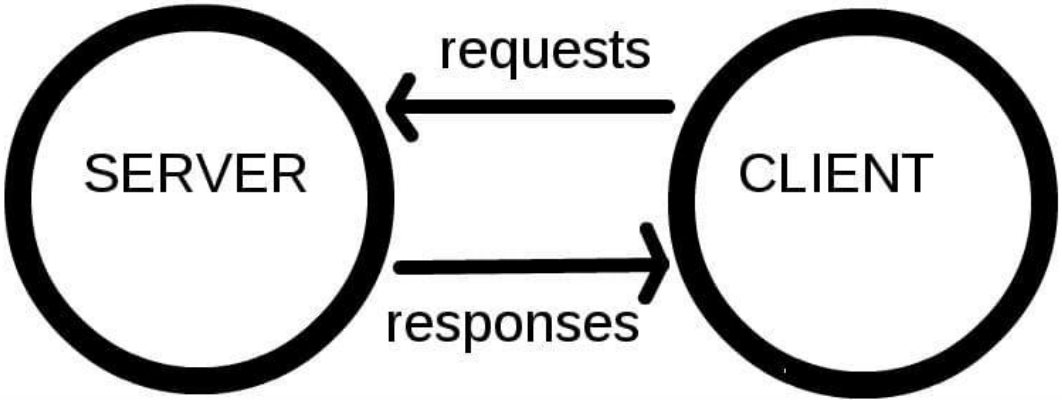

Рисунок 1.1 – Зв’язок між клієнтом та сервером

Запити надсилаються за допомогою протоколів передачі даних, таких як HTTP, який використовується для передачі вебсторінок. Коли клієнт надсилає запит, сервер обробляє його і надсилає відповідь у вигляді даних або вебсторінки, яку клієнт може переглянути у веббраузері. Крім того, для

забезпечення безпеки передачі даних, Інтернет використовує шифрування за допомогою SSL або його наступника, TLS [5].

Отже, ось як відбувається процес отримання вебсторінки в браузері:

– коли адреса вебсайту вводиться у браузер, браузер автоматично перетворює цю адресу в IP-адресу (цифрову адресу) за допомогою DNS-сервера. Цей процес дозволяє браузеру знайти відповідний сервер, де розміщений вебсайт;

– після того, як браузер знаходить адресу сервера, він надсилає HTTP-запит до сервера з проханням надіслати копію сайту клієнту. Це повідомлення та всі інші дані, що передаються між клієнтом і сервером, передаються через інтернет-з'єднання за допомогою протоколу TCP/IP;

– якщо сервер схвалює запит клієнта, він надсилає клієнту статус 200 OK, що означає, що клієнт може переглядати сайт. Потім сервер починає надсилати браузеру невеликі сегментовані файли сайту, які називаються пакетними даними;

– браузер збирає невеликі частини разом, щоб відобразити повний вебсайт. Пакети даних містять окремі елементи вебсторінки, такі як текст, зображення та відео. Пакети даних надсилаються браузеру в порядку, визначеному вебсайтом, щоб браузер міг об'єднати їх і відобразити вебсторінку у вікні.

1.3 Структура сучасного вебсайту

Структура сайту – це розташування його сторінок, категорій, підкатегорій та продуктів. Це своєрідний план, який показує логічні зв'язки між сторінками. З технічної точки зору, навігація вебсайту – це набір URL-адрес, розташованих у певному порядку. Вона тісно пов'язана з семантичним ядром. Воно визначає, які папки і документи повинні бути на сайті [6].

Якщо є семантичне ядро і розуміння того, який матеріал має бути на сайті, то має бути і схема структури ресурсу. Іншими словами, ієрархія, логічний і послідовний ланцюжок створення та подання інформації. Якого виду може бути схема, залежить від призначення сайту. Відповідно, обирається найбільш підходящий метод:

– алфавітна організація. Алфавітна організація вебсайту є методом структурування вмісту за допомогою алфавіту, що допомагає полегшити пошук необхідної інформації та зробити навігацію користувачів більш зручною. Головним елементом такої організації є алфавітний показник, що містить усі літери алфавіту. Він може розміщуватись у верхній частині сторінки сайту, де він стає посиланням на сторінки з вмістом, що починається на відповідну літеру. Також він може бути включений до бічної панелі для ще більш зручної навігації. Окрім алфавітного показника, алфавітна організація може використовувати алфавітний порядок для розташування вмісту на сторінці, що забезпечує зручність організації списків, каталогів та інших типів вмісту на сайті. Користувачі можуть швидко знаходити потрібну інформацію, використовуючи індексні посилання або пошукові функції. Алфавітна організація може бути корисною для різних типів вебсайтів, таких як блоги, інтернет-магазини, каталоги товарів, пошукові системи та багато інших. Вона допомагає користувачам легко знаходити необхідну інформацію та робить навігацію по сайту більш зручною;

– хронологічна. Хронологічна структура сайту відображає вміст сайту за допомогою дат, коли вміст був опублікований або створений, тому новіші матеріали відображаються спочатку, а старіші – в кінці. Це дозволяє користувачам швидко відслідковувати розвиток подій та оновлення вмісту на сайті. Для цього використовуються дата та час публікації, які можуть бути включені в заголовки, мітки або метадані кожної сторінки сайту. Крім того, користувачі можуть використовувати фільтри, щоб переглядати вміст за конкретну дату, місяць або рік. Хронологічна структура особливо корисна для новинних сайтів, блогів, журналів та інших сайтів з великою кількістю

оновлюваного вмісту. Також можуть використовуватися інші методи організації вмісту, такі як категорії, теги, пошукові функції та інші, щоб доповнити хронологічну структуру і допомогти користувачам знаходити інформацію, яку вони шукають, за різними критеріями;

– деревоподібна. Деревоподібна структура сайту використовує ієрархічну організацію сторінок та розділів, яка схожа на дерево. Згідно з цим методом, на вершині дерева знаходиться головна сторінка сайту, а далі слідують підсторінки та підкатегорії, що належать до головних категорій. Кожен розділ може містити підкатегорії, які розкривають тему більш детально. Таким чином, користувач може переходити на різні рівні глибини сайту, щоб знайти необхідну інформацію. Наприклад, якщо сайт присвячений спорту, то головна сторінка міститиме загальну інформацію про сайт та всі доступні види спорту. Підсторінки можуть включати кожен окремий вид спорту, а підкатегорії – конкретні команди, турніри та правила гри. Деревоподібна структура сайту забезпечує зручний та логічний пошук інформації для користувача, зокрема на великих сайтах з багатьма розділами та сторінками. Такий метод допомагає зберегти послідовність і зрозумілість структури сайту;

– тематична. Тематична організація сайту полягає у структуруванні вмісту сайту за тематичними категоріями. Цей підхід допомагає зрозуміти структуру сайту та легко знайти потрібну інформацію. Весь вміст сайту групується за тематикою, наприклад, різними видами мистецтва. Кожна тема може містити сторінки з описом творів, авторів, історії та техніки, пов’язаної з даною темою. Такий підхід допомагає зберегти логічну послідовність та зробити сайт зрозумілим для користувачів, а також швидко знайти необхідну інформацію;

– лінійна. Лінійна організація сайту визначається як метод структурування вмісту, де сторінки сайту відображаються в послідовній лінії, щоб користувачі могли переходити від однієї сторінки до іншої, пройшовши всі попередні сторінки в даній послідовності. Цей метод особливо підходить

для сайтів з великою кількістю інформації, де користувачі мають прочитати вміст у певному порядку, наприклад, сайти з онлайн-курсами або послідовними кроками для розв’язання певної проблеми. Основна перевага лінійної організації полягає в тому, що вона зберігає послідовність та логічність відображення вмісту, дозволяючи користувачам переходити на наступну сторінку з наступним кроком у процесі, що допомагає зберегти час та спростити навігацію. Однак, недоліком може стати обмеження у виборі користувача, який хоче переглядати інформацію в іншому порядку, тому вони можуть відчути необхідність повернутися до початку та шукати необхідну інформацію.

Загалом, існує декілька методів організації вмісту на сайті, кожен з яких має свої переваги та недоліки. Кожен метод організації може бути корисним в певних випадках, тому перед створенням сайту необхідно ретельно обміркувати мету та цільову аудиторію сайту, щоб обрати оптимальний метод організації вмісту.

1.4 Технології розпізнавання тексту із зображень

Розпізнавання тексту на зображеннях є технологією, яка має великий потенціал у галузі комп’ютерного зору та штучного інтелекту. Ця технологія дозволяє автоматично перетворювати текст на зображеннях у формат, який може бути оброблений комп’ютером, що значно полегшує роботу з документами та збільшує ефективність процесів, пов’язаних з обробкою тексту [7].

Основою технології є алгоритми машинного навчання, які навчають систему розпізнавання тексту на основі даних, що містять зображення з текстом та відповідні текстові дані. Після навчання система може автоматично визначати текст на зображеннях та перетворювати його у формат, який можна обробляти комп’ютером [8].

Технології розпізнавання тексту використовуються в багатьох галузях, таких як медицина, автомобільна промисловість, фінансовий сектор, маркетинг, архівна справа, друкарство та інші. Наприклад, в медицині системи розпізнавання тексту використовуються для аналізу медичних зображень, що допомагає лікарям отримувати більш точну інформацію про стан пацієнта. У автомобільній промисловості системи розпізнавання тексту можуть використовуватися для визначення номерів автомобілів на вулицях, що допомагає поліцейським у роботі.

Крім того, технології розпізнавання тексту можуть допомогти в управлінні транспортними потоками та виявленні порушень правил дорожнього руху.

Фінансовий сектор також може використовувати технології розпізнавання тексту для автоматизації процесів обробки фінансових документів. Наприклад, банк може використовувати системи розпізнавання тексту для автоматичного розпізнавання та обробки чеків, що дозволяє значно зменшити час та затрати на обробку цих документів [9].

У маркетингу технології розпізнавання тексту можуть використовуватися для аналізу зображень та визначення, який товар або бренд зображений на зображенні, що допомагає підвищити ефективність рекламних кампаній та взаємодії зі споживачами.

Технології розпізнавання тексту також можуть бути використані для архівної справи. Наприклад, система розпізнавання тексту може використовуватися для автоматичного розпізнавання та індексування документів, що дозволяє зменшити час та затрати на обробку та зберігання документів.

Друкарство також може скористатися технологіями розпізнавання тексту для автоматичного розпізнавання тексту на зображеннях та їх подальшої обробки. Наприклад, система розпізнавання тексту може використовуватися для автоматичного визначення розміру та місця

розташування тексту на зображенні, що допомагає поліпшити якість друку та зберегти час та затрати на обробку зображень.

Загалом, технології розпізнавання тексту із зображень є дуже важливими технологіями, що знаходять все більш широке застосування в різних галузях. Вони дозволяють автоматизувати процеси обробки документів та зображень, що значно зменшує час та затрати на ці процеси. Більше того, технології розпізнавання тексту допомагають поліпшувати якість обробки та аналізу даних, що дозволяє бізнесу приймати більш обґрунтовані рішення та покращувати ефективність своєї діяльності.

Проте, варто зазначити, що технології розпізнавання тексту не є повністю безпомилковими. Наприклад, у випадку розпізнавання тексту на зображеннях, неправильне освітлення, погана якість зображення або наявність інших елементів на зображенні можуть вплинути на точність розпізнавання тексту. Тому, перед використанням технологій розпізнавання тексту, варто уважно перевірити їх точність та здатність працювати в різних умовах.

Підсумовуючи вище сказане, технології розпізнавання тексту із зображень мають великий потенціал для застосування в різних галузях та сферах діяльності. Вони дозволяють автоматизувати процеси обробки документів та зображень, що зменшує час та затрати на ці процеси, а також покращує якість обробки та аналізу даних. Проте, варто пам’ятати, що точність роботи технологій розпізнавання тексту може залежати від багатьох факторів, тому важливо уважно перевіряти їх роботу перед використанням у виробничих процесах.

1.5 Постановка задачі

Після проведення аналізу предметної області, необхідно поставити конкретні завдання для розробки вебзастосунку з розпізнаванням тексту із зображень. Об’єктом роботи є дослідження тексту із зображень.

Метою роботи є розробка вебзастосунку з розпізнавання тексту із зображень.

