Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна технологія пошуку інформації в текстах»

Виконав: студент 2-го курсу, групи 1КН-21м спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
Варнава В.Ю.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
Сілагін О.В.
(прізвище та ініціали)
«16» 12 2022 р.

Опонент: к.т.н., доцент каф. АІІТ
Іванов Ю.Ю.
(прізвище та ініціали)
«15» 12 2022 р.

Допущено до захисту
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
(прізвище та ініціали)
«16» 12 2022 р.

Вінниця ВНТУ - 2022 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації Кафедра комп'ютерних наук
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 122 «Комп'ютерні науки»
Освітньо-професійна програма – «Системи штучного інтелекту»

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
Д.т.н., проф. Яровий А.А.
14 вересня **2022 року**

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Варнава Владислав Юрійович
(прізвище, ім'я, по батькові)

1. Тема роботи: Розробка інформаційної технології пошуку інформації в текстах

керівник роботи к.т.н., доцент кафедри КН Сілагін О.В.
затверджена наказом вищого навчального закладу від 14.09.2022 р. №203

2. Строк подання студентом роботи 18 листопада 2022 року
3. Вихідні дані до роботи:
   Мова об'єктно-орієнтованого програмування, браузерне середовище роботи додатку, кількість критеріїв класифікації = 7 од., мова текстів, що класифікуються – англійська, кількість символів в тексті = 5000 , легкодоступність додатку за посиланням, відповідність стандарту ODBS.
4. Зміст текстової частини:
   Вступ, аналіз сучасного стану розвитку систем пошуку інформації в текстах, обґрунтування методу розв'язання задачі, обґрунтування вибору інструментів технічної реалізації, проектування інтелектуальної системи, програмна реалізація інтелектуальної системи, аналіз результатів тестування, економічна частина, розробка інструкції користувача, економічна частина, висновки, список використаних джерел, додатки.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень)
   Загальна структурна схема інформаційної технології, схема алгоритму роботи модуля збереження та аналізу тексту , діаграма послідовності, діаграма розгортування, загальний вигляд інтерфейсу користувача, приклад роботи програми

6. Консультанти розділів роботи

Розділ	Прізвище, ініціалита посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
1-3	Сілагін О.В., к.т.н., доц. каф. КН	14.09.22	11.22
4	Нікіфорова Л. О., к. е. н., доц. каф. ЕПВМ	19.09.22	12.11.22

7. Дата видачі завдання 14 вересня 2022 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	При-мітка
1	Аналіз сучасного стану розвитку систем пошуку інформації в текстах	14.05.22	01.10.22
2	Обґрунтування методу розв'язання задачі, обґрунтування вибору інструментів технічної реалізації	02.10.22	16.10.22
3	Проектування інформаційної технології. Програмна реалізація інформаційної технології. Аналіз результатів тестування	17.10.22	7.11.22
4	Підготовка економічної частини	08.11.22	11.11.2022
5	Апробація та/або впровадження результатів дослідження	23.11.22	01.12.22
6	Оформлення пояснювальної записки, графічного матеріалу та презентації	02.11.22	14.12.22

Студент _____ (підпис) Варнава В.Ю.

Керівник роботи _____ (підпис) Сілагін О.В.

АНОТАЦІЯ

УДК 004.8

Варнава В.Ю. Інформаційна технологія пошуку інформації в текстах. Магістерська кваліфікаційна робота зі спеціальності 122 – комп’ютерні науки, освітня програма - комп’ютерні науки. Вінниця: ВНТУ, 2022. 105 с.

На укр. мові. Бібліогр.: 21 назв; рис.: 18; табл. 7.

Дана магістерська кваліфікаційна робота присвячена розробці програмного забезпечення для вилучення інформації в тексту за запитом природньою мовою . Були розглянуті та проаналізовані існуючі програмні рішення і їх функціональні можливості, та обрано гібридний режим функціонування. Було проаналізовано різні підходи до вирішення задачі аналізу текстів. Було спроектовано програму пошуку інформації в текстах, написану мовою програмування JavaScript з використанням бібліотеки React для клієнтської частини, Node.js і Express для серверної частини та СУБД MongoDB для управління базами даних.

Графічна частина складається з 7 плакатів.

У економічному розділі розраховано суму витрат на розробку та виготовлення нового технічного рішення, яка складає _____ гривень, спрогнозовано орієнтовану величину витрат по кожній з статей витрат, розраховано чистий прибуток, термін окупності витрат для виробника ____ роки та економічний ефект для споживача при використанні даної розробки.

Ключові слова: класифікація тексту, природна мова, веб-клієнт, клієнт-серверна архітектура.

ABSTRACT

Varnava V.Г. Information technology for searching information in texts. Master's qualification thesis on specialty 122 - computer science, educational program - computer science. Vinnytsia: VNTU, 2022. 105 p.

In Ukrainian speech Bibliography: 21 titles; Fig.: 18; table 7.

This master's thesis is devoted to the development of software for extracting information from text on request in natural language. Existing software solutions and their functionality were considered and analyzed, and a hybrid mode of operation was chosen. Different approaches to solving the problem of text analysis were analyzed. A program for searching for information in texts was designed, written in the JavaScript programming language, using the React library for the client part, Node.js and Express for the server part, and the MongoDB DBMS for database management.

The graphic part consists of 7 posters.

In the economic section, the amount of costs for the development and production of a new technical solution is calculated, which is _____ hryvnias, the estimated amount of costs for each of the cost items is predicted, the net profit, the payback period for the manufacturer ____ years and the economic effect for the consumer when using this development are calculated.

Keywords: text classification, natural language, web client, client-server architecture.

ЗМІСТ

ВСТУП

Актуальність теми досліджень. В сучасному світі, створюються людством величезні обсяги інформації, більші ніж будь-коли і їх виробництво цих даних зростає щоденно. Щоб отримати з цих даних щось корисне необхідно їх проаналізувати і відфільтрувати. Зараз щомиті по всьому світу створюються гігабайти нових даних різного виду: наукові статті, пишуться сотні нових законів в різних країнах світу, сотні записів під постами у Facebook, десятки рецензій до художніх та документальних фільмів в стрімінгових сервісах, статті про економіку та політику. Більша частина цих не піддається читанню. Щоб витягти з них якусь користь, їх потрібно відфільтрувати і обробити [1].

Все це доводилося робити вручну, коли технології ще не були розвинені. На це витрачався годинник, дні, тижні і навіть місяці. Такі завдання є рутинними і потребують часу прийняття рішень. Крім того, у міру збільшення кількості коментарів для розгляду вам потрібно найняти більше співробітників. А з огляду на те, що раніше інформації для обробки було набагато менше, неважко зрозуміти, що вручну обробити такий обсяг було б неможливо. Тому було розроблено безліч алгоритмів, які дозволяють робити це за допомогою комп'ютерної техніки. Саме про такі методи, що стосуються пошуку інформації в текстах і піде мова в даній роботі. Так як дана система значно підвищує точність пошуку інформації робота є актуальною [2].

Зв'язок роботи з науковими програмами, планами, темами. Магістерська кваліфікаційна робота виконана відповідно до напряму наукових досліджень кафедри комп’ютерних наук Вінницького національного технічного університету 22 К1 «Моделі, методи, технології та пристрої інтелектуальних інформаційних систем управління, економіки, навчання та комунікацій» та плану наукової та навчально-методичної роботи кафедри.

Мета та завдання дослідження. є розширення функціональних можливостей в аналізі текстів за допомогою створення додатку для браузера, що дозволить покращити доступність системи.

Для досягнення поставленої цілі необхідно вирішити наступні поставлені **задачі**:

- провести аналіз сучасного стану розвитку систем пошуку інформації в текстах;
- Обґрунтування методу розв’язання задачі.
- Проектування інтелектуальної системи пошуку інформації в текстах.
- виконати Програмна реалізація інтелектуальної системи пошуку інформації в текстах.
- Виконати Тестування інтелектуальної системи аналізу пошуку інформації в текстах.
- Розробка інструкції користувача;
- Обрахувати економічну доцільність розробленого програмного засобу

Об’єктом дослідження є процеси пошуку інформації з текстах.

Предметом дослідження є алгоритми та програмне забезпечення, що організовує процес пошуку інформації з текстах.

Методи дослідження - методи аналізу тексту, теорія глибинних нейронних мереж, методи масштабування та інтеграції програмного забезпечення, методи та підходи до розробки систем штучного інтелекту, методи та підходи до розробки веб-орієнтованих програмних додатків, методи об’єктно-орієнтованого програмування.

Наукова новизна одержаних результатів полягає в наступному: Розроблено інтелектуальну систему аналізу текстів, що відрізняється від існуючих аналогів додатковими методами запиту, додатком для браузера та віддаленою базою даних.

Практичне значення одержаних результатів полягає у наступному:

- Розроблено алгоритм пошуку інформації в текстах
- На відміну від аналогів, які не мають зручного інтерфейсу, пропонується додати можливість вводу запитів за допомогою голосу та створення віддаленої бази користувачів. Це дозволяє використовувати вашу програму віддалено і з декількох пристроїв.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується строгістю постановки задач, коректним застосуванням математичних методів під час доведення наукових положень, строгим виведенням аналітичних співвідношень, порівнянням результатів з відомими.

Особистий внесок магістранта. Усі результати, наведені у магістерській кваліфікаційній роботі, отримані самостійно.

Апробація роботи.

Публікації: наукова конференція Молодь у Науці.

1 АНАЛІЗ СУЧАСНОГО СТАНУ РОЗВИТКУ СИСТЕМ ПОШУКУ ІНФОРМАЦІЇ В ТЕКСТАХ

1.1 Огляд проблем в області пошуку інформації в текстах

Такі технології стають дедалі актуальнішими, оскільки людям різних професій необхідно приймати рішення з урахуванням аналізу великих обсягів неструктурованого тексту. У ньому перераховані ряд завдань, що вимагаються користувачем, кожна з яких вимагає свого технічного рішення. У цілому нині набір цих завдань то, можливо дуже широкимі. До них слід віднести:

- класифікацію;
- кластеризацию;
- побудова семантичних мереж;
- витяг фактів, понять (feature extraction);
- витяг думок;
- анотування, суммарізація (summarization);
- відповідь на запит (question answering);
- тематичне індексування (thematic indexing);
- пошук за ключовими словами (keyword searching);
- створення таксономій і тезаурусів.