Для досягнення цієї мети необхідно реалізувати наступні завдання:

– розробити вебзастосунок, який зможе приймати зображення та повертати розпізнаний текст;

– використовуючи готові бібліотеки та фреймворки, забезпечити високу точність розпізнавання тексту на зображеннях;

– розробити інтуїтивно зрозумілий та зручний інтерфейс користувача;

– перевірити працездатність розробленого вебзастосунку шляхом тестування та порівняння з аналогами;

Постановка цих завдань дозволить досягти мети дипломної роботи та створити вебзастосунок, який буде корисним у різних сферах діяльності.

2 МОДЕЛЮВАННЯ ВЕБЗАСТОСУНКУ

2.1 Архітектура вебзастосунків

Побудова архітектури застосунку включає в себе ряд заходів, що допомагають чітко визначити, як саме буде розроблена система. В даному випадку, я розглядаю лише верхній рівень визначення архітектури, який можна визначити при обмеженій кількості інформації.

Важливо зрозуміти, що не існує універсального рішення при визначенні принципу проєктування архітектури. Неможливо сказати, що конкретний архітектурний патерн підходить для будь-якої задачі і його слід використовувати виключно. У кожному випадку все залежить від конкретного проєкту та його завдань [10].

Хоча можна побудувати будь-яку систему на будь-якій архітектурі, це не завжди є доцільним рішенням. Загалом відокремлюють три типи верхньорівневої архітектури вебзастосунків, це:

- моноліт;
- мікросервіси;
- серверлес.

2.1.1 Моноліт

Архітектурний патерн Моноліт (Monolith) є одним з найбільш поширених патернів у розробці програмного забезпечення. Він описує структуру програмного забезпечення, де весь функціонал застосунку вбудований в один великий блок, відомий як моноліт (рис. 2.1) [10].

У такій архітектурі весь код застосунку розташовується в одному місці, що спрощує розробку, тестування та розгортання. Однак, збільшення обсягів

функціоналу та розміру проєкту може призвести до труднощів у підтримці та розширенні системи.

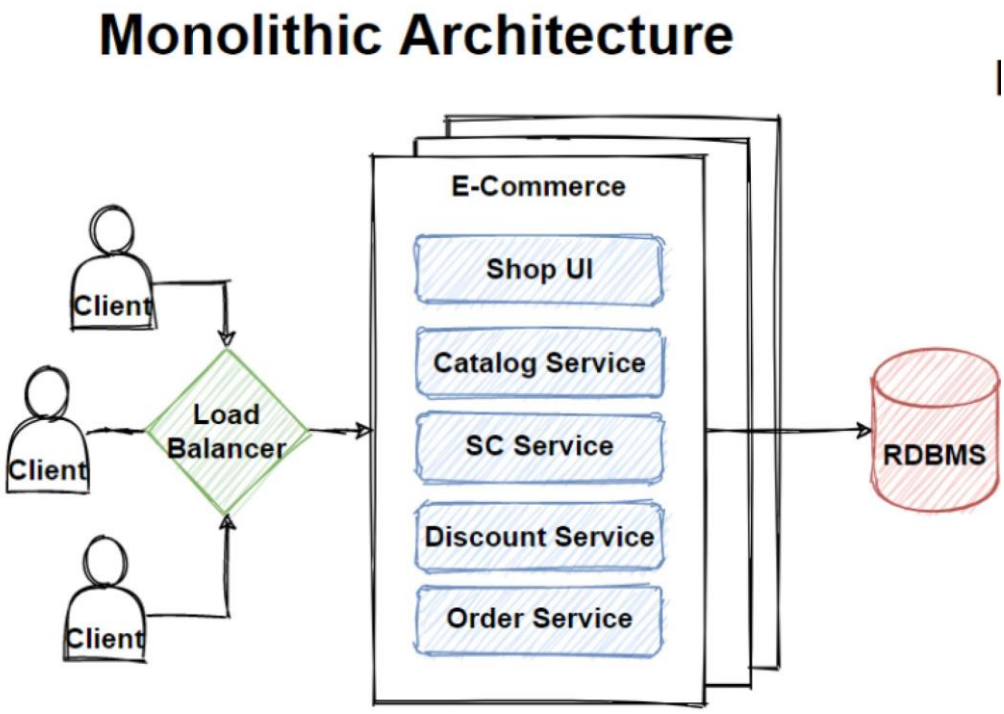

Рисунок 2.1 – Приклад архітектури «Моноліт»

Основними перевагами архітектурного патерну Моноліт є:

– простота. Відсутність складної інфраструктури та додаткових компонентів спрощує процес розробки та розгортання застосунку;

– швидкість розробки. Розробка застосунку в монолітній архітектурі зазвичай є швидшою та менш складною;

– простота тестування: Тестування всього застосунку в одному місці спрощує процес тестування та дозволяє легко виявляти та виправляти помилки.

Однак, існують деякі недоліки такої архітектури, серед яких:

– складність розширення. Збільшення обсягів функціоналу та розміру проєкту може призвести до труднощів у підтримці та розширенні системи;

– витратність масштабування. Зростання навантаження на застосунок може призвести до зниження продуктивності та необхідності розширення апаратного забезпечення;

– недостатня незалежність компонентів. У монолітної архітектури всі компоненти зазвичай залежать один від одного, що може призвести до складнощів.

У світі існують численні успішні проєкти, побудовані на базі архітектурного патерну Моноліт. Проте, наявність залежностей між компонентами може призвести до зниження ефективності розробки, оскільки необхідно враховувати взаємодію між ними при внесенні змін. Також, збільшення обсягів функціоналу та розміру проєкту може призвести до труднощів у підтримці та розширенні системи.

Недоліки монолітної архітектури зазвичай виникають при масштабуванні застосунку. Зростання навантаження на застосунок може призвести до зниження продуктивності та необхідності розширення апаратного забезпечення. Також, в разі збільшення обсягу коду можуть виникнути проблеми з його розгортанням та конфігурацією.

Усі перелічені недоліки монолітної архітектури можуть бути усунені за допомогою інших архітектурних патернів, таких як мікросервісна архітектура, Серверлес архітектура тощо. В той же час, монолітна архітектура може бути вибрана для невеликих проєктів або застосунків з невеликою кількістю функціоналу, які не вимагають високої масштабованості.

2.1.2 Мікросервіси

Архітектурний патерн Мікросервіси (Microservices) є підхідом до розробки програмного забезпечення, в якому застосунок складається з невеликих автономних компонентів, що називаються мікросервісами. Кожен мікросервіс відповідає за виконання певної функції та може бути розгорнутий та масштабований окремо від інших компонентів (рис. 2.2) [11].

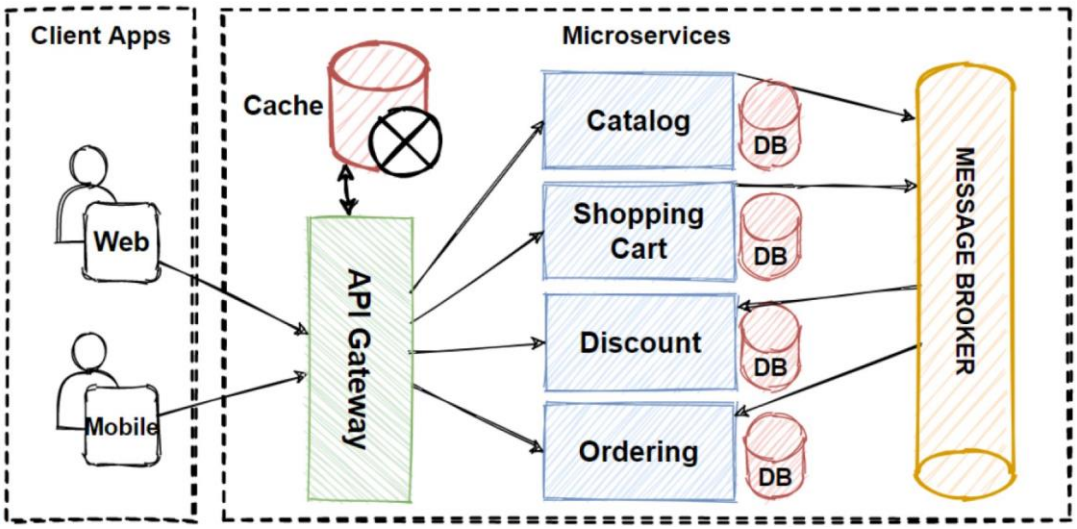

Рисунок 2.2 – Приклад архітектури «Мікросервіси»

Основні переваги архітектурного патерну Мікросервіси:

– гнучкість. Компоненти можуть бути розгорнуті та масштабовані незалежно від інших компонентів, що дозволяє швидко відповідати на зміни вимог та попиту;

– легкість в розгортанні. Кожен мікросервіс може бути розгорнутий на окремому сервері або контейнері, що дозволяє швидко розгортати та відлагоджувати окремі компоненти;

– висока доступність. Якщо один мікросервіс відмовляє, інші компоненти можуть продовжувати працювати, що забезпечує високу доступність системи в цілому;

– швидкість розробки. Розробники можуть працювати над окремими компонентами, що дозволяє швидко випускати нові функції та оновлення;

– легкість у тестуванні. Окремі компоненти можуть бути легко тестовані незалежно від інших компонентів, що дозволяє зменшити кількість помилок в системі.

Однак, існують деякі недоліки такої архітектури, серед яких:

– складність в управлінні. Мікросервіси можуть бути розроблені на різних мовах програмування та мати різні вимоги до інфраструктури, що може зробити управління системою складнішим;

– складність в тестуванні взаємодії між компонентами. Важко знайти та відлагоджувати помилки, які можуть виникати в процесі взаємодії між компонентами;

– накладні витрати на мережевий трафік. Збільшення кількості мікросервісів може призвести до збільшення мережевого трафіку та накладних витрат на мережу.

Усі ці недоліки можна уникнути або зменшити, якщо добре спроєктувати архітектуру та правильно розбити програму на компоненти. Також існують різні інструменти та фреймворки, які допомагають управляти мікросервісною архітектурою та зменшувати накладні витрати на управління та трафік.

2.1.3 Серверлес

Архітектурний патерн Серверлес (Serverless) – це підхід до розробки програмного забезпечення, в якому розробники можуть створювати та запускати функції без необхідності керування інфраструктурою, такої як сервери, віртуальні машини або контейнери (рис. 2.3) [10].

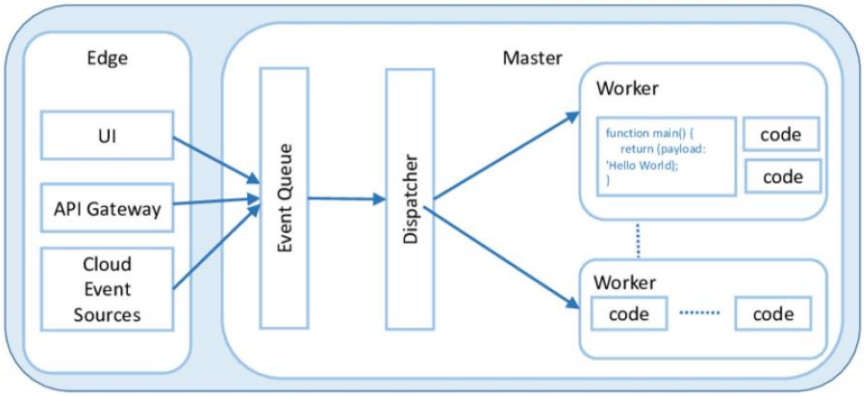

Рисунок 2.3 – Приклад архітектури «Серверлес»

Основні переваги архітектурного патерну Серверлес:

– економія коштів. Розробники можуть зосередитися на написанні функцій, інтерфейсів та бізнес-логіки, замість того, щоб тратити час та гроші на налаштування та управління інфраструктурою;

– гнучкість. Функції можуть бути розгорнуті в будь-який час, коли це необхідно, та можуть бути автоматично масштабовані в залежності від попиту;

– висока доступність. Функції можуть бути автоматично перенаправленні на інші сервери, якщо виникає проблема з доступністю одного з них;

– швидкість розгортання. Функції можуть бути швидко розгорнуті та запущені, що дозволяє розробникам прискорювати розробку та випускати нові функції швидше.