Вищенаведений список є далеко неповним, але вже відкриває широкий спектр проблем, які можна вирішувати.

Процес інформаційного пошуку являє собою зіставлення інформаційного запиту користувача з інформаційними ресурсами, що доступні системі, яка здійснює пошук. Ефективне виконання пошуку залежить як від засобів подання запиту, так і від засобів подання знань про інформаційні ресурси, а також від способів їхнього співставлення [3].

Основні критерії оцінки ефективності інформаційно-пошукових систем (ІПС) - це швидкість, точність і повнота відповідей. Точність визначається тим, яка частина інформації, що надається у відповідь на запит, є релевантною, тобто стосується цього запиту. Повнота характеризується співвідношенням між усією релевантною інформацією, що є в базі, і тією її частиною, що включена у відповідь. Крім цього, при оцінці пошукових систем враховується, з якими типами даних може працювати та або інша система, у якій формі представляються результати пошуку і який рівень підготовки користувачів необхідний для роботи в цій системі [3].

Найбільше розвинені можливості пошуку надають сьогодні системи пошуку за ключовими словами. Сучасні механізми пошуку в Web аналогічні за своєю дією традиційним системам здобуття інформації. Вони поділяються на дві групи - пошукові машини та каталоги [3].

Пошукові машини звичайно містять три компоненти:

- програму індексації інформаційних ресурсів (робота), що автоматично переглядає різні сайти й індексує їх,
- базу даних (індекс),
- програму сканування, що дозволяє за запитом знайти відповідні інформаційні ресурси.

При цьому кожна пошукова система намагається самостійно проіндексувати всю мережу. Чим більше вузлів покриває пошукова машина, тим вище частка помилкових посилань, що у деяких випадках може досягати навіть 10%. Деякі пошукові служби відносяться до повнотекстових - вони шукають ключові слова не тільки в заголовку (і в метатегах), але й у тілі сторінки. Інші обмежуються пошуком тільки в заголовках і метатегах. Те ж саме відноситься і до глибини дослідження вузлів: одні обробляють тільки заголовну сторінку, інші - усі посилання до певного рівня, треті - Web-вузол цілком. Крім того, деякі служби мають спеціалізацію (явну або неявну) і приділяють більше уваги вузлам, присвяченим певної темі. До пошукових машин відносяться AltaVista, HotBot, Google і Rambler [3].

Розвиток можливостей цих систем, спрямований на підвищення точності інформації, призводить до ускладнення мови запитів цих систем. Крім того, у кожній із систем є свій синтаксис мови запитів. Тому більшість користувачів просто ігнорують розвинені можливості систем пошуку і використовують тільки базові можливості, що призводять до низької якості результатів пошуку) [3].

Це призводить до того, що користувач змушений самостійно опрацьовувати (прочитувати й відсортовувати) велику кількість документів (причому більша частина яких йому не потрібна). Для постійної роботи користувача (як в Інтернет, так і на окремому комп'ютері або в локальній мережі) характерна довгострокова зацікавленість користувача в інформації з однієї або декількох вузьким областей. Тому доцільно надати користувачу персонального інформаційного агента, що, з одного боку, дозволило б автоматизувати задачу збору і накопичення тематичної інформації, з огляду на як специфіку цих областей (і формуючи відповідні бази знань), так і преваги конкретного користувача, а з іншого - підвищило б релевантність пошуку інформації в цих областях [3].

Механізми пошуку в Web, як правило, розглядають запити на пошук ізольовано один від одного. Результати, отримані у відповідь на даний запит, не залежать від користувача або контексту, у якому користувач створював запит. Часто вони пропонують застарілу інформацію, індексують лише частину доступної в Web інформації, не індексують документи, для доступу до яких необхідна аутентифікація, і тому багато документів залишаються за рамками пошуку. Крім того, різні сайти індексуються неоднаково [3].

Нові технології інформаційного пошуку враховують реакцію користувача на результати, отримані ними під час попередніх звертань до механізму пошуку, передбачають обробку запитів природною мовою, явне або автоматизоване додавання контекстної інформації тощо. Однак очевидно, що

універсального рішення, однаково зручного для всіх категорій користувачів, просто не існує [3].

Запит користувача являє собою опис того інформаційного ресурсу, доступ до якого хоче отримати користувач. Він може містити ключові слова, пов'язані логічними операторами; документ-зразок; тип документа (текстовий документ, зображення, відеоролик тощо); тему документа за класифікатором; списки рекомендованих або заборонених інформаційних джерел; обмеження часу або обсягу пошуку; параметри документа - обсяг, час створення, мова, автори, інші специфічні параметри даного типу документа, тип запиту - постійний або одноразовий [3].

Традиційні підходи до організації пошуку інформації можна розділити на три групи: методи індексного пошуку, статистичні методи і методи, засновані на базах знань [3].

Індексний пошук застосовується головним чином для роботи зі структурованими базами даних. У таких методах слова інтерпретуються як послідовності закодованих символів. Використовуючи формальний синтаксис мови запитів, система вибирає точну відповідність для окремих слів або словосполучень, що пов'язані логічними операторами. Застосування штучної мови запитів призводить до необхідності навчання користувачів. Такі системи не враховують різні форми і значення слів; користувачу непросто угадати точні слова і фрази, що були використані авторами в документах. Крім того, вони не можуть також впорядковувати документи за ступенем відповідності запиту, тому користувач змушений читати кожен документ, щоб визначити, наскільки він відповідає запиту [3].

Статистичні методи ґрунтуються на розрахунку різних частотних характеристик: частоти входження слова в документ, зваженої частоти входження і частоти спільного входження кількох слів. При цьому передбачається, що чим частіше зустрічається те або інше слово запиту в документі, тим у більшому ступені даний документ відповідає наданому

запиту. Основною одиницею інформації, якою оперують статистичні методи, є окреме слово, однак зв'язки між словами розглядаються винятково з математичної, а не з лінгвістичної точки зору. На відміну від методів бінарного пошуку, статистичні методи не вимагають застосування формальної мови запитів. Вони дозволяють проводити ранжирування документів за ступенем відповідності запиту, що істотно підвищує ефективність роботи з пошуковими системами. Однак такі методи не завжди дозволяють одержати бажані точність і повноту відповідей, оскільки важливість того або іншого терміна не завжди безпосередньо зв'язана з частотою його використання в документі [3].

Системи, що базуються на базі знань, використовують для пошуку інформації певні зовнішні знання (метазнання). Вони використовують концептуальні відносини, що не застосовуються при статистичному пошуку [3].

1.2 Аналіз існуючих програмних рішень

Проблему пошуку інформації та аналізу текстів вирішують велика кількість компаній як сервісних, так і таких, що розробляють програмне забезпечення. Розглянемо деякі з існуючих рішень вирішення проблеми аналізу текстів.

TextAnalyst розроблений як інструмент для аналізу змісту текстів, семантичного пошуку інформації, формування електронних архівів. У продукті реалізована синергія від використання технологій лінгвістичного аналізу і нейронних мереж [4].

Система TextAnalyst допоможе швидко резюмувати, ефективно управляти і об'єднувати в групи документи в текстовій базі. Вона полегшує пошук семантичної інформації або може сфокусувати вивчення тексту на якомусь певному предметі [4].

Система забезпечує вирішення таких завдань, як складання головної думки великого за об’ємом тексту, дає розуміння про текст, вирішує проблему ефективного здійснення навігації по великим текстовим документам і пошук інформації за допомогою запитів природною мовою. Але дана програма має малу чітуість роботи та не має засобів роботи з користувачем [4].

Головну сторінку TextAnalist наведено на рисунку 1.1.

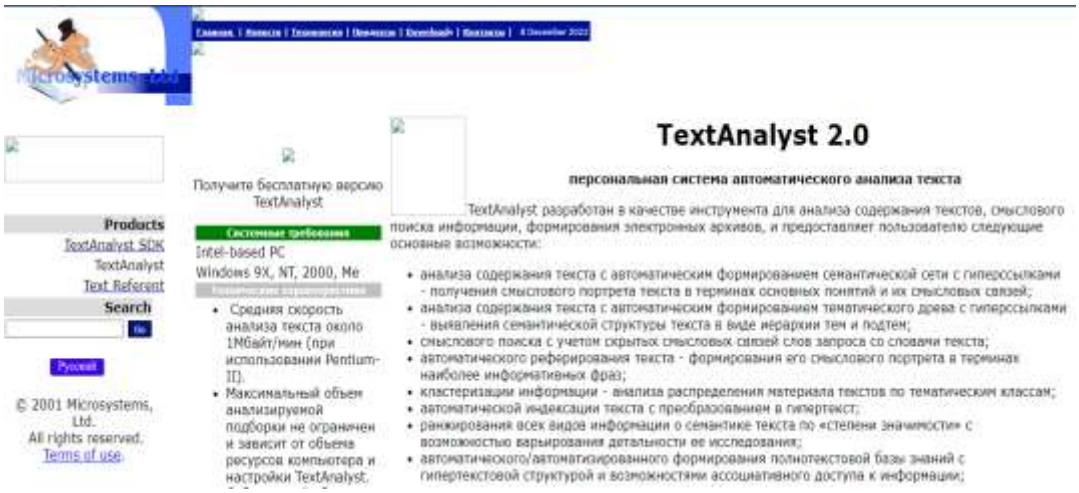


Рисунок 1.1 – Загальний вигляд головної сторінки TextAnalist

Аналіз тексту SAP BusinessObjects — це інструмент у стеку SAP BusinessObjects Data Services, який дає змогу перетворювати неструктуровані дані на структуровану інформацію, на основі якої можна керувати організацією. Інструмент SAP BusinessObjects Text Analysis допоможе вирішити питання щодо неструктурованої бізнес-аналітики.

Перетворення неструктурованих даних у структуровані дані дає величезну конкурентну перевагу тим компаніям, які бажають інвестувати в це. Аналіз тексту SAP BusinessObjects, інтегрований у стек продуктів SAP BusinessObjects EIM, дозволяє збагачувати дані про продукт даними про ваших конкурентів, знайденими в Інтернеті. У цій глобальній економіці вкрай важливо, щоб уся інформація була зафіксована, навіть якщо вона написана іншою мовою: тому аналіз тексту SAP BusinessObjects має кілька аналізаторів

тексту різними мовами, тому жодні неструктуровані дані, які ви хочете охопити, не виникнуть. [5].

Головну сторінку BusinessObjects Text Analysis наведено на рисунку 1.2.

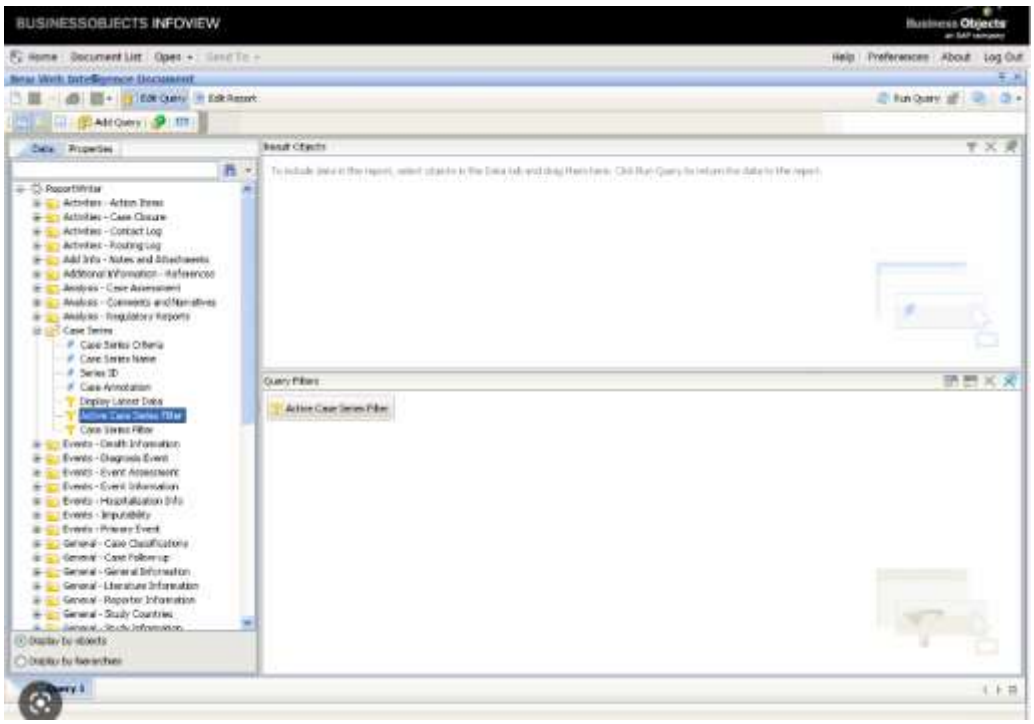


Рисунок 1.2 – Загальний вигляд головної сторінки BusinessObjects Text Analysis

Уніфікована платформа аналізу клієнтів Chattermill допомагає компаніям розкрити реальність своїх клієнтів. Використовуючи Chattermill, компанії можуть уніфікувати дані відгуків клієнтів у оглядах, заявках у службу підтримки, розмовах і соціальних мережах, щоб дізнатися, чого клієнти хочуть, потребують і очікують від їхніх продуктів і послуг.

Chattermill об’єднує відгуки клієнтів, підтримку клієнтів і відгуки про продукти в єдину платформу та використовує глибоке навчання штучного інтелекту (AI) для аналізу даних клієнтів у масштабі та надання корисної інформації. Ця програма дозволяє витягувати інформацію, що відноситься до конкретних об'єктів (людям, організаціям, географічним об'єктам і т. д.), ключовим фразам (конкретні часи, грошові суми) і т. д. Рішення також

аналізує відносини між суб'єктами, вирішує проблему множинних значень одного і того ж суб'єкта, виявляє відносини між суб'єктами, витягує події (хто, де, коли), витягує теми (суб'єкти, їх визначення) та визначає час. Місця подій, визначення місць, які можна прикріпити до карток. Перевагою є точність програми, але в нього немає достатньо зручного інтерфейсу та бази даних користувача. [6].

Головну сторінку Chattermill наведено на рисунку 1.3.

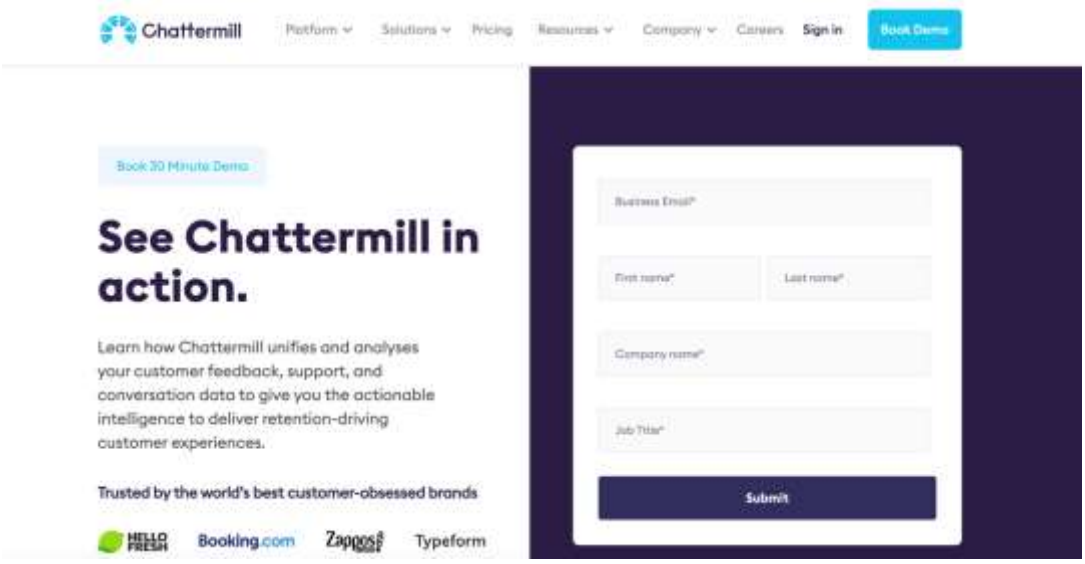


Рисунок 1.3 – Загальний вигляд головної сторінки Chattermill

Порівняльна характеристика наведених аналогів приведена у таблиці 1.1.

Таблиця 1.1 – Порівняльна аналіз характеристик існуючих аналогів інтелектуального модуля аналізу текстів на наявність образливих висловлювань

Характеристика	TextAnalyst	BusinessObjects Text Analysis	ChatterMill

Ресурсозатратність	+	-	+
Авторизація користувача	-	-	+
Можливість зберігання у базі даних	-	-	+
Точність	-	+	-
Зручний інтерфейс	-	-	-
Браузерний додаток	-	-	-

Тому в даному розділі визначили актуальність та неоднозначність проблеми контент-аналізу та окреслили основні проблеми існуючих рішень. Зокрема, проблеми у цій галузі — відсутність аутентифікації користувача, зберігання критичних для користувача даних та зручних інтерфейсів. Звідси можна сформувати основне завдання даної роботи – розробка технологій, які значно розширюють її можливості.

1.3 Постановка задачі і формулювання вимог до інтелектуальної системи пошуку інформації в текстах

Вхідними даними для програмного модуля пошуку інформації в текстах є: користувачі, їх тексти, кількість символів у тексті не більша за 500, кількість символів у запиті не більша за 200, мова - англійська.

Вихідними даними є зручний додаток та інформація, що була знайдена за запитом. Так як модуль матиме форму веб-додатка, він буде кросплатформеним і не залежатиме від конкретного апаратного забезпечення.

Отже створюваний програмний модуль має володіти рядом функціональних можливостей:

- містити весь функціонал у доповнені для браузера;
- запит за набором тексту;
- поле для завантаження тексту;
- голосовий запит;
- збереження результатів пошуку у базі даних;

Розроблене програмне забезпечення має функціонувати у всіх браузерах та пристроях. Усі наявні елементи повинні чітко працювати без будь-яких помилок.

Початком розробки інтелектуальної технології пошуку інформації в текстах є вибір методів аналізу тексту. Далі – розробка загальної структурної схеми додатку, розробка загальної схеми алгоритму роботи додатку, створення UML-діаграм розгортування та послідовності для деталізації складових подальшої розробки. Наступним етапом є вибір технологій реалізації та безпосередня розробка інтелектуального додатку. Заключними етапами є тестування розробленого інтелектуальної технологііі розробка інструкції користувача.

1.4 Висновок до розділу 1

В даному розділі було розглянуто проблему пошуку інформації в текстах, визначено основні труднощі даного процесу; проаналізовано існуючі рішення проблеми пошуку інформації в текстах; визначено задачі та вимоги до інтелектуальної технології пошуку інформації в текстах, що дозволяє перейти до етапу вибору методів аналізу та перетворення текстів для виконання інтелектуальної частини додатку.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ПОШУКУ ІНФОРМАЦІЇ В ТЕКСТАХ

2.1 Обгрунтування вибору методів пошуку інформації в текстах

Метод першого типу ґрунтується на технології розпізнавання іменованих сутностей. Іменований об'єкт - це група слів у тексті, що описує фактичний об'єкт. Наприклад, Apple Inc., John Broun, Information Extraction та ін. Пошук іменованих сутностей здійснюється в тексті з використанням шаблонів. Метод пошуку закономірностей можна розділити на основі правил і статистичний підходи. [1].