Однак, існують деякі недоліки такої архітектури, серед яких:

– обмежена підтримка. Деякі інфраструктурні сервіси можуть не підтримувати певні мови програмування або фреймворки;

– обмежена тривалість. Функції можуть бути обмежені за часом виконання, що може призвести до труднощів у виконанні довгих операцій;

– проблеми зі зберіганням даних. Деякі серверні сервіси можуть бути обмежені у збереженні даних на довгий термін, що може призвести до проблем зі збереженням та обробкою даних;

– важкість у відлагодженні. Під час відлагодження функції можуть бути складно знайти помилки через обмеження доступу до логів та інформації про середовище.

Проте, з урахуванням переваг та недоліків, патерн Серверлес може бути корисним для розробки деяких застосунків, особливо тих, які потребують високої гнучкості та швидкості розгортання. Деякі приклади використання архітектури Серверлес включають створення мікросервісів, обробки даних, відправки сповіщень та інші вебзастосунки.

2.2 Шаблони проєктування

Шаблони проєктування є важливою складовою процесу розробки програмного забезпечення, оскільки вони допомагають створити код, що має чітку структуру та організацію з певними правилами та конвенціями. У даному пункті розглянемо три найбільш популярні шаблони проєктування [12]:

– MVC;
– MVP;
– MVVM.

2.2.1 Шаблон проєктування MVC

Шаблон проєктування MVC (Model-View-Controller) є архітектурним підходом до розробки програмного забезпечення, який розділяє застосунок на три основні компоненти: модель (Model), представлення (View) та контролер (Controller) (рис. 2.4) [13].

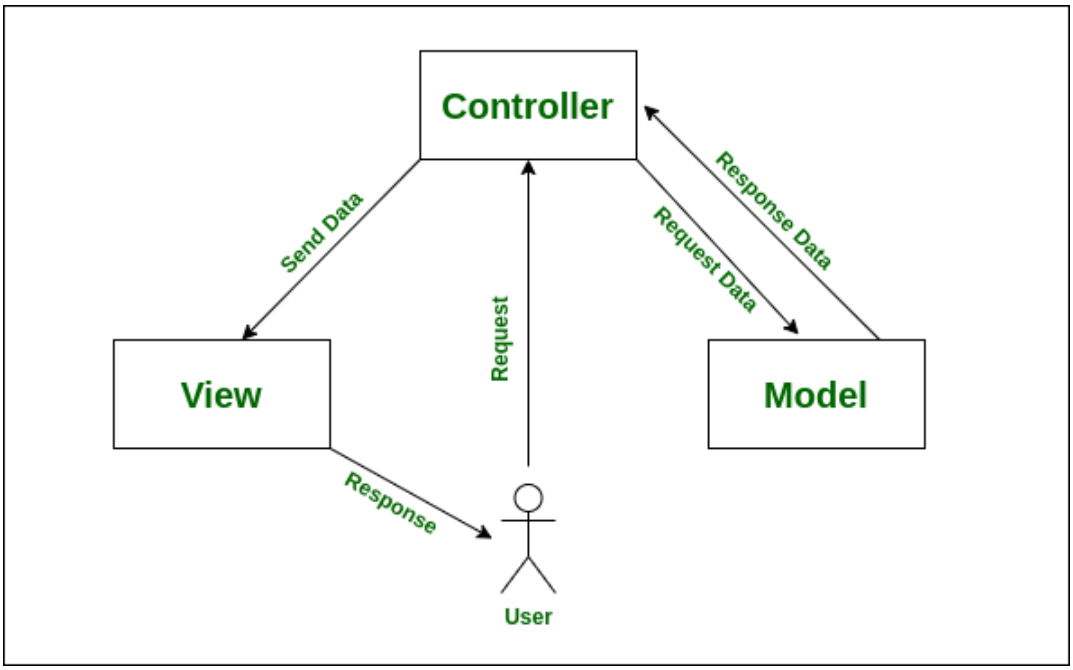

Рисунок 2.4 – Приклад шаблону MVC

Модель представляє собою складову системи, яка відповідає за збереження та обробку даних застосунку. Це може бути база даних або будь-яке інше джерело даних.

Представлення відображає дані користувачеві та дозволяє користувачеві взаємодіяти з застосунком. Це може бути, наприклад, вебсторінка, графічний інтерфейс користувача або будь-який інший спосіб відображення даних.

Контролер управляє взаємодією між моделлю та представленням. Він обробляє вхідні дані від користувача та передає відповідні команди моделі та представленню.

Шаблон проєктування MVC дозволяє розділити логіку застосунку на незалежні компоненти, що забезпечує його високу розширюваність та повторне використання коду. Крім того, цей підхід дозволяє ефективно використовувати ресурси та забезпечує зручний інтерфейс для користувача.

Плюси цього шаблону:

– шаблон проєктування MVC дозволяє розбити код на окремі компоненти, що робить його більш організованим та сприяє зручності розуміння та внесення змін в майбутньому;

– можливість окремого тестування кожного компонента в рамках шаблону MVC дозволяє знизити кількість помилок та покращити якість коду;

– шаблон дозволяє вносити зміни до окремих компонентів без впливу на решту коду, що забезпечує більшу гнучкість програмного продукту.

До мінусів можна віднести неправильну організацію розділення на компоненти, що може зробити код більш складним та заплутаним в рамках шаблону MVC.

2.2.2 Шаблон проєктування MVP

Шаблон проєктування MVP (Model-View-Presenter) – це архітектурний підхід, який використовується для розробки програмного забезпечення з графічним інтерфейсом користувача. В основі цього підходу лежить розділення програми на три основні компоненти: модель (Model), представлення (View) та презентер (Presenter) (рис. 2.5) [14].

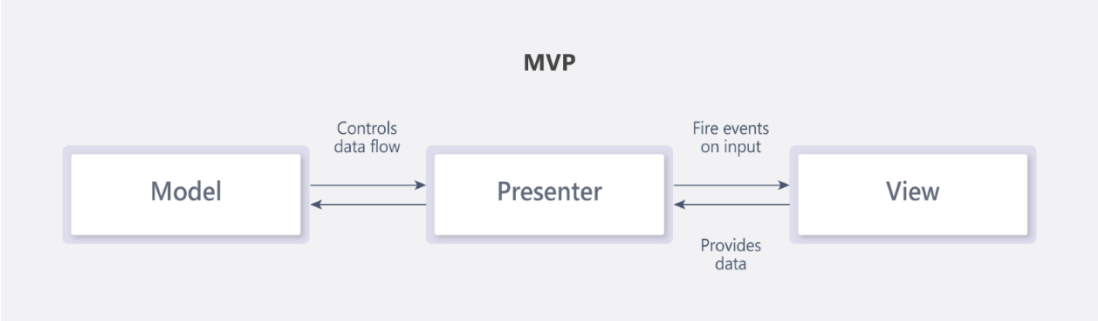

Рисунок 2.5 – Приклад шаблону MVP

Модель – це компонент, що містить бізнес-логіку програми та забезпечує зберігання та обробку даних.

Представлення – це компонент, який відповідає за відображення інтерфейсу користувача та взаємодію з користувачем.

Презентер – це компонент, який використовується для зв’язку між моделлю та представленням. Він отримує дані від моделі та передає їх у відповідні елементи у представленні.

Загалом, шаблон проєктування MVP є потужним інструментом для розробки програмного забезпечення з графічним інтерфейсом користувача. Він дозволяє забезпечити чітке розділення між компонентами програми та підвищити її розширюваність та підтримуваність. Крім того, MVP дозволяє створювати більш структурований та організований код, що дозволяє зберегти час та ресурси при розробці програмного забезпечення.

Плюси цього шаблону:

– розподіл відповідальностей між різними компонентами шаблону допомагає знизити залежність між кодом;

– за допомогою шаблону можна змінювати один з компонентів без впливу на решту.

Мінусом цього шаблону є створення Presenter для кожного View, що збільшує складність застосунку.

2.2.3 Шаблон проєктування MVVM

Шаблон проєктування MVVM (Model-View-ViewModel) є одним з найпоширеніших шаблонів проєктування програмного забезпечення, який використовується для розробки застосунків з графічним інтерфейсом користувача. Він дозволяє відокремити логіку бізнес-логіки від графічного інтерфейсу та забезпечує простір для тестування коду. MVVM складається з трьох основних компонентів: моделі (Model), представлення (View) та моделі представлення (ViewModel) (рис. 2.6) [15].

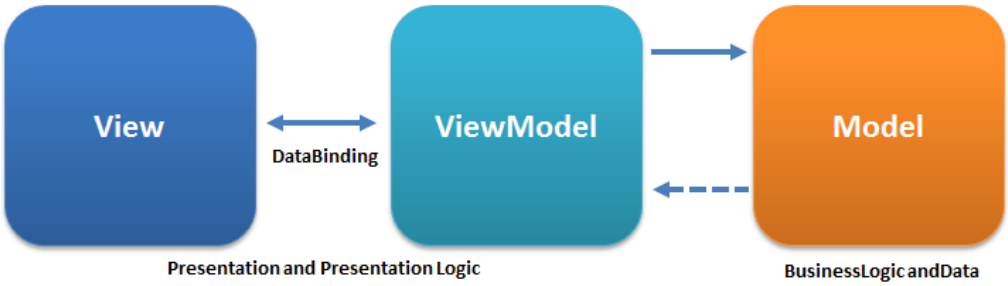

Рисунок 2.6 – Приклад шаблону MVVM

Модель відповідає за управління даними та бізнес-логікою застосунку. Вона не залежить від будь-яких інших компонентів і не знає нічого про представлення або модель представлення.

Представлення відображає дані користувачеві та обробляє події, що пов’язані з користувачем. Воно знає про модель представлення, але не знає нічого про модель.

Модель представлення діє як посередник між моделлю та представленням. Вона отримує дані від моделі та надсилає їх до представлення для відображення. Вона також отримує дані від представлення та передає їх до моделі для зберігання.

MVVM зазвичай використовує зв’язування даних (data binding) для зв’язку представлення та моделі представлення. Це означає, що зміни в моделі представлення автоматично оновлюють відповідні елементи в представленні, та зміни від користувача автоматично оновлюють відповідні властивості моделі представлення.

Плюси цього шаблону:

– розділення бізнес-логіки від графічного інтерфейсу відбувається чітко та належним чином;

– забезпечує більш простий процес тестування;

– надає можливість роботи з окремими складовими частинами застосунку.

Мінуси:

– вимагає додаткових інструментів та навичок для реалізації;

– вимагає більш глибокого розуміння для застосування.

2.3 Технології обробки зображень в вебзастосунках

Розвиток технологій у сфері фотографії та графіки дозволив зробити обробку зображень в вебзастосунках досить поширеною. Сьогодні на ринку існує безліч бібліотек та фреймворків, які дозволяють працювати з зображеннями в вебзастосунквх [16, 17].

Для розпізнавання тексту на зображенні використовуються різні алгоритми та технології, включаючи OCR. Цей алгоритм працює шляхом аналізу зображення та ідентифікації символів, які воно містить. OCR може бути реалізований за допомогою різних програмних бібліотек, таких як Tesseract, Google Cloud Vision та Amazon Rekognition. Використання машинного навчання дозволяє покращити точність розпізнавання тексту [18, 19].

Наприклад, Tesseract – це відкрите програмне забезпечення для розпізнавання тексту, розроблене командою Google. Ця бібліотека може розпізнавати багато мов та форматів зображень, включаючи рукописний текст. Tesseract доступний для різних мов програмування та операційних систем, його можна навчити розпізнавати нові мови та шрифти [20].

Незважаючи на потужність Tesseract, результати розпізнавання можуть бути неточними, особливо якщо зображення не є якісним або містить багато шуму. Тим не менш, Tesseract – це потужний інструмент для розпізнавання тексту, який може бути використаний для автоматизації обробки документів, підвищення ефективності роботи та інших завдань.

2.4 Розробка моделі вебзастосунку

Для успішного розроблення вебзастосунку, як правило, необхідно враховувати потреби користувачів та їх взаємодії з програмою. Перед початком розробки важливо провести дослідження, щоб збільшити розуміння потреб користувачів та їх поведінки. Це можна зробити, наприклад, за допомогою опитувань, фокус-груп або аналізу конкурентів [21, 22].