Методи, засновані на правилах знаходять паттерн в тексті, редуцируючи узагальнені правила. Наприклад, з правила є числом виходять більш специфічні правила є 4-значним числом або є дробовим числом. Для обчислення правил використовуються розмічена вручну навчальна база, тому для великих текстів даний підхід не застосовується [1].

Статистичні підходи (наприклад, система Nymble]) використовують варіації EM-алгоритму для знаходження розподілів токенів по сутностей. Зокрема, в Nymble використовуються приховані Марковскі моделі. Оскільки припущення, що токени смислу розподілені по нормальному закону в усьому тексті може бути неприйнятно в разі неструктурованого джерела, даний підхід нам також нецікавий [1].

Методи засновані на базах данних представлені підходом, описаним Matthew Michelson і Craig A. Knoblock і характеризуються використанням при аналізі змісту тексту якоїсь бази знань про об'єкти будь-якого типу. Текст, що аналізується автори підходу розглядають просто як набір токенів без будь-якої структури так званий текстовий пост. База знань представлена записами про об'єкти у вигляді сукупності пар {атрибут: значення} і в рамках статті називається безліч кандидатів. У своїй статті автори описують рішення

завдання розмітки токенів поста заданим набором міток-атрибутів і особливої мітки «junc», що символізує неналежність токена жодному з атрибутів об'єкта [1].

Задача вирішується в декілька етапів:

- З множини кандидатів за допомогою приблизного порівняння вибирається підмножина записів.
- По посту і кожного відібраного запису будується вектор ознак.
- Вектор ознак потрапить на вхід SVM-класифікатором, навченому відносити запис до одного з 2 класів: запис є схемою поста і запис не є схемою поста.
- Найкращий кандидат стає схемою посту.
- За схемою поста і кожному токені поста будується вектор ознак.
- Вектор ознак потрапить на вхід Multi-Class SVM класифікатора, навченому кожному токені поста ставити у відповідність мітку-атрибут.
- Для групи токенів одного атрибута проводиться чистка відсіваються токени з міткою junc.
- Залишається розмічений пост - необхідний результат роботи.

Даний підхід цікавий тим, що вирішує завдання вилучення інформації для повністю позбавленого структури джерела. Також для навчання класифікаторів не потрібно розмічених вручну даних.

2.2 Розробка алгоритму на основі Міхельсона

Автори підходу пропонують знаходити рішення за допомогою так званого безлічі кандидатів [2]. Кожен елемент цієї множини являє собою сукупність пар {атрибут: значення} і описує рівно один об’єкт з наявної бази знань.

У рамках вирішення поставленого завдання необхідно вирішити наступні підзадачі:

- Пошук схеми поста
- Вилучення інформації

Спершу необхідно з'ясувати структуру поста. Для цього пост порівнюється з елементами безлічі кандидатів. Для кожного кандидата будується вектор ознак, який потім подається вхід класифікатором, навченому відносити вектор до одного з двох класів: кандидат відповідає посаді і кандидат не відповідає посаді. Шуканої схемою поста є кандидат, віднесений до класу схем з найбільшою ймовірністю.

Ця задача вирішується у два кроки:

- Передобробка (побудова підмножини кандидатів)
- Зв’язування (подуба вектора і класифікація)

Розглянемо докладніше кожен з кроків.

Для цього безліч кандидатів кластеризуються за значеннями кількох конкретних атрибутів. В один кластер потрапляє група кандидатів з однаковими значеннями атрибутів, зазначених при кластеризації. Один кластер може бути описаний у вигляді правила, яким задовольняють всі елементи цього кластера. Правило зручно представляти у вигляді кон'юнкції пар {атрибут: значення}. Наприклад, нехай безліч кандидатів складається з 4 елементів:

{имя: "стол"}	{длина: 120}	{ширина: 80}	{материал: "дерево"}	[1]
{имя: "стол"}	{длина: 120}	{ширина: 60}	{материал: "стекло"}	[2]
{имя: "табурет"}	{длина: 70}	{ширина: 70}	{материал: "дерево"}	[3]
{имя: "табурет"}	{длина: 60}	{ширина: 60}	{материал: "сталь"}	[4]

Тоді правилом {ім'я: стіл} {довжина: 120} відповідають елементи 1 і 2, а правилом {ім'я: табурет} {матеріал: сталь} - 4 елемент.

Обробляти і класифікувати в подальшому має бути елементи одного обраного кластера. Вибраний кластер повинен відповідати 2 вимогам:

- Він повинен складатися з якомога меншої кількості елементів (щоб будувати менше векторів ознак)

- Він повинен мати достатньо елементів, щоб утримувати кандидатів, найбільш схожих на зміст посту - перспективних кандидатів

Для контролю вимог над правилом вводяться 2 заходи:

- ReductionRatio(RR) $= 1 - \mathrm{C}/N$, де C потужність кластера, описуваного правилом, а N потужність безлічі кандидатів
- PairCompleteness(PC) $= \frac{T_n}{S_n}$, де , $\mathrm{T_n}$ потужність безлічі перспективних кандидатів, що містяться в кластері, описуваному правилом, а $\mathrm{S_n}$ потужність безлічі перспективних кандидатів в множині всіх кандидатів.

Завдання знаходження підмножини кандидатів зводиться до знаходження правил, що описують систему кластерів, що містять всіх перспективних кандидатів. Для цього використовується sequentional covering algorythm, суть якого в наступному.

Для поста і кожного кандидата з підмножини, отриманого в ході попередньої обробки, будується вектор ознак $\mathrm{V_{rl}}$, що має наступну структуру:

$$
\begin{aligned}
V_{rl}(&nocm, кандидат) = (\\
&RL\_scores(nocm, amp_1)\\
&RL\_scores(nocm, amp_2)\\
&\dots\\
&RL\_scores(nocm, amp_1 amp_2 \dots amp_n)
\end{aligned}
$$

Вектор RL_scores ($\mathrm{str_1}$, $\mathrm{str_2}$) має наступну структуру:

$$
\begin{aligned}
RL\_scores(&str_1, str_2) = (\\
&JaccardSim(str_1, str_2)\\
&LevensteinDist(str_1, str_2)\\
&JaroWinklerSim(str_1, str_2)\\
&SmithWatermanSim(str_1, str_2)\\
&Soundex(str_1, str_2)\\
&PorterStemmer(str_1, str_2)
\end{aligned}
$$

Для отриманих векторів проводиться процедура binary rescoring, яка полягає у приведенні векторів дійсних чисел до двійковим векторах. Правило приведення таке: компонента вектора стає 1, якщо її значення максимально серед значень цієї ж компоненти всіх інших векторів, і 0, якщо інакше. Якщо

кілька векторів мають максимальним значенням будь-якої компоненти, то у всіх цих векторів компонента в процесі binary rescoring стане рівною 1. Наприклад, такі вектори:

```
( 0.21 0.0 0.5 -0.2 2.0 )

( 0.5 0.0 0.5 -0.3 2.0 )

( -0.12 0.3 0.3 -0.7 2.0 )
```

Після binary rescoring виглядають так:

```
( 0 0 1 1 1 )

( 1 0 1 0 0 )

( 0 1 0 0 1 )
```

Потім вектори передаються класифікатору SVM, навченому відносити вектори одного з двох класів. Кандидат відповідає посаді, кандидат відповідає посаді. Серед усіх кандидатів, які віднесені до класу, кандидат відповідає посаді, з більшою ймовірністю буде обрано єдиного кандидата, віднесеного до цього класу класифікатором. Вибраний кандидат є схемою посту.На кроці вилучення інформації для знайденої схеми і кожного токена поста будується вектор ознак V_{ie}, що має наступну структуру:

V_{ie}(токен, схема) = (
IE_scores(токен, схема$_1$)
IE_scores(токен, схема$_2$)
…
IE_scores(токен, атр$_n$)

Вектор IE_scores (str_1, str_2) має наступну структуру:

$$RL\_scores(str_1, str_2) = ($$
$$LevensteinDist(str_1, str_2)$$
$$JaroWinklerSim(str_1, str_2)$$
$$SmithWatermanSim(str_1, str_2)$$
$$Soundex(str_1, str_2)$$
$$PorterStemmer(str_1, str_2)$$

Далі вектори подаються на вхід класифікатором Multi-Class SVM, навченому кожному токені поста ставити у відповідність мітку того атрибуту, до якого найбільш ймовірно токен належить. Після цього для групи токенів, віднесених до одного і того ж атрибуту, проводиться процедура чистки, а саме зміни мітки токена, який насправді не є значенням обраного атрибута, на пустушку. Пустушками в пості є токени, які не належать значенням жодного з атрибутів [1]. Наприклад, в оголошенні про продаж товару це можуть бути токени cheap, for sale, used і т. Д. Суть чистки наступна:

- З групи токенів формується значення атрибута в пості (наприклад, строкаtok1 tok2...tokn)
- Для отриманого рядка і значення співвіднесеного атрибута зі схеми поста обчислюється 2 заходи: JaccardSim і JaroWinklerSim. Отримані значення вважаються базовими
- Потім по черзі з рядка забирається по одному токен, і для отриманого рядка і значення атрибута заново обчислюються заходи
- Якщо значення обох заходів зросли в порівнянні з базовими, то отримані значення самі стають базовими, а прибраний токен стає кандидатом на видалення (заміщаючи попереднього кандидата, якщо він був)
- Після перевірки всіх токенів кандидату на видалення ставиться мітка пустушка
- Процес повторюється, поки в при перевірці токенів вдається знайти хоча б одного кандидата на видалення
- Коли кандидатів на видалення більше немає, перевіряються поточні значення заходів. Якщо вони менші від необхідних мінімумів (для

кожної заходи свій), то всім токені атрибута ставиться мітка пустушка. Після очищення атрибутів залишається пост, кожен токен якого позначений міткою-атрибутом або міткою-пустушкою - необхідний в завданні результат.

2.3 Обґрунтування вибору платформи для розробки інформаційної технології пошуку інформації в текстах

Зазвичай платформа відноситься до операційної системи, як Windows або MacOS та комп'ютерного обладнання. Платформа є набором правил, стандартів, які дозволяють програмістам розробляти програмні застосунки використовуючи найкращий набір технологій. Ці стандарти дозволяють бізнесу і менеджерам придбати відповідні програмні застосунки та обладнання.

Веб-додаток — це програма, яку можна переглядати в Інтернеті. На перший погляд вони виглядають так само, як і звичайні сайти, але є величезна різниця. Два види популярних веб-застосунків. Це SPA (односторінкова програма), SSR (рендер стрінки здійснюється на сервері), PWA (прогресивний веб-додаток) [7].

PWA (Progressive Web App) - це технологія у веб-розробці, з її допомогою можна побудувати сайт візуально і функціонально нагадує мобільний додаток, тільки він відображатиметься в браузері [7].

PWA сайти будуються за допомогою HTML, JavaScript, CSS, і в браузері виглядають як звичайний сайт, але взаємодіють із відвідувачем як мобільний додаток. Розширений функціонал дозволяє додавати сайт або окрему сторінку на екран будь-яких пристроїв, а також відправляти push-сповіщення [7].

PWA дає можливість сайту-додатку працювати в автономному режимі. Наочним прикладом служить Google Docs - це онлайн-офіс, але з можливістю взаємодії з ним в режимі офлайн.

Для розробки веб-сайту можуть бути використані готові шаблонні рішення з коробки (CMS) або користувальницькі рішення [7].

CMS, а саме налаштування готового шаблону, - це швидкий спосіб розробки вашого продукту, так як бібліотека, що використовується розробником, вже містить великий список стандартних елементів [8].

Основні переваги та недоліки використання CMS показані в таблиці 2.1.

Таблиця 2.1 – Переваги та недоліки CMS

Переваги CMS	Недоліки CMS
можна створити сайт самостійно та за короткий проміжок часу;	потрібно стежити за оновленнями та сумісністю нових версій із доповненнями
не потрібно розбиратися в дизайні та програмуванні;	поганий рівень безпеки
розробка сайту коштуватиме дешевше;	Погано працює з пошуковими системами
Зручно користуватись вмістом сайту.	Обмеження кастомізації сайтів, всі дизайни можуть буди занадто схожими один на одного
	продуктивність зазвичай знижується, якщо на сайті є багато доповнень; не підходять для нетипових завдань

Server-side rendering (SSR) є популярною технологією для рендеру a client-side single page application (SPA) на сервері і ось повідомленні, що повністю rendered page to the client. Це дозволяє для динамічних компонентів, щоб служити як статичне HTML маркування [7].

SSR додатки розбивають JavaScript і CSS на шматки, оптимізовують файли, і рендкрять сторінки на сервері до служби клієнта браузера, що ведеться в фастері початкового часу [7].

Основні переваги та недоліки розробки користувальницького веб-сайту занесено в таблицю 2.2.

Таблиця 2.2 – Переваги та недоліки розробки користувальницького веб-сайту

Переваги	Недоліки
Безпека данних	Витрачається більше часу на розробку у порівнянні з CMS, а тому для реалізації проекту потрібно витратити більший бюджет.
Високий рівень кастомізації інтерфейсу	
можливість до власного SEO;	
сайт легко збільшувати і додавати новий функціонал	

На основі наведених вище недоліків і плюсів було обрано для розробки користувальницький веб-сайт. В перспективі система додатково буде розроблена CMS-система, яка буде пришвидшувати процес розробки веб-сайтів шляхом створення початкових шаблонів.

2.4 Обґрунтування вибору архітектури проектування серверної частини інформаційної технології пошуку інформації в текстах

Існує деклька основних правил до побудови стабільних бекенд додатків: атомарні додатки і додатки зроблені з невеликих мікросервісів. Монолітний додаток розробляється як єдиний уніфікований блок, тоді як архітектура мікросервісів - це сукупність менших, незалежно розгорнутих служб [11].

Термін «моноліт» походить від стародавнього зображення величезної скелі. Коли ми говоримо про програмне забезпечення, моноліти - це не що

інше, як великий блок кодів, що має кілька модулів. Ці модулі щільно з'єднані один з одним. Програма та бізнес-логіка інкапсульовані в єдиному двійковому файлі, який можна розгорнути, який називається монолітом. Зазвичай моноліт складається зі звичайної трирівневої архітектури, а саме бази даних, інтерфейсу користувача та програми на стороні сервера.

Внесення змін до такої програми вимагає оновлення всього стека з використанням доступу до кодової бази для створення та розгортання оновленої версії інтерфейсу на стороні служби. Це обмежує оновлення та займає більше часу [11].

Монолітні програми існували споконвіку. Численні інструменти дійсно полегшують розробку та розгортання стратегій. Розробникам потрібно виконати єдиний фрагмент коду, який можна розгорнути, замість того, щоб робити оновлення в окремих сутностях. Структура монолітної архітектури наведена на рис. 2.1.

Основні переваги та недоліки використання монолітної архітектури занесено в таблицю 2.3.

Рисунок 2.1 – Загальна структура монолітної архітектури

Таблиця 2.3 – Переваги та недоліки монолітної архітектури

Переваги монолітної архітектури	Недоліки монолітної архітектури
Виконуваний файл або каталог полегшує **розгортання.**	**Менша швидкість розробки** - Велике монолітне застосування робить розробку більш складною та повільною.
Розробка – Коли програма побудована з однією кодовою базою, її легше розробити.	**Масштабованість** – Ви не можете масштабувати окремі компоненти.
Централізовані бази коду та репозиторії часто дозволяють одному API виконувати ту саму функцію, що й кілька API у мікросервісах.	Надійність, помилка в одному модулі може вплинути на доступність всієї програми.
Спрощений тест. Монолітне додаток є єдиною централізованою одиницею, тому наскрізне тестування може виконуватися швидше, ніж розподілене додаток.	**Бар'єр для впровадження технологій** - Зміни у фреймворках та мовах впливають на всі програми, тому зміни часто є дорогими та трудомісткими..
Легке налагодження – З усім кодом, розташованим в одному місці, легше виконати запит і знайти проблему.	**Відсутність гнучкості** - Моноліт стримується технологіями, які вже використовуються в моноліті.
	Розгортання - Невелика зміна монолітного додатка вимагає передислокації всього моноліту.

Архітектура мікросервісів (також відома як мікросервіси) - це метод архітектури, заснований на наборі незалежних сервісів, що розгортаються. Ці

послуги мають свою бізнес-логіку та базу даних з певною метою. Оновлення, тестування, розгортання та масштабування виконуються для всіх сервісів. Мікросервіси об'єднують великі бізнес-завдання, пов'язані з предметною областю, окремі незалежні кодові бази. Мікросервіси широко застосовуються багатьма технологічними гігантами. Реалізація мікросервісів залежить від випадку до випадку [34].

За допомогою мікросервісів окремі члени команди можуть працювати над окремими модулями. Кожна особа може створити модуль і розгорнути модуль самостійно, тим самим зменшуючи операційне тертя команди та підвищуючи гнучкість доставки програмного забезпечення. Структура монолітної архітектури наведена на рис. 2.2.

Основні переваги та недоліки використання монолітної архітектури занесено в таблицю 2.4.

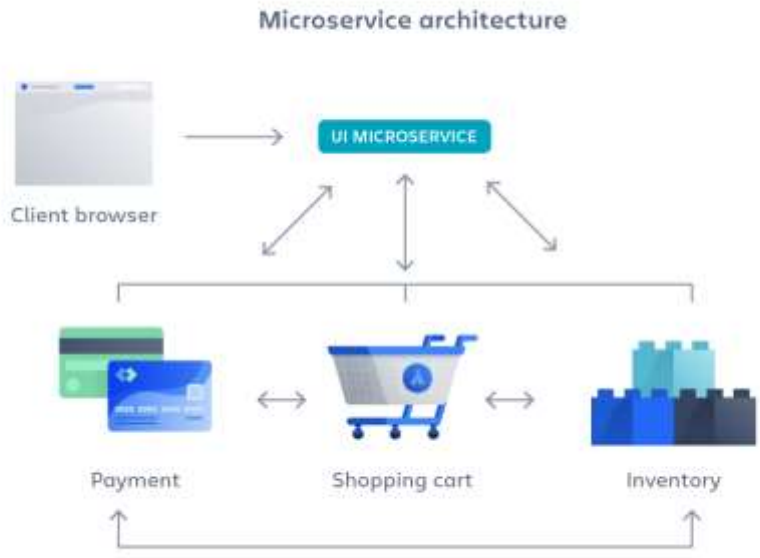


Рисунок 2.2 – Загальна структура мікросервісної архітектури

Таблиця 2.4 – Переваги та недоліки монолітної архітектури

Переваги	Недоліки
Гнучкі способи роботи з невеликими командами, які часто розгортають додатки.	**Розподілення розробки** – Мікросервіси складніші в порівнянні з монолітними архітектурами, тому що існує більше сервісів у більшій кількості місць, створених кількома командами. Розповсюдження розробки, що погано керується, призводить до повільної розробки і низької продуктивності.
Гнучке масштабування – Коли мікросервіс сягає меж своїх можливостей, нові екземпляри цього сервісу можна швидко розгорнути в супутніх кластерах, щоб знизити навантаження. Це дозволяє одночасно працювати з великою кількістю клієнтів, розподілених на різні сервери, і підтримувати сервери більшого розміру.	**Експоненціальні витрати на інфраструктуру** – Кожен новий мікросервіс може мати свої витрати на набори тестів, плани розгортання, інфраструктуру хостингу, інструменти моніторингу тощо.
Безперервне розгортання – Використання мікросервісів дозволяє пришвидшити цикл випуску.	**Додаткові організаційні накладні витрати** – Командам потрібно додати ще один рівень комунікації та співпраці для координації оновлень та інтерфейсів.