Після визначення потреб користувачів, наступним кроком є вибір технології розробки. Це важливо зробити на основі потреб проєкту та технічних можливостей платформи, на якій буде запущений продукт. Наприклад, якщо вебзастосунок повинен бути швидким та масштабованим, то

можна використовувати технології, які дозволяють розробляти високопродуктивні застосунки, такі як Node.js, React або Angular. Для свого вебзастосунку обрано React, адже він має високу продуктивність та підтримку великої спільноти, що надає великий вибір інструментів для кращої взаємодії з користувачем.

Крім того, важливо вибрати правильну архітектуру для вебзастосунку. Архітектура вебзастосунку визначається залежно від багатьох факторів, таких як обсяги даних, кількість користувачів, рівень безпеки та складність функцій. Роблячи висновки з потреб вебзастосунку обрано використовувати монолітну архітектуру, що означає, що весь код застосунку розміщується в єдиному блоку. Цей підхід дозволяє простішу розробку та підтримку застосунку, оскільки всі його компоненти взаємодіють між собою без необхідності у складних механізмах комунікації між ними.

Шаблон проєктування (або дизайн-патерн) є стандартним підходом до розв’язання типових проблем проєктування програмного забезпечення. Він дає можливість розробникам використовувати вже перевірені та ефективні рішення для розв’язання проблем, які зазвичай виникають під час розробки програмного забезпечення. Найкращим варіантом для вебзастосунку буде шаблон проєктування MVC, адже він дозволяє розділити компоненти вебзастосунку на три логічні частини: модель, вид та контролер. Цей підхід дозволяє забезпечити більшу гнучкість та розширюваність вебзастосуку, оскільки кожна частина може бути розроблена та тестована незалежно від інших частин.

Технології обробки зображень в вебзастосунках є дуже важливими, оскільки вони дозволяють розширити можливості вебзастосунків та покращити користувацький досвід. Tesseract є найкращим варіантом для вебзастосунку, адже це потужна інструментальна бібліотека, яка забезпечує високу точність OCR (розпізнавання символів зображення) для різноманітних мов та документів.

Отже, обравши необхідні інструменти для вебзастосунку, наступним кроком буде його розробка. При цьому важливо забезпечити якість програмного продукту та його стабільну роботу, зокрема умови високих навантажень та забезпечення безпеки даних.

3 РОЗРОБКА ВЕБЗАСТОСУНКУ ТА ТЕСТУВАННЯ

3.1 Обґрунтування вибору середовища програмної реалізації

У рамках кваліфікаційної роботи був розроблений застосунок з розпізнавання тексту із зображень. Для реалізації було обрано середовище розробки JavaScript. JavaScript є широко використовуваною мовою програмування, яка дозволяє створювати вебзастосунки та інтерактивні елементи на стороні клієнта. Для розробки застосунку з розпізнавання тексту із зображень використання JavaScript дозволяє зручно і легко інтегрувати функціональність розпізнавання тексту безпосередньо в вебзастосунок [23].

JavaScript (JS) має безліч переваг, які зробили його однією з найпопулярніших мов програмування. Ось кілька основних переваг JavaScript:

– універсальність. JavaScript є мовою програмування, яка може виконуватися в браузері користувача, а також на сервері за допомогою платформи Node.js. Це робить його універсальним і дозволяє розробникам створювати повноцінні вебзастосунки як на клієнтській, так і на серверній стороні;

– простота використання. JS має простий синтаксис, який легко читається і розуміється навіть новими розробниками. Він базується на мові програмування ECMAScript, яка визначає стандарти для JavaScript, що робить його більш передбачуваним і легко освоюваним;

– масштабованість. JS підтримує модульність, що дозволяє розбивати застосунки на багато незалежних компонентів, що спрощує розробку і підтримку коду. Завдяки цьому JavaScript використовується не тільки для створення невеликих вебсторінок, але й для розробки складних вебзастосунків і навіть серверних застосунків;

– велике співтовариство та екосистема. JavaScript є однією з найпопулярніших мов програмування, тому має велике співтовариство розробників. Це означає, що завжди можна знайти безліч ресурсів, бібліотек,

фреймворків та інструментів, які допоможуть у розробці. Крім того, існують численні спільноти, форуми та онлайн-курси, де можна отримати допомогу і підтримку від досвідчених розробників.

Це лише кілька переваг JavaScript, які роблять його потужним і популярним інструментом для розробки вебзастосунків та інших програмних рішень.

Використовуючи JavaScript можна не задумуватись про кросплатформенність, адже JavaScript є мовою, яка вбудована в браузери, тому код, написаний на JavaScript, автоматично виконується на будь-якій платформі, яка підтримує браузер. Це означає, що вам не потрібно писати окремий код для кожної платформи або використовувати різні мови програмування для різних пристроїв.

Незалежно від того, чи користувач відкриває ваш вебзастосунок на комп'ютері, смартфоні, планшеті або навіть телевізорі з підтримкою браузера, JavaScript код буде виконуватись і забезпечувати необхідну функціональність.

Завдяки цій універсальності, ви можете розробляти вебзастосунки, які працюють на різних платформах, з одним і тим же кодом. Це збільшує ефективність розробки і спрощує підтримку вашого застосунку в майбутньому.

Додатково до JavaScript було обрано популярний та потужний фреймворк React.js. Він забезпечував високу продуктивність, гнучкість та широкі можливості для розробки вебінтерфейсу. Крім того, React.js мав велику спільноту розробників, що значно спрощувало пошук допомоги та знайдення необхідних бібліотек та компонентів.

Для стилізації елементів обрано використовувати бібліотеку стилів Bootstrap, яка надає широкий набір готових класів. Bootstrap спрощує процес розробки, дозволяючи розміщувати компоненти на сторінці, налаштовувати їх вигляд і поведінку за допомогою простих класів (рис. 3.1). Завдяки цьому, розробникам не потрібно витрачати багато часу на написання власних стилів з нуля.

```
<div className="container d-flex flex-column align-items-center">
```

Рисунок 3.1 – Приклад класів Bootstrap

Bootstrap також надає можливість розширення за допомогою власних стилів. Розробники можуть додавати власні CSS-класи або змінювати існуючі класи, щоб налаштувати вигляд компонентів під свої потреби. Крім того, є можливість використання плагінів, які додають додаткові функціональні можливості, наприклад, каруселі, модальні вікна, валідацію форм і багато іншого.

Для реалізації обробки зображення використано бібліотеку Tesseract.js. Ця бібліотека є портівкою популярного OCR-движка Tesseract на JavaScript, що дозволяє здійснювати розпізнавання тексту на зображеннях безпосередньо у браузері або на сервері.

Однією з головних переваг Tesseract.js є його швидкість та висока точність розпізнавання. Бібліотека використовує нейромережеву модель, навчену на великій кількості зображень, що дозволяє досягти вражаючих результатів у розпізнаванні тексту. Вона підтримує багато мов, включаючи англійську, іспанську, французьку, німецьку, українську та багато інших.

Завдяки своїй простоті використання, швидкості та точності, Tesseract.js став популярним вибором для реалізації обробки зображень та розпізнавання тексту у вебпроєктах. Він забезпечує широкі можливості для розширення функціональності і налаштування під конкретні потреби, що дозволяє створювати потужні та ефективні застосунки для обробки тексту на зображеннях [24].

Додатково потрібно вибрати середовище розробки програмного забезпечення. Переглянувши відомості про такі середовища програмування, як Eclipse, Visual Studio Code, WebStorm, було обрано Visual Studio (VS) Code, що володіє наступними перевагами:

– легкість використання. VS Code має дружній інтерфейс користувача, який дозволяє швидко орієнтуватися та почати працювати. Він має інтуїтивно зрозумілі команди та швидкі клавіатурні скорочення;

– розширюваність. VS Code забезпечує можливість розширювати його функціональність за допомогою розширень. У вас є доступ до великого екосистеми розширень, які допомагають вам в роботі з різними мовами програмування, системами керування версіями та іншими інструментами;

– велика спільнота. VS Code має широку та активну спільноту розробників, де ви можете знайти відповіді на свої питання, отримати підказки та поділитися своїм досвідом. У спільноті є безліч розширень, тем оформлення та інших ресурсів, які полегшують вашу роботу.

Отже, розробка здійснювалася у середовищі Visual Studio Code з використанням мови JavaScript з фреймворком React.js, бібліотекою стилів Bootstrap та бібліотекою OCR Tesseract.js.

3.2 Програмна реалізація

Вебзастосунок було вирішено розробляти тільки на фронтенд частині з кількох причин. По-перше, це дозволило швидше розгортати ідеї без необхідності встановлення серверів та баз даних. Крім того, такий підхід дозволяв зосередитися на розробці інтерфейсу користувача та забезпеченні його зручності та ефективності [25].

По-друге, фокусування на фронтенд частині дозволило зменшити вимоги до обладнання та інфраструктури, необхідної для запуску застосунку. Замість того, щоб мати потужний серверний парк і бази даних, розробники могли обмежитися використанням хмарних сервісів або легких серверів для збереження необхідної інформації.

По-третє, розробка тільки на фронтенді дозволила економити час і зусилля, які були б витрачені на взаємодію з серверною частиною. Замість

того, щоб вирішувати питання зв’язку з сервером, обробки даних та керування базами даних, розробники могли концентруватися на покращенні функціональності та вигляду користувацького інтерфейсу.

Однак, важливо зазначити, що такий підхід має свої обмеження. Наприклад, при використанні тільки фронтенд розробки можуть виникати питання щодо безпеки даних. Також, якщо застосунок потребує складних операцій обробки даних або великої кількості обміну даними з сервером, може виникнути потреба в серверній частині для оптимізації та забезпечення ефективності [26, 27].

Після вибору React.js вебзастосунку було розроблено гнучку та масштабовану архітектуру. Були визначені головні компоненти і їх взаємодія. Зазвичай у React.js використовуються компоненти класів або функціональні компоненти, залежно від потреб проєкту (рис. 3.2, 3.3). Компоненти були створені для різних частин інтерфейсу, таких як заголовок, форми вводу даних, та інші (рис. 3.4).

```
import React, { Component } from 'react'

class Result extends Component {
  render() {
    return (
      <div>Result</div>
    )
  }
}

export default Result
```

Рисунок 3.2 – Приклад класового компонента

```
import React from 'react'

const Result = () => {
  return (
    <div>Result</div>
  )
}

export default Result
```

Рисунок 3.3 – Приклад функціонального компонента

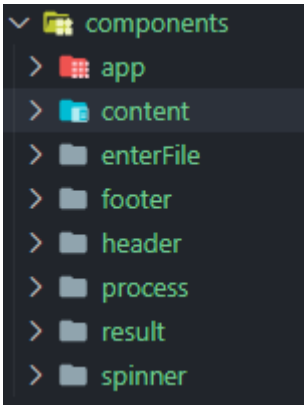

Рисунок 3.4 – Компоненти вебзастосунку

Розглянемо ці компоненти:

– Header: шапка вебзастосунку;

– Footer: підвал вебзастосунку;

– EnterFile: компонент вебзастосунку, який надає можливість завантажувати зображення;

– Process: компонент який відображає статус та прогрес розпізнавання тексту;

– Result: компонент який приймає результати розпізнавання тексту та відображає їх на екрані;

– Spinner: компонент який відображав елемент loader;

– Content: компонент який слугував обгорткою для EnterFile, Process, Result;

– App: головній компонент який збирає в собі всі інші компоненти й передається для відображення в браузері.

Одним з ключових аспектів розробки вебзастосунку була реактивна поведінка. За допомогою контексту React, стан застосунку було організовано в одному місці та контролювалося за допомогою змінних стану (рис. 3.5). Це дозволило зручно відстежувати та керувати змінами стану застосунку та автоматично оновлювати відображення інтерфейсу користувача при зміні даних.

```
const [status, setStatus] = useState();
```

Рисунок 3.5 – Приклад створення зміни стану

Після створення всіх компонентів та їх стилізації можна перейти до наступного кроку.