Продовження табл. 2.4

Легкість тестувати та підтримки – Команди можуть спробувати нові функції та повернутися, якщо щось не працює. Це спрощує оновлення коду та прискорює виведення нових функцій на ринок. Крім того, збої та помилки в окремих сервісах легко виявляються та усуваються.	Проблеми налагодження. Кожна мікрослужба має власний набір журналів, що ускладнює налагодження. Крім того, один бізнес-процес може виконуватися на кількох комп'ютерах, що ще більше ускладнює налагодження.
Незалежна розгортка – Оскільки мікросервіси є окремими додадтками, вони дозволяють швидко і легко самостійно розгортати індивідуальні функції.	**Відсутність стандартизації** - Без спільної платформи може спостерігатися розділення мов, стандартів реєстрації та моніторингу.
Гнучкість технологій – Архітектура мікросервісів дає командам свободу вибору інструментів, які вони бажають.	**Відсутність чіткої власності** - У міру появи нових сервісів зростає і кількість команд, які керують цими сервісами. Згодом стає складно зрозуміти, які послуги доступні вашій команді та до кого звернутися за підтримкою.
Висока надійність – Зміни можуть бути розгорнуті в певних службах без ризику збою всієї програми.	

Порівняння схем побудови монолітної та мікросервісної архітектури наведено на рис. 2.3.

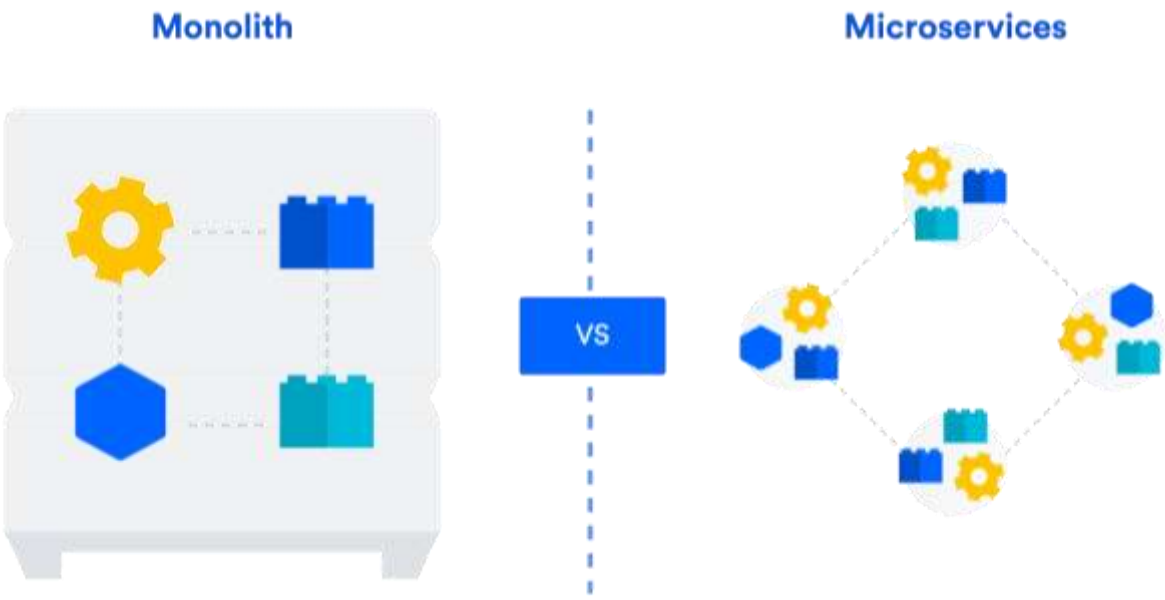


Рисунок 2.3 – Порівняння монолітної та мікросервісної архітекутур

Для інформаційної технології пошуку інформації було обрано монолітний підхід так як простіший у побудові, тестуванні та розгортуванні.

2.5 Обґрунтування вибору архітектури інформаційної технології пошуку інформації в тексті

Комплексна архітектура - це підхід до побудови надійної архітектури з використанням найкращих у своєму роді API. Складові архітектури забезпечують достатній досвід користувача, без усієї роботи та обслуговування системи або монолітного рішення.

Команди зі складною архітектурою можуть вибирати API, які найкраще підходять для їхнього варіанта використання, розміру команди та бюджету, та об'єднувати їх разом, щоб надати їм усі необхідні функції. Вам не потрібно платити за функції, які вам потрібні. Послуги можна легко підключати або відключати в міру зміни цих потреб, звільняючи вашу команду від дорогої прив'язки до постачальника в минулому.

Команди отримують вигоду від безперервної доставки функцій та обслуговування, характерних для продуктів SaaS, і їм не потрібно турбуватися про власні виправлення або довгострокове обслуговування критично важливого програмного забезпечення. Натомість ви можете використовувати наш гнучкий, надійний та простий сервіс для створення продуктів, які ви хочете створити. Щоб створити ефективну складову архітектуру, команди повинні використовувати основний концентратор контенту, який керує потоком даних між системами, а потім підключати інші найкращі у своєму класі служби, що відповідають їхньому варіанту використання.

Однак вони дорогі і негнучкі, і команди часто платять за функції, які їм не потрібні чи не потрібні. Коли команди створюють універсальні інструменти, які можуть робити все, часто відображають цифрові продукти минулого, а не майбутнього. [12].

На відміну від складних архітектур монолітні системи є потужними інструментами. Комплексна архітектура дозволяє командам вибирати спеціалізовані інструменти, що відповідають кращим практикам для конкретних випадків використання. Команди можуть створювати робочі процеси, які вони хочуть, використовуючи ті функції, які їм потрібні, без додаткової оплати. Після того, як ви вибрали основні служби та завершили початкове налаштування, вашій команді не доведеться турбуватися про їх обслуговування. Команди можуть почати створювати сучасний цифровий продукт, який вони уявляють, використовуючи інструменти, призначені для цієї мети. Монолітні системи часто відображають світ у веб-уявленнях. Сервіси на основі API передають останні знання про цифрові продукти інтерфейсу, від мобільних додатків до смарт-годин та смарт-помічників. [12].

Кілька аспектів підпадають під критерій «вартість». Вартість початку роботи, вартість обслуговування, вартість розробки, вартість якості, вартість швидкості та продуктивності та вартість володіння. Вартість є суттєвим фактором, який спадає на думку керівникам, коли вони приймають остаточне рішення про прийняття будь-якої архітектури програмного забезпечення. [12].

Що стосується надійності, то мікросервіси також має перевагу. Моноліти — це не що інше, як велика частина двійкових файлів програми. Випадково, якщо розгортання не вдається, вся програма вийде з ладу. У порівнянні з монолітами мікросервіси дуже надійні. Навіть у випадках збою служби додаток не вийде з ладу в цілому. Існує кілька інструментів виявлення служб, наприклад Hashicorp Consul, який перевіряє серцебиття кожної служби. На високому рівні надійність вище в програмах, що працюють на мікросервісах. Моноліти можна масштабувати різними способами. Одним із них є використання багатьох віртуальних машин, а потім маршрутизація запиту за допомогою балансувальника навантаження. Архітектура мікросервісу більш детальна, а отже, масштабування кожного мікросервісу є більш детальним і гнучким. Масштабованість є контрастним фактором будь-якої корпоративної програми. [12].

Отже, на ринку доступно кілька методів, які гарантують, що мікросервіси масштабуються як горизонтально, так і вертикально. Серед популярних інструментів, доступних на ринку, є Amazon EKS, Amazon ECS, Docker і Kubernetes. Для точного масштабування та кращого використання ресурсів мікросервіси є очевидно кращим вибором[12].

SPA (односторінкова програма) — це реалізація веб-програми, яка завантажує лише один веб-документ, а потім оновлює основний вміст цього єдиного документа за допомогою API JavaScript, таких як XMLHttpRequest і Fetch, коли має бути показано інший вміст. Таким чином, це дозволяє користувачам використовувати веб-сайти без завантаження цілих нових сторінок із сервера, що може призвести до збільшення продуктивності та більш динамічного досвіду, з деякими компромісними недоліками, такими як SEO, більше зусиль, необхідних для підтримки стану, реалізації навігації та досягнення значущої продуктивності моніторинг. [12].

Односторінкова програма зазвичай покладається на маршрутизатор. Маршрутизатори складаються з маршрутів, які описують розташування, якому вони повинні відповідати. Це можуть бути статичні (/about) або

динамічні (/album/:id, де значення :id може мати будь-яку кількість варіантів) шляхи. Після підбору маршруту маршрутизатор запустить повторну візуалізацію програми. Реальна реалізація цього в основному залежить від маршрутизатора. Наприклад, маршрутизатор може використовувати шаблон спостерігача, де ви надаєте маршрутизатору функцію, яка знає, як ініціювати повторне відтворення, і маршрутизатор викличе її після того, як вона збігається з маршрутом. [12].

Основні переваги та недоліки використання SPA занесено в таблицю 2.3.

Таблиця 2.5 – Переваги та недоліки SPA

Переваги	Недоліки
Початкове завантаження може тривати деякий час. Після завантаження програми додаткові завантаження не потрібні.	У міру збільшення розміру та складності вашої програми час початкового завантаження може серйозно знизитись, що призведе до погіршення взаємодії з користувачем.
Добре підходить для динамічної взаємодії, коли ваша команда потребує індивідуального підходу до роботи з користувачем.	Час завантаження та відсутність початкового HTML-контенту роблять майже неможливим підтримання хорошого SEO.
Команди краще контролюють архітектуру та можуть використовувати сучасні веб-фреймворки.	Великі файли для складних веб-додатків можуть стати складними в обслуговуванні та організації
Може використовуватися в тандемі з іншими технологіями	Проблеми зі SPA-підходом вимагають обхідних шляхів, які можуть бути дорогими та трудомісткими

Для інформаційної технології пошуку інформіції в текстах було обрано SPA підхід так як він дає користувачу кращу динаміку у використанні порівняно з іншими підходами.

2.6 Проектування інформаційної технології пошуку інформації в текстах

Модель – це абстракція, яка створюється з метою осмислення чого завгодно перед тим, як його створювати [9].

Абстрагування – це вибіркове вивчення деяких аспектів проблеми. Основна мета абстрагування полягає в тому, щоб ізолювати аспекти, важливі для деякої цілі, і відкинути всі інші [9].

Уніфікована мова моделювання (UML) була створена для створення спільної, семантично та синтаксично багатої мови візуального моделювання для архітектури, дизайну та реалізації складних програмних систем як структурно, так і поведінково. UML має додатки, окрім розробки програмного забезпечення, такі як потік процесів у виробництві [15].

Він аналогічний кресленням, які використовуються в інших областях, і складається з різних типів діаграм. У сукупності діаграми UML описують межі, структуру та поведінку системи та об’єктів у ній. Діаграма активності є аналогом звичної схеми алгоритму програми, тобто відображає, як потік управління переходить з одної діяльності в іншу [15].

Діаграма послідовності - UML-діаграма, на якій для деякого набору об'єктів на єдиній часовій осі показаний життєвий цикл об'єкта (створення-діяльність-знищення певної сутності) та взаємодія акторів (діючих осіб) інформаційної системи в рамках прецеденту [15].

Діаграми послідовностей зазвичай містять об'єкти, які взаємодіють у рамках сценарію, повідомлення, якими вони обмінюються, і результати, пов'язані з повідомленнями, що повертаються. Втім, результати, що часто повертаються, позначають лише в тому випадку, якщо це не очевидно з контексту. Об'єкти позначаються прямокутниками з підкресленими іменами (щоб відрізнити їхню відмінність від класів) [15].

Таким чином, отримаємо 5 об’єктів: веб-клієнт, програмний інтерфейс, система вилучення інформації, база даних, користувач. Розроблена діаграма послідовностей зображена на рисунку 2.4.

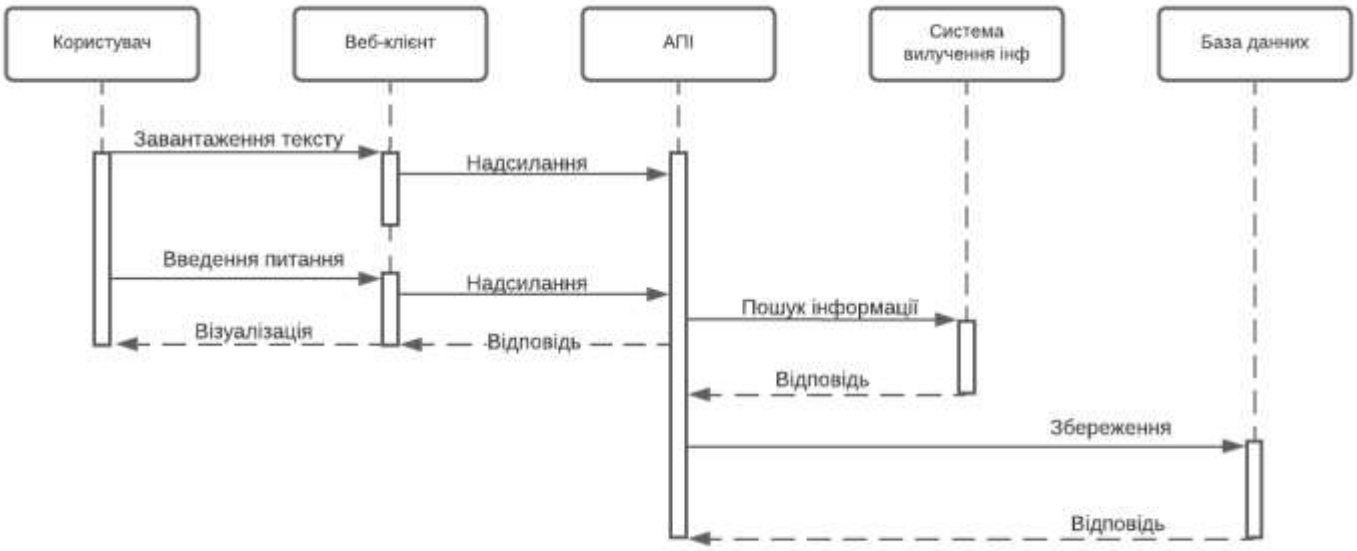


Рисунок 2.4 – Діаграма послідовності інтелектуальної технології аналізу пошуку інформації в текстах

Для розробки фізичної та логічної організації модулів інтелектуальної технології пошуку інформації в текстах було використано діаграму розгортування. Діаграма розгортання — це тип діаграми UML, яка показує виконавчу архітектуру системи, включаючи такі вузли, як апаратні або програмні середовища виконання та проміжне програмне забезпечення, яке їх з'єднує. Схема розгортання технології інтелектуальної текстової аналітики включає такі вузли:

- сервер;
- база даних;
- веб-клієнт;

- хостинг бази даних;
- СУБД;
- програмний інтерфейс;
- система пошуку інформації в тестах

Усі компоненти пов’язані між собою мережею, тобто зв’язком на основі протоколів TCP/IP. Розроблена діаграма розгортування представлена на рисунку 2.5.

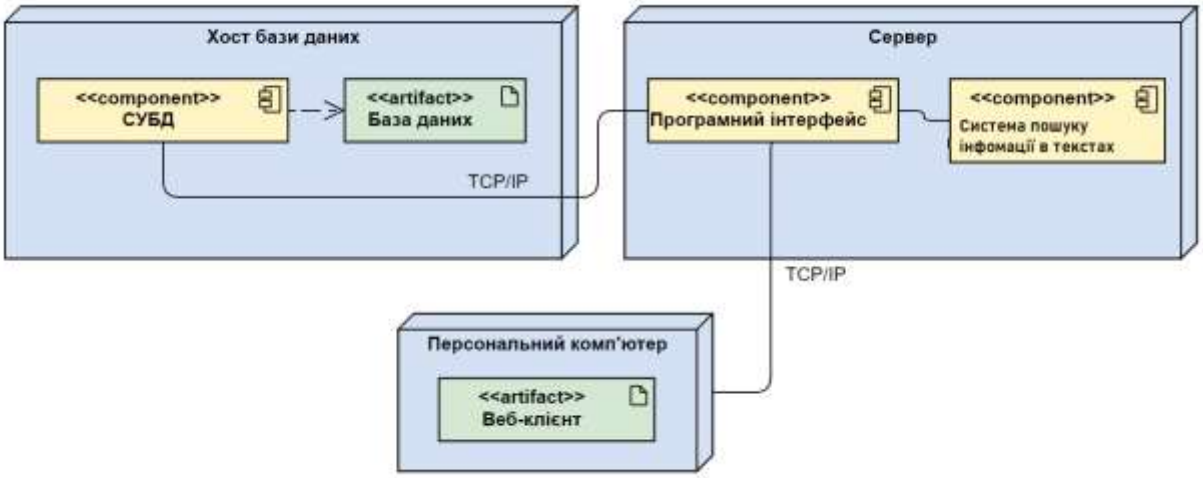


Рисунок 2.5 – Діаграма розгортування інтелектуальної технології пошуку інформації в текстах

2.8 Висновок до розділу 2

Таким чином, у цьому розділі висвітлюються ключові компоненти інтелектуальної технології, яка знаходить інформацію в тексті, створює загальну структурну схему програми, створює основні алгоритми програми та створює UML-діаграми розгортань та послідовностей. розвинений. Розроблені схеми та діаграми докладно описують структуру та принцип роботи програмного забезпечення. Це дозволить просунутися вперед у впровадженні інтелектуальних технологій пошуку інформації у тексті.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕЛЕКТУАЛЬНОГО ДОДАТКУ ПОШУКУ ІНФОРМАЦІЇ В ТЕКСТАХ

3.1 Вибір мови програмування та баз даних для реалізації інтелектуальної пошуку інформації в текстах

Визначимо інструменти, за допомогою яких буде створюватись програмна реалізація інтелектуального додатку пошуку інформації в текстах. Так, нам потрібно знайти мову програмування для розробки веб-додатку, панелі адміністрування, програмного інтерфейсу, модуля авторизації, а також системи керування базами даних(СКБД) для збереження даних.

Для аналізу і порівняння оберемо 3 мови програмування: C++, TypeScript, Python.

C++ — це об’єктно-орієнтована мова програмування загального призначення. Його створив Б’ярн Страуструп у Bell Labs приблизно в 1980 році. C++ дуже схожий на C (винайдений Деннісом Річі на початку 1970-х років). C++ настільки сумісний із C, що він, ймовірно, скомпілює понад 99% програм на C, не змінюючи жодного рядка вихідного коду. Хоча C++ є значною мірою добре структурованою та безпечнішою мовою, ніж C, оскільки вона базується на ООП [16].

TypeScript додає явну систему типів до JavaScript, дозволяючи суворе дотримання типів змінних. TypeScript виконує перевірку типу під час транспіляції — форми компіляції, яка перетворює код TypeScript на код JavaScript, який розуміють веб-браузери та Node.js. TypeScript відрізняється від JavaScript можливістю явного статичного призначення типів, підтримки використання повноцінних класів (як у традиційних об’єктно-орієнтованих мовах), а також підтримки підключення модулів, що призвело підвищити швидкість розробки, полегшити читаємість, рефакторинг і повторне використання коду, допомогти здійснити пошук помилок на етапи розробки та компіляції, і, можливо, прискорити виконання програми [16].

Python — це динамічна, високорівнева та інтерпретована мова програмування загального призначення. Він підтримує підхід об'єктно-орієнтованого програмування для розробки програм. Він простий і легкий у вивченні та надає багато високорівневих структур даних [16].

У таблиці 3.1 наведена порівняльна характеристика мов програмування.

Таблиця 3.1 – Порівняльна характеристика мов програмування

Критерій/мова	C++	TypeScript	Python
Керування пам’яттю	+	-	+
Компіляця	+	+	-
Робота у браузері	+	-	+
Виконання на клієнтській частині браузера	-	+	-
Підключення сценаріїв	-	+	+
Робота на серверній частині	+	+	+
Наявність пакетів для машинного навчання	-	-	+
Робота на багатоьох платформах	+	-	-

Мова програмування TypeScript була обрана для створення програмного забезпечення, тому що нам потрібно створювати веб-програми. Це важливо при роботі на стороні клієнта браузера. Крім того, TypeScript також можна використовувати на серверній частині, що знижує витрати на розробку та спрощує майбутню підтримку.