Щоб почати використовувати Tesseract.js, спочатку потрібно встановити його за допомогою менеджера пакетів npm або підключити через CDN. Після цього можна використовувати бібліотеку у своєму JavaScript-коді (рис. 3.6).

```
import Tesseract from 'tesseract.js';

Tesseract.recognize(
  'https://tesseract.projectnaptha.com/img/eng_bw.png',
  'eng',
  { logger: m => console.log(m) }
).then(({ data: { text } }) => {
  console.log(text);
})
```

Рисунок 3.6 – Приклад використання Tesseract.js

Основний процес розпізнавання тексту за допомогою Tesseract.js включає кілька кроків. По-перше, потрібно завантажити зображення, на якому

потрібно розпізнати текст. Далі, можна викликати функцію розпізнавання тексту, передаючи завантажене зображення. Після обробки Tesseract.js поверне розпізнаний текст або результати розпізнавання у вигляді об’єкта з координатами та знайденими словами [28].

Після добавляння Tesseract.js можна сказати, що наш вебзастосунок готовий. Для вільного доступу до нього вирішено використовувати хостинг GitHub.

3.3 Інструкція користувача

Загальні поради та рекомендації:

– перед завантаженням зображення переконайтеся, що воно має достатньо високу якість та чіткість. Розпізнавання тексту може бути кращим на якісних зображеннях;

– уникайте завантаження занадто великих файлів зображень, оскільки це може сповільнити процес розпізнавання та обробки;

– перевірте, чи мова тексту на зображенні відповідає мові, яку підтримує вебзастосунок для розпізнавання. Деякі вебзастосунки можуть підтримувати лише обмежений набір мов;

– у разі, якщо розпізнання тексту на зображенні недостатньо точне або неправильне, спробуйте інші зображення з кращою якістю або вирівняйте зображення перед завантаженням.

Крок 1. Вхід до вебзастосунку. Відкрийте браузер і перейдіть до вебзастосунку для розпізнавання тексту із зображень (рис. 3.7, 3.8).

Рисунок 3.7 – Посилання на вебзастосунок

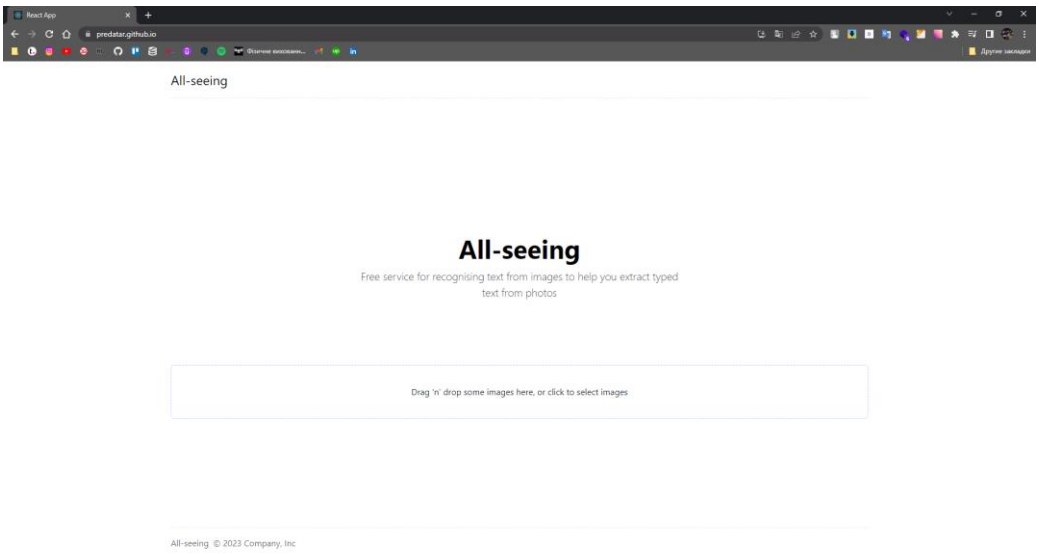

Рисунок 3.8 – Головна сторінка вебзастосунку

Крок 2. Завантаження зображення. На головній сторінці вебзастосунку знайдіть кнопку «Завантажити зображення» або аналогічний елемент управління (рис. 3.9). Клацніть на цю кнопку, щоб відкрити діалогове вікно вибору файлу (рис. 3.10). Виберіть зображення з вашого комп’ютера та натисніть «Відкрити» або аналогічну кнопку у діалоговому вікні (рис. 3.11).

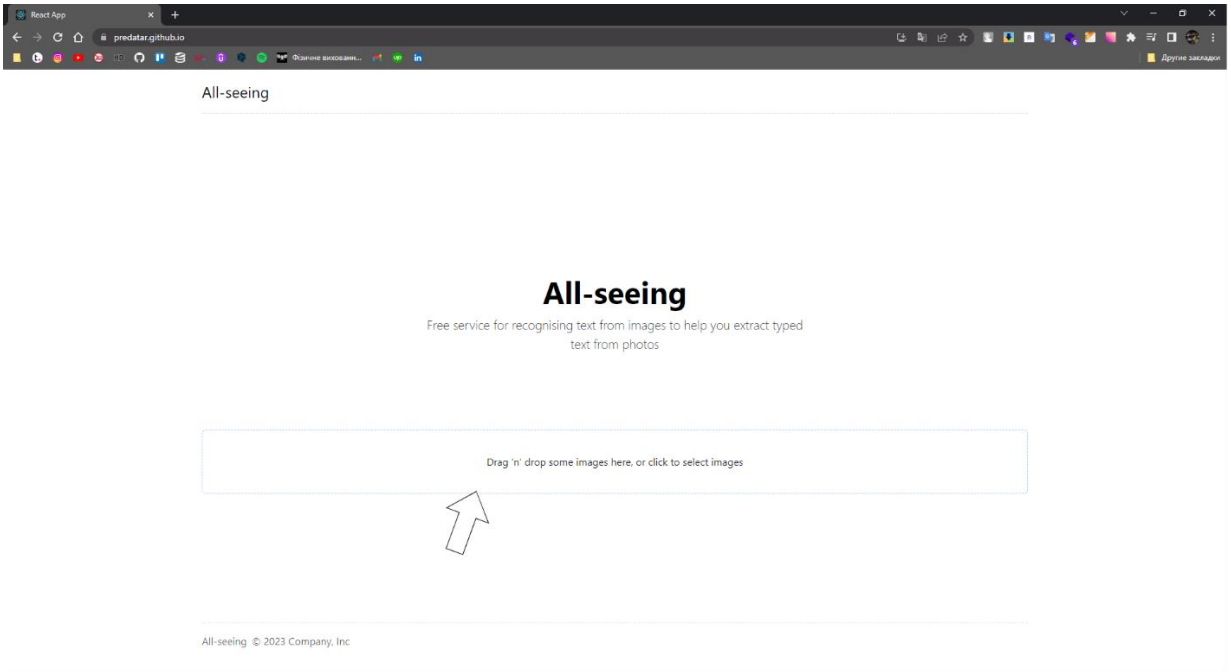

Рисунок 3.9 – Елемент завантаження зображення

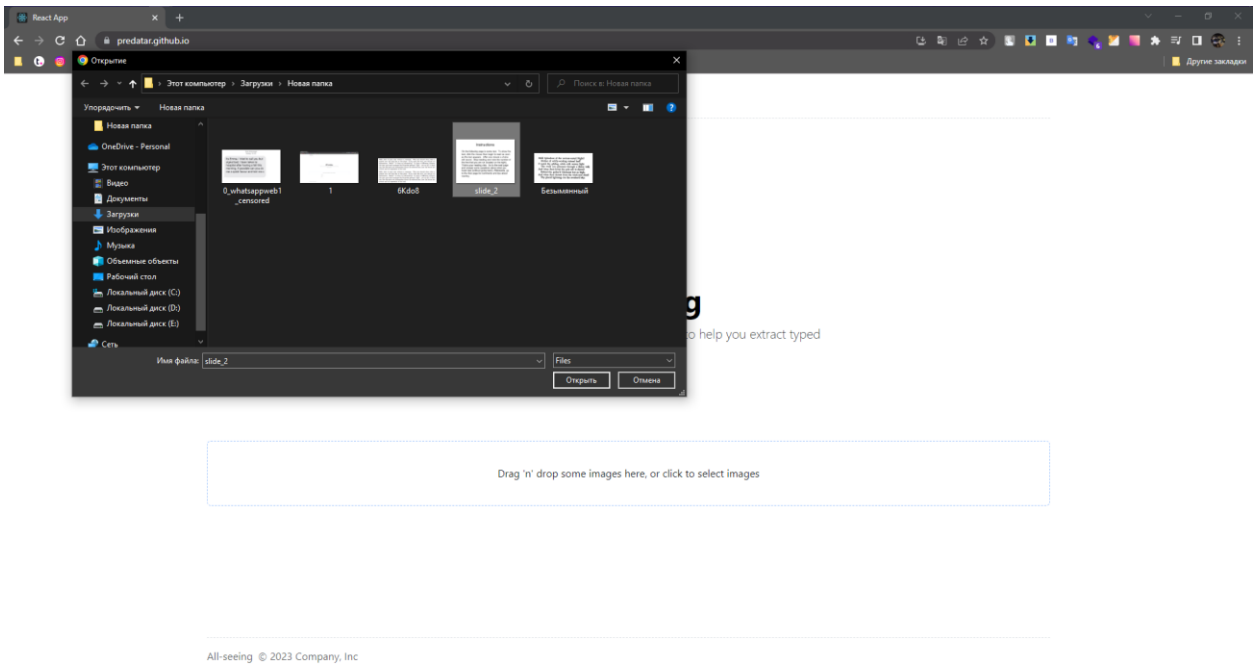

Рисунок 3.10 – Відкрите діалогове вікно

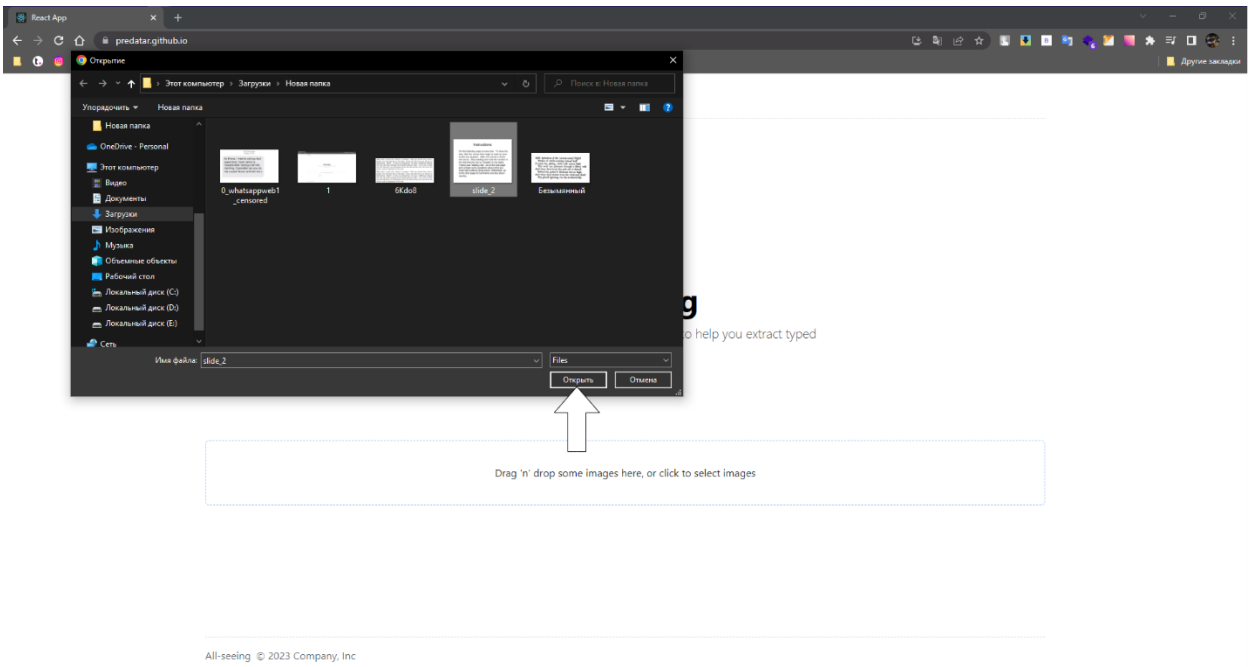

Рисунок 3.11 – Обрання зображення та підтвердження вибору

Крок 3. Розпізнавання тексту. Після завантаження зображення, вебзастосунок автоматично розпізнає текст на ньому (рис. 3.12). Зачекайте кілька секунд, поки процес розпізнавання завершиться. Після завершення розпізнавання текст зображення буде відображено на екрані вебзастосунку.

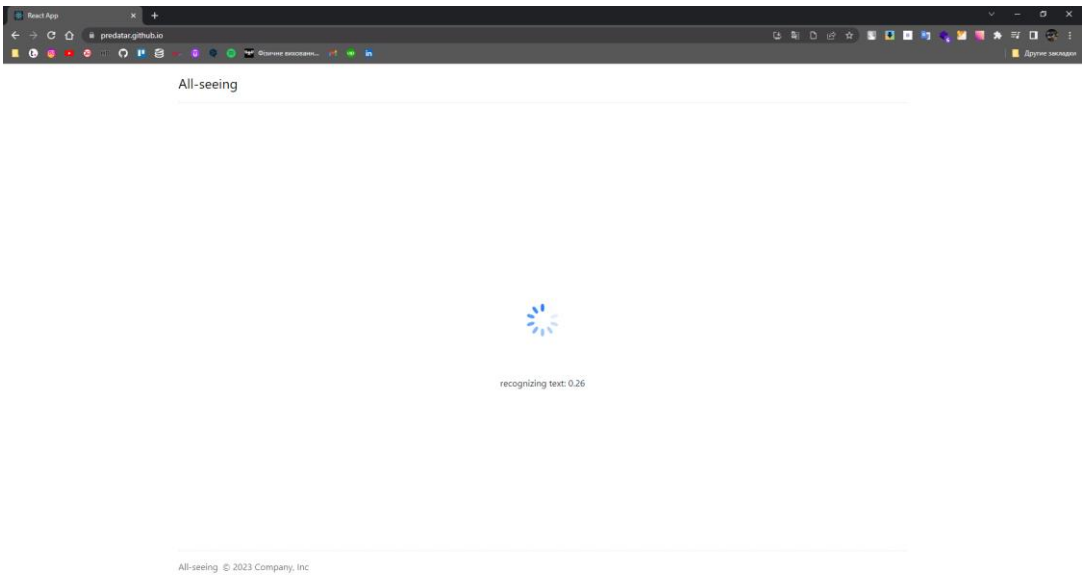

Рисунок 3.12 – Процес розпізнавання тексту

Крок 4. Результати та подальші дії. Вебзастосунок може показати розпізнаний текст прямо на екрані (рис. 3.13). Якщо текст відображається прямо на екрані, ви можете скопіювати його в буфер обміну для використання в інших програмах (рис. 3.14).

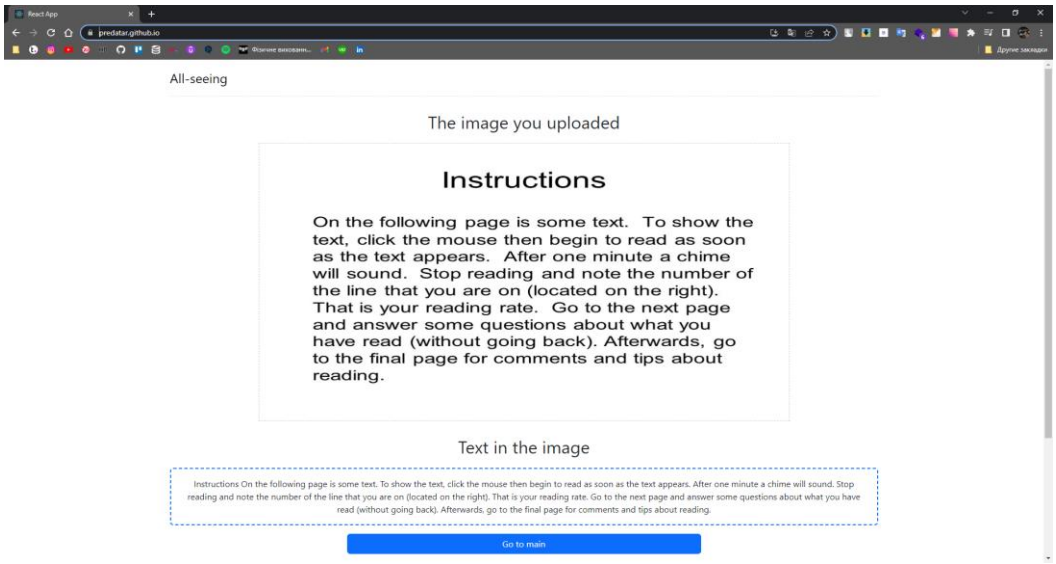

Рисунок 3.13 – Розпізнаний текст

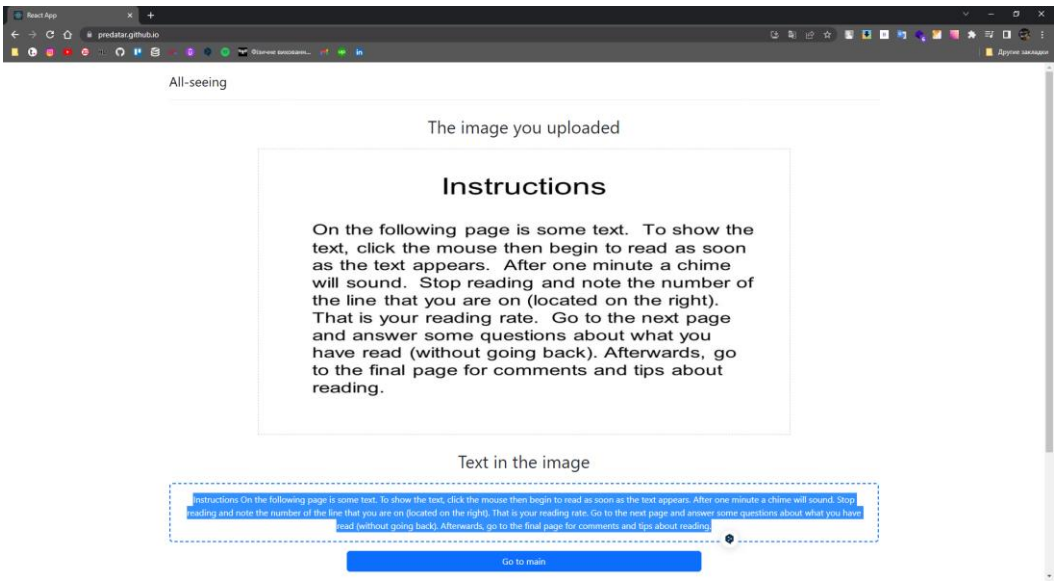

Рисунок. 3.14 – Можливість копіювання тексту

Крок 5. Вихід з вебзастосунку. Закрийте вкладку браузера або сам браузер (рис. 3.15).

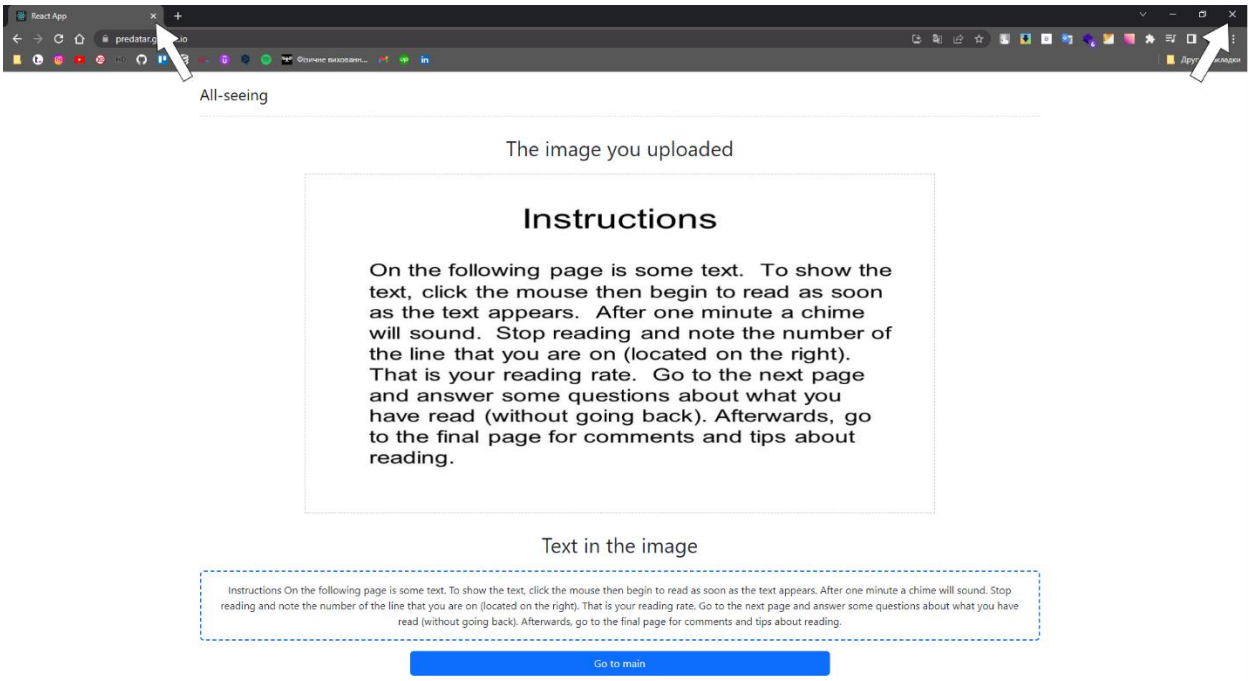

Рисунок. 3.15 – Вихід з вебзастосунку

3.4 Тестування розробленої моделі

Тестування програмного забезпечення є процесом, який включає в себе дослідження програмного продукту з метою оцінки його якості. Цей процес формально перевіряє, чи програмний продукт задовольняє використовувані методи і методології, і виявляє відсутність дефектів (помилок) згідно з цими методами. Важливо зауважити, що неможливо гарантувати повну відсутність дефектів у програмному продукті, з урахуванням людського фактора, який присутній на всіх етапах життєвого циклу ПЗ.

Після завершення розробки вебзастосунку необхідно протестувати його, щоб переконатися, що він відповідає функціональним вимогам та успішно реалізує задуману функціональність для сценаріїв використання, для яких він був розроблений і реалізований.

Для тестування обрано 5 зображення: 3 з різною якістю, 1 з текстом під нахилом і 1 з маленьким текстом.

3.4.1 Тестування зображень з різною якістю зображення

Перше тестування проводимо на зображенні з гарною якістю (рис. 3.16).

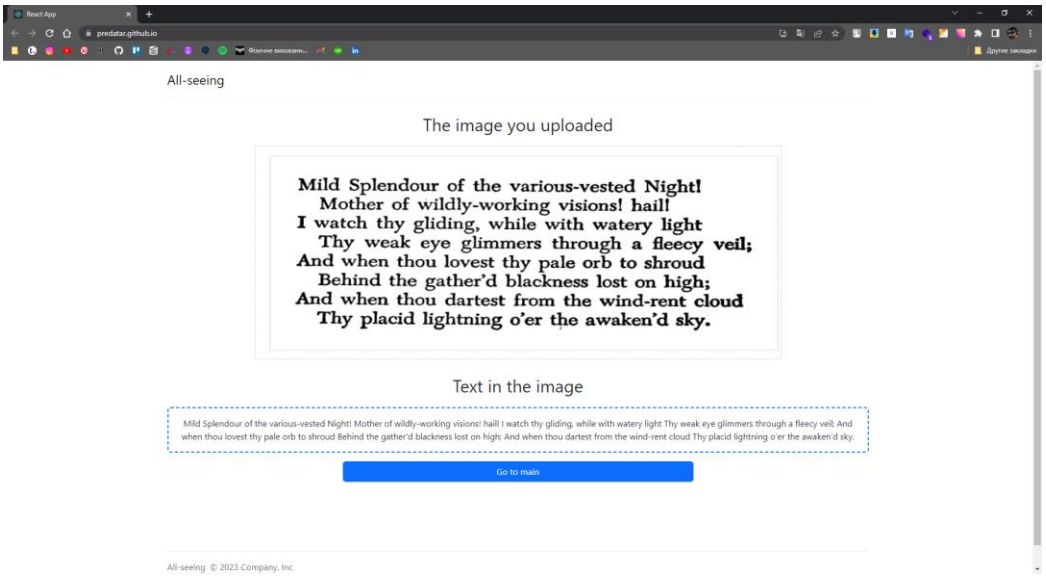

Рисунок 3.16 – Тестування №1. Зображення з гарною якістю

Як можна помітити, вебзастосунок добре розпізнає текст на зображенні з гарною якістю. Однак, коли якість зображення не є ідеальною, можуть виникати проблеми з розпізнаванням тексту. Наприклад, якщо фотографія була зроблена у поганих умовах освітлення або з великим рівнем шуму, то точність розпізнавання може суттєво знизитися (рис. 3.17).

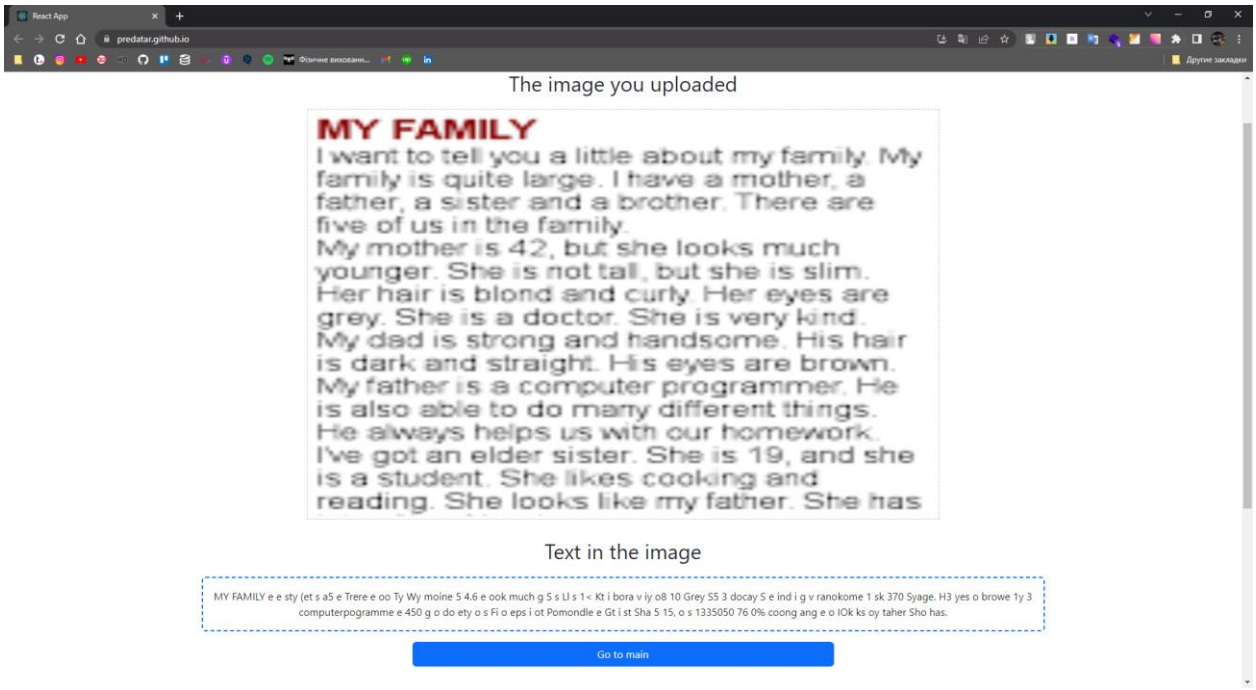

Рисунок 3.17 – Тестування №2. Зображення з поганою якістю

Нажаль через погану якість зображення текст розпізнано теж погано. Варто зауважити, що такі зображення є рідкістю сьогодні, адже пристрої сканування працюються гарно та створюють зображення в гарній якості.

Ще одне зображення для тестування має середню якість (рис. 3.18).

Як бачите, вебзастосунок розпізнав текст, але є помилки через якість зображення.

В результаті проведених тестувань виявлено, що для досягнення кращих результатів у процесі розпізнавання тексту зображення перед його завантаженням потрібно піддати покращенню. Це означає, що перед подачею зображення на алгоритм розпізнавання тексту необхідно виконати певні кроки

для поліпшення якості зображення і забезпечення кращої читабельності тексту.

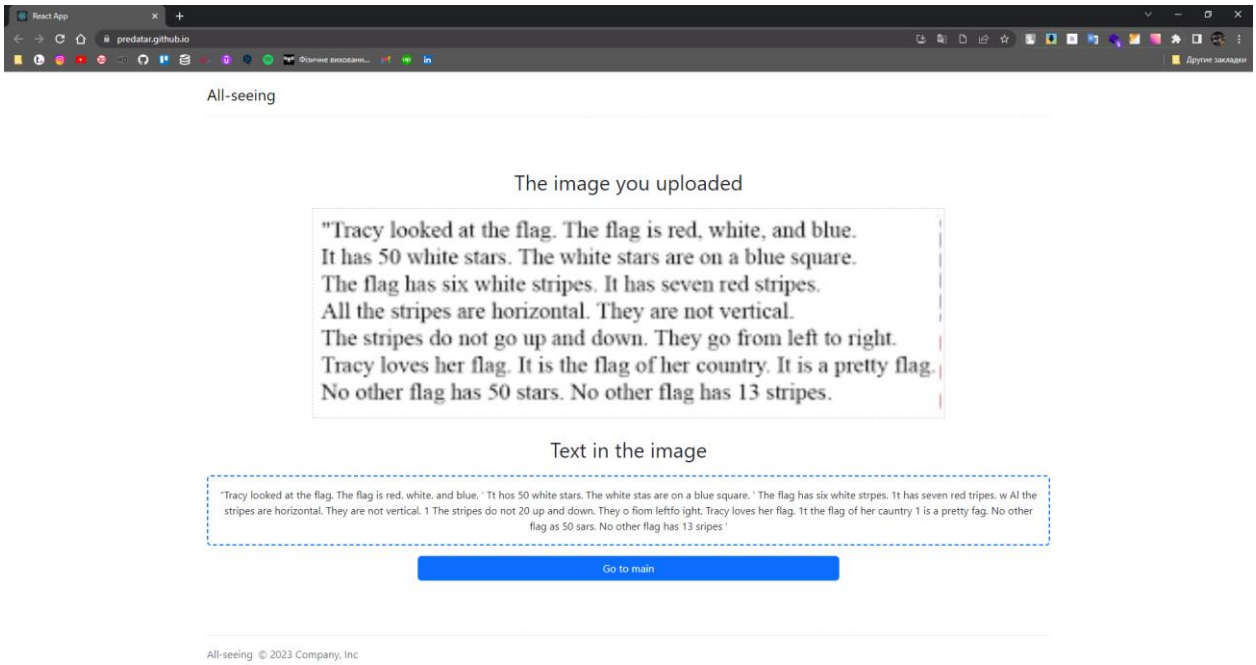

Рисунок 3.18 – Тестування №3. Зображення з середньою якістю

3.4.2 Тестування зображень з маленьким текстом

Часто можна зустріти ситуації, коли зображення містить невеликий текст, який розташований дуже щільно. Це може бути використано в різних контекстах, наприклад, на рекламних банерах, постерах, флаєрах, буклетах, упаковці продуктів або дизайні вебсайтів. Проведемо тестування на розпізнавання тексту такого зображення (рис. 3.19).

Після ретельного аналізу можна встановити, що зображення, про яке йдеться, має якість, яка може бути оцінена як помірна. Інформація на зображенні представлена у вигляді тексту, який займає значну площу зображення. Хоча вебзастосунок здійснив досить успішну роботу,

відтворивши текст з високою точністю, слід зазначити, що в результаті розпізнавання все ж таки виявлено невелику кількість помилок.

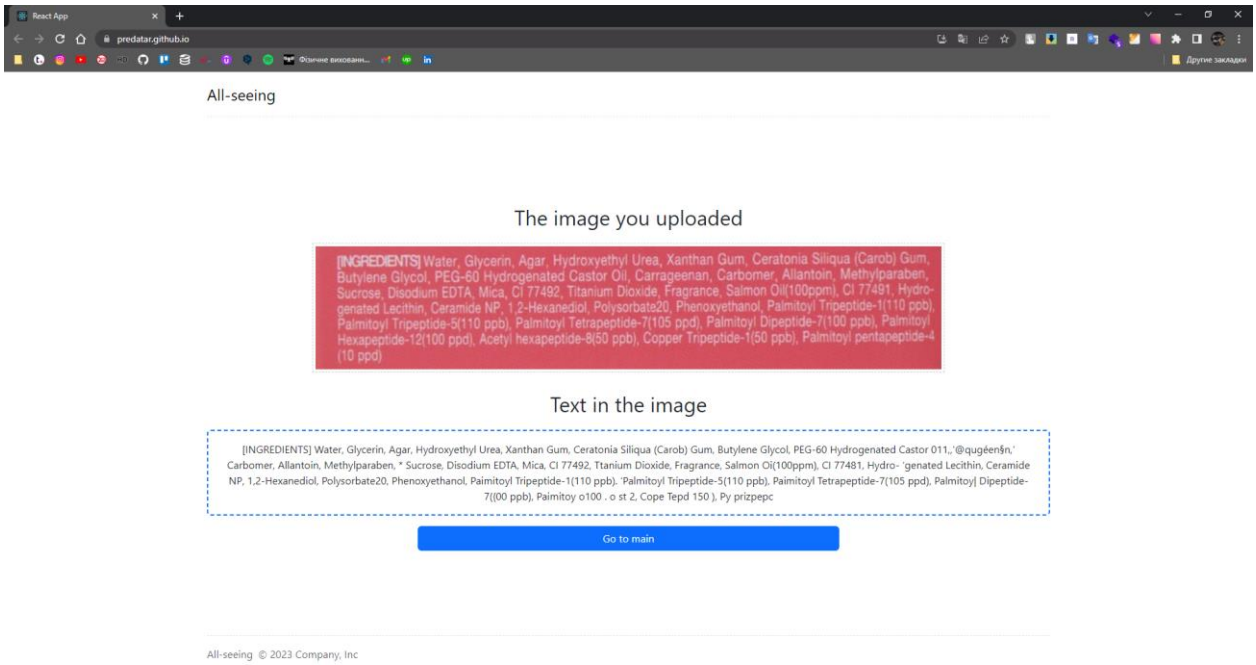

Рисунок 3.19 – Тестування №4. Зображення з маленьким текстом

Після проведення тестувань виявлено, що для досягнення кращих результатів у процесі розпізнавання тексту зображення потрібно покращити якість зображення і забезпечити кращу читабельність тексту перед його завантаженням для розпізнавання.

3.4.3 Тестування зображення з текстом під нахилом

Завдяки появі смартфонів фотографування стало невід’ємною частиною нашого повсякденного життя. У наші дні, доступна можливість фотографувати інформацію з використанням камери у будь-якому місці і в будь-який час. Проте, з використанням смартфонів, зручність нерідко призводить до ситуацій, коли зображення, яке було зроблено, має незначний нахил вліво або вправо. Проведено тестування такого зображення (рис. 3.20).

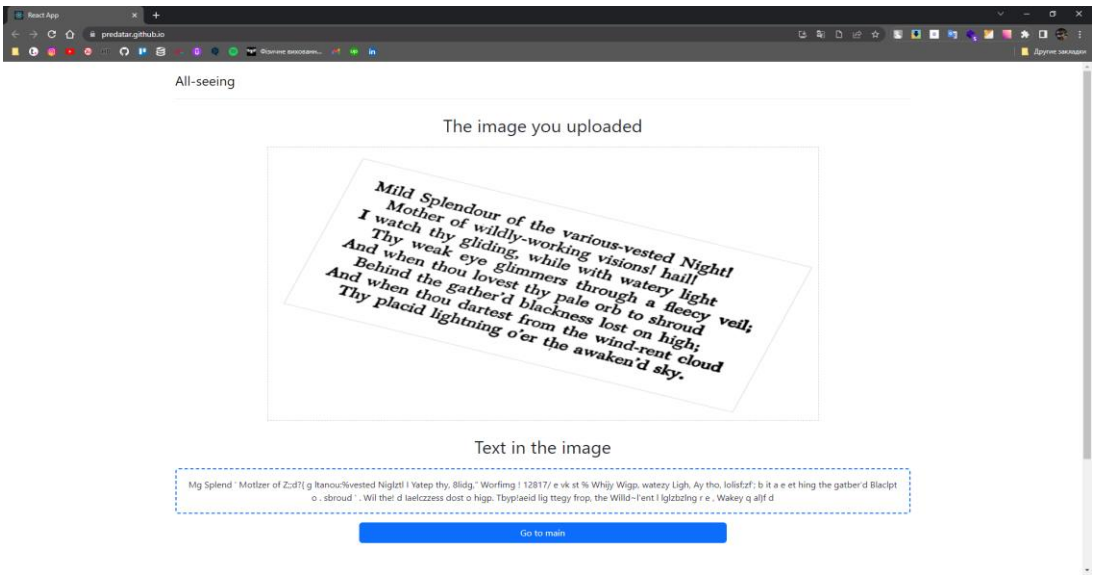

Рисунок 3.20 – Тестування №5. Зображення з текстом під нахилом

Як можемо побачити, таке зображення є найгіршим варіантом для розпізнавання тексту.

Для поліпшення розпізнавання тексту, необхідно провести певну підготовку зображення. Один з ключових кроків – це вирівняти зображення. Вирівнювання зображення означає розташування тексту в горизонтальному положенні, щоб забезпечити рівні основні лінії тексту.

3.5 Перспективи подальшої роботи

Під час подальшої роботи планується зосередитися на покращенні інтерфейсу користувача, щоб зробити вебзастосунок ще більш зручним і легким у використанні. Це може включати перегляд та оптимізацію розташування елементів на сторінках, удосконалення графічного дизайну та інші вдосконалення, спрямовані на поліпшення користувальницького досвіду [29].

Окрім того, планується додавання нових можливостей до вебзастосунку. Це можуть бути нові функції, які розширять функціональність і зроблять його ще більш потужним і корисним. Наприклад, це може включати можливість

перекладання тексту на обрану мову, вибір мови розпізнавання, підтримку інтеграції з іншими сервісами або платформами, розпізнавання тексту в реальному часі, або будь-які інші функції, які будуть відповідати потребам користувачів.

Під час розробки нових можливостей буде приділятися увага як функціональності, так і надійності та безпеки. Важливо забезпечити, щоб нові функції були добре інтегровані в існуючу систему і працювали стабільно і безпечно. Також будуть проводитися тестування та збір фідбеку від користувачів, щоб виявити можливі проблеми та вдосконалити функціонал згідно їх потреб [30].

Загалом, подальша робота над вебзастосунком спрямована на забезпечення найкращого користувальницького досвіду і задоволення потреб користувачів. Покращення інтерфейсу користувача та додавання нових можливостей дозволять розширити функціонал вебзастосунку і забезпечити більш широкий спектр використання. Процес подальшої роботи буде включати наступні етапи:

– дослідження та аналіз потреб користувачів. Буде проведений детальний аналіз поточного користувацького досвіду та зібрані фідбеки від користувачів для визначення пріоритетних областей покращення та нових можливостей. Це може включати опитування, спостереження за користувачами або аналіз використання даних;

– проєктування інтерфейсу користувача. За результатами дослідження будуть розроблені концепції та прототипи нового інтерфейсу користувача. Будуть враховані принципи зручності використання, візуальної привабливості та консистентності. Дизайн інтерфейсу буде протестований з метою виявлення можливих проблем та його оптимізації перед реалізацією;

– розробка нових можливостей. На основі виявлених потреб користувачів та враховуючи прототипи інтерфейсу, команда розробників приступить до реалізації нових функцій та можливостей. Це включатиме

програмування, тестування та впровадження нових функцій з метою забезпечення їх відповідності вимогам та стабільності роботи;

– тестування та вдосконалення. Розроблені нові можливості будуть піддані ретельному тестуванню для виявлення можливих помилок, недоліків або проблем в роботі. Зібрані фідбеки користувачів будуть враховуватися для вдосконалення та оптимізації нових функцій перед їх широкомасштабним впровадженням;

– збір фідбеку та оновлення. Після впровадження нових можливостей буде продовжуватися збір фідбеку від користувачів. Це дозволить виявити можливі проблеми, запити на поліпшення або додаткові функції. Зібраний фідбек буде аналізований та використовуваний для планування майбутніх оновлень і покращень вебзастосунку [31].

Загалом, подальша робота над вебзастосунком спрямована на постійне поліпшення користувальницького досвіду та розширення його функціональності. Команда розробників буде працювати в напрямку забезпечення зручного інтерфейсу користувача, додавання нових можливостей та підтримки високої надійності та безпеки. При цьому враховуватимуться потреби та фідбек користувачів, щоб забезпечити оптимальне використання вебзастосунку.

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений і реалізований вебзастосунок для розпізнавання тексту з зображень.

За допомогою сучасних методів розробки реалізовано вебзастосунок, який здатен розпізнавати текст на зображеннях. Основна мета проєкту полягала в забезпеченні високої точності розпізнавання тексту, тому для досягнення цього були використані готові бібліотеки та фреймворки, які забезпечують надійну та ефективну роботу з розпізнаванням тексту.

Вебзастосунок був підданий аналізу та тестуванню з використанням зображень різної якості та з різними позиціями тексту. Процес аналізу включав етап тестування, де були використані реальні вхідні дані, щоб оцінити ефективність розпізнавання тексту. У результаті цього аналізу були виявлені потенційні обмеження та питання, які було вирішено шляхом налагодження та вдосконалення алгоритмів та параметрів вебзастосунку. В цілому, завдяки проведеному аналізу та тестуванню, вебзастосунок з розпізнавання тексту був оптимізований для роботи з різними типами зображень та позиціями тексту

Проведено дослідження щодо можливостей вдосконалення вебзастосунку з метою подальшого поліпшення його функціоналу і користувацького досвіду.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Історія інтернету від ARPANET до сьогодні – UCloud. URL: https://ucloud.ua/istoriya-internetu-vid-arpanet-do-sogodni/ (дата звернення 12.04.2023).

2. Історія Інтернету — Вікіпедія. URL: https://uk.wikipedia.org/wiki/%D0%86%D1%81%D1%82%D0%BE%D1%80%D1%96%D1%8F_%D0%86%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82%D1%83 (дата звернення 12.04.2023).

3. Інтернет — Вікіпедія. URL: https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82 (дата звернення 13.04.2023).

4. How the web works - Learn web development | MDN. URL: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/How_the_Web_works (дата звернення 13.04.2023).

5. How does the Internet work? - Learn web development | MDN. URL: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/How_does_the_Internet_work (дата звернення 13.04.2023).

6. Структура сайту: основні типи та особливості їх проєктування – WebTune. URL: https://webtune.com.ua/statti/web-rozrobka/struktura-sajtu/ (дата звернення 13.04.2023).

7. Lyashenko, V., Lyubchenko, V., Mohammad, A., Alveera, K., & Kobylin, O. (2016). The methodology of image processing in the study of the properties of fiber as a reinforcing agent in polymer compositions.

8. Lyashenko, V., Babker, A., & Lyubchenko, V. (2017). Wavelet Analysis of Cytological Preparations Image in Different Color Systems.

9. Гороховатський, В. О., & Творошенко, І. С. (2021). Методи інтелектуального аналізу та оброблення даних: навч. посібник.

10. Архітектура вебзастосунків - як вибрати?: стаття з блогу IT-школи Hillel. URL: https://blog.ithillel.ua/articles/web-application-architecture (дата звернення 18.04.2023).

11. Мікросервісна архітектура. Причини використання, переваги та… | by Ivan Zmerzlyi | Medium. URL: https://medium.com/@IvanZmerzlyi/microservices-architecture-461687045b3d (дата звернення 18.04.2023).

12. Архітектура вебзастосунків – Telegraph. URL: https://telegra.ph/Arh%D1%96tektura-veb-dodatk%D1%96v-04-17 (дата звернення 19.04.2023).

13. Модель-вид-контролер — Вікіпедія. URL: https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C-%D0%B2%D0%B8%D0%B4-%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%B5%D1%80 (дата звернення 20.04.2023).

14. Model-View-Presenter — Вікіпедія. URL: https://uk.wikipedia.org/wiki/Model-View-Presenter (дата звернення 20.04.2023).

15. Model-View-ViewModel — Вікіпедія. URL: https://uk.wikipedia.org/wiki/Model-View-ViewModel (дата звернення 20.04.2023).

16. Shafronenko, A., Bodyanskiy, Y. V., Klymova, I., & Holovin, O. (2020, May). Online credibilistic fuzzy clustering of data using membership functions of special type. In CMIS (pp. 744-753).

17. Kobylin, O. A., Gorokhovatskyi, V. O., Tvoroshenko, I. S., & Peredrii, O. O. (2020). The application of non-parametric statistics methods in image classifiers based on structural description components. Telecommunications and Radio Engineering, 79(10).

18. Gorokhovatskyi, V., Gorokhovatskyi, O., Yevgenyi, P., & Olena, P. (2018, August). Quantization of the space of structural image features as a way to

increase recognition performance. In 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP) (pp. 464-467). IEEE.

19. Gorokhovatskyi, V. O., Tvoroshenko, I. S., & Peredrii, O. O. (2020). Image classification method modification based on model of logic processing of bit description weights vector. Telecommunications and Radio Engineering, 79(1).

20. Tesseract — Вікіпедія. URL: https://uk.wikipedia.org/wiki/Tesseract (дата звернення 21.04.2023).

21. Gorokhovatskyi, V. O., Tvoroshenko, I. S., & Vlasenko, N. V. (2020). Using fuzzy clustering in structural methods of image classification. Telecommunications and Radio Engineering, 79(9).

22. Tvoroshenko, I. S., & Gorokhovatskyi, V. O. (2020). Effective tuning of membership function parameters in fuzzy systems based on multi-valued interval logic. Telecommunications and Radio Engineering, 79(2).

23. Гороховатский, В. А. (2014). Структурный анализ и интеллектуальная обработка данных в компьютерном зрении.

24. Mashtalir, S., & Mikhnova, O. (2017). Detecting significant changes in image sequences. Multimedia Forensics and Security: Foundations, Innovations, and Applications, 161-191.

25. Ye, B., Tyshchenko, O., & Shafronenko, A. (2019). Fuzzy clustering of incomplete data by means of similarity measures-Proc. 2019 IEEE 2nd Ukr. Conf. on Electrical and Computer Engineering (UKRCON), Track 6. Lviv, Ukraine, 149-152.

26. Ibrahim, D. Y., Tvoroshenko, I. S., Gorokhovatskyi, V. O., Latiff, L. A., & Ahmad, N. (2021). Development of Effective Methods for Structural Image Recognition Using the Principles of Data Granulation and Apparatus of Fuzzy Logic.

27. Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., Gadetska, S., & Al-Dhaifallah, M. (2021). Methods of classification of images on the basis of the values of statistical distributions for the composition of structural description components. IEEE Access, 9, 92964-92973.

28. Bodyanskiy, Y. V., Tyshchenko, O. K., & Mashtalir, S. V. (2019). Fuzzy clustering high-dimensional data using information weighting. In Artificial Intelligence and Soft Computing: 18th International Conference, ICAISC 2019, Zakopane, Poland, June 16–20, 2019, Proceedings, Part I 18 (pp. 385-395). Springer International Publishing.

29. Shafronenko, A., Bodyanskiy, Y., Pliss, I., & Popov, S. (2020, September). Evolving neo-fuzzy system for distorted data online processing. In 2020 10th International Conference on Advanced Computer Information Technologies (ACIT) (pp. 352-355). IEEE.

30. Bodyanskiy, Y. V., Shafronenko, A., & Rudenko, D. (2019). Online Neuro Fuzzy Clustering of Data with Omissions and Outliers based on Completion Strategy. In CMIS (pp. 18-27).

31. Shafronenko, A., Bodyanskiy, Y., Pliss, I., & Irina, K. (2021, September). Online Credibilistic Fuzzy Clustering Method Based on Cauchy Density Distribution Function. In 2021 11th International Conference on Advanced Computer Information Technologies (ACIT) (pp. 704-707). IEEE.