Розглянемо три системи управління базами даних: Firebase, MySQL та Redis.

Firebase — це хмарна система, яка дає розробникам потужні інструменти для створення програм для Android, iOS і веб-додатків. Він пропонує такі функції, як автентифікація користувачів, зберігання файлів і розміщення

програм. Firebase також полегшує інтеграцію з іншими службами, такими як Google Analytics і AdMob. Firebase є зручним рішенням для розробки таких програм. Це серверна платформа розробки, яка пропонує послуги баз даних у реальному часі та хостинг. Крім того, він також допомагає в управлінні безпекою програми та автентифікацією користувачів. У цьому блозі ми обговоримо переваги та недоліки Firebase. База даних Google Firebase пропонує надійні можливості для розробки веб-додатків і мобільних додатків. Деякі популярні Firebases включають базу даних у реальному часі, служби хмарного зберігання, такі як Crashlytics, або Analytics із такими функціями, як єдиний вхід [17].

MySQL — це система керування реляційною базою даних або RDBMS, що означає, що вона зберігає та представляє дані в табличній формі, організованій у рядки та стовпці. Паревагою є більш безпечний, оскільки він складається з надійного рівня безпеки даних для захисту конфіденційних даних від зловмисників, а паролі в MySQL зашифровані, доступний для безкоштовного завантаження та використання з офіційного сайту MySQL, забезпечує можливість запуску клієнтів і сервера на одному комп’ютері або на різних комп’ютерах через Інтернет або локальну мережу. Недоліком моделі реляційної бази даних є те, що вона використовує багато зовнішньої пам'яті та може призвести до аномалій без нормалізації. MySQL не дуже ефективний при роботі з великими базами даних. [7].

Redis – резидентна система управління базами даних, відноситься до NoSQL-систем. Це одна з найшвидших технологій кешування на ринку. Його пропускна здатність часто обмежується доступною потужністю або розміром мереж. Дані зберігаються у вигляді «ключ – значення». Його гнучкі структури даних підтримують різноманітні сценарії зберігання даних. Кожна пара ключ-значення може зберігати до 512 МБ даних, що дає можливість зберегти величезну кількість даних лише в одному об’єкті.. Redis не є традиційною СУБД та використовується для специфічних задач, наведених вище [8].

Проаналізувавши і порівнявши наведені вище системи управління базами даних, вона не вимагає жодних додаткових інструментів для вилучення даних, тому більше підходить для розробки інтелектуальних додатків, що гарантують швидкість розробки, підтримку та використання при використанні мови TypeScript. Вибрано СУБД Firebase. Легко розширюється до будь-якого необхідного формату та має гнучку схему даних, що спрощує подальшу підтримку розробленої системи [8].

Виберіть модель SPA веб-застосунків для реалізації клієнтської частини інтелектуальної технології. SPA (односторінкова програма) — це тип веб-програми, в якому програма не перезавантажується при перемиканні сторінок, що забезпечує кращий інтерфейс користувача. Розгляньте бібліотеку React та фреймворки Vue.js та Angular, щоб вибрати інструмент для реалізації SPA

React – бібліотека, що розроблена компанією Facebook. React використовує мову шаблонів JSX, що спрощує розробку об'єктної моделі документа (DOM). Створити динамічну веб-програму саме з рядками HTML було складно, оскільки це вимагає складного кодування, але React JS вирішив цю проблему та спростив її. Він забезпечує менше кодування та надає більше функціональних можливостей. Він використовує JSX (розширення JavaScript), який є особливим синтаксисом, що дозволяє HTML-лапкам і синтаксису тегів HTML відтворювати окремі підкомпоненти. Він також підтримує побудову машиночитаних кодів. Веб-програма ReactJS складається з кількох компонентів, і кожен компонент має власну логіку та елементи керування. Ці компоненти відповідають за виведення невеликого багаторазового фрагмента HTML-коду, який можна повторно використовувати будь-де, де вам це потрібно. Багаторазовий код допомагає спростити розробку та підтримку ваших програм. Ці компоненти можна вкладати в інші компоненти, щоб дозволити створювати складні програми з простих будівельних блоків. ReactJS використовує механізм на основі віртуальної DOM для заповнення даними в HTML DOM. Віртуальний DOM працює швидко, оскільки змінює

лише окремі елементи DOM замість того, щоб щоразу перезавантажувати повний DOM. [18].

Vue js — це прогресивний фреймворк, розроблений для інтерфейсної розробки веб-додатків і веб-сайтів. Він дотримується архітектури Model–View–View-Model (MVVM) і в основному використовується для створення інтерфейсів користувача та односторінкових програм. До переваг Vue можна віднести Подібно до Angular і React, фреймворк Vue js має компонентну архітектуру. Це означає, що весь зовнішній код програми можна розділити на незалежні компоненти. Ці компоненти, що складаються з шаблону, логіки та стилів, пов’язані разом, щоб сформувати веб-програму. Компонентний підхід Vue дає змогу створювати повторно використовувані однофайлові компоненти. Усередині компонента шаблон, логіка та стилі поєднані між собою. Замість того, щоб розділяти код на довільні шари, Vue об’єднує компоненти, які можна повторно використовувати, у функцію.

Angular — це фреймворк із відкритим вихідним кодом, який підтримує Google. Angular підтримує вас за допомогою впевненої платформи, оскільки ваша команда та програми зростають. Це дозволяє створювати придатні для тестування та масштабовані інтерфейси користувача, які добре працюють у міру їхнього зростання. Інтерфейс командного рядка (CLI) Angular підтримує запуск проекту та його розгортання до продуктивності, зберігаючи узгодженість між командами та проектами. Angular є найбільш повною з цих платформ, надаючи інструменти для всього циклу розробки веб-застосунків, включаючи управління зберіганням даних, навігацію та багато іншого. Складність фреймворку також може бути недоліком, оскільки для освоєння принципів роботи Angular потрібно набагато більше часу, ніж для інших фреймворків. [21].

Зваживши плюси та мінуси перерахованих вище інструментів, я вибираю JS-бібліотеку React для програмної реалізації клієнтської частини мого інтелектуального додатка для пошуку інформації за текстом виписок. Він пропонує оптимальну роботу з деревами DOM, із простими процесами

низькорівневого входу та подальшою підтримкою розроблених додатків. Розглянемо інструменти для спрощення написання серверної частини інтелектуального додатку: бібліотеки Express.js, Rest, GraphQL.

Express — це простий і гнучкий фреймворк веб-додатків Node.js, який забезпечує надійний набір функцій для веб- і мобільних додатків. Завдяки безлічі методів утиліт HTTP та проміжного програмного забезпечення у вашому розпорядженні створити надійний API можна швидко й легко. Зазначена модель запиту спрощує процес обробки запитів, охоплюючи більшість завдань, які необхідно вирішити під час обміну інформацією між клієнтом та сервером. Express забезпечує тонкий рівень основних функцій веб-додатків, не приховуючи функцій Node.js. Express є найпопулярнішою бібліотекою для створення серверних додаткіі додатків на TypeScript [12].

NestJS — це розширювана, універсальна, прогресивна платформа Node.js з відкритим кодом для створення привабливих і вимогливих серверних систем. Наразі це найшвидше зростаючий фреймворк Node.js у TypeScript. NestJS використовується для написання масштабованих, тестованих і слабозв’язаних програм. Це виводить масштабовані сервери Node.js на абсолютно новий рівень. Він підтримує такі бази даних, як PostgreSQL, MongoDB, MySQL. NestJS знаходиться під сильним впливом Angular, React і Vue і пропонує впровадження залежностей.

GraphQL — це мова запитів для API та середовище виконання для виконання цих запитів із наявними даними. GraphQL надає повний і зрозумілий опис даних у вашому API, дає клієнтам можливість вимагати саме те, що їм потрібно, і нічого більше, полегшує розвиток API з часом і надає потужні інструменти розробника. Сервіс GraphQL створюється шляхом визначення типів і полів у цих типах, а потім надання функцій для кожного поля кожного типу. Недоліком GraphQL можна визначити Реалізувати спрощений кеш за допомогою GraphQL складніше, ніж реалізувати його в REST. В REST API ми отримуємо доступ до ресурсів за допомогою URL-адрес, тому ми можемо кешувати на рівні ресурсу, оскільки URL-адреса

ресурсу є ідентифікатором. З іншого боку, у GraphQL це дуже складно, оскільки кожен запит може відрізнятися, навіть якщо він працює з тією самою сутністю. Але більшість бібліотек, створених на основі GraphQL, пропонують ефективний механізм кешування [21].

Після аналізу сильних та слабких сторін інструментів реалізації серверної частини для створення інтелектуальної програми для аналізу тексту було вибрано бібліотеку Express.js. Це пов'язано з простотою архітектури клієнт-серверної взаємодії, великою базою документації та великою спільнотою розробників. Прискоріть та спростіть процес розробки бекенда.

3.2 Розробка структури інформаційної технології пошуку інформації в текстах

Розробимо загальну структурну схему інформаційної технології пошуку інформації в текстах. Технологія складається з п’яти основних компонентів:

- веб-клієнт;
- програмний інтерфейс додатку(API);
- система вилучення інформації;
- база даних
- модуль авторизації

Опишемо кожен із компонентів.

Веб-клієнт стає спрощеним майданчиком для спілкування, де як зареєстровані, так і анонімні користувачі можуть залишати свої повідомлення. Клієнт отримує дані із сервера та відправляє опубліковані пости на аналіз.

Програмний інтерфейс представляє собою серверну частину, включає обробку даних, збереження, редагування та видалення записів у базі даних.

Система вилучення інформації проводить аналіз та класифікацію текстів за заданим запитанням. Проводиться вилучення інформації за допомогою алгоритму Michelson’a.

База даних виконує функцію збереження даних про результати роботи системи вилучення інформації.

Основна структурна схема роботи системи показана на рисунку 3.1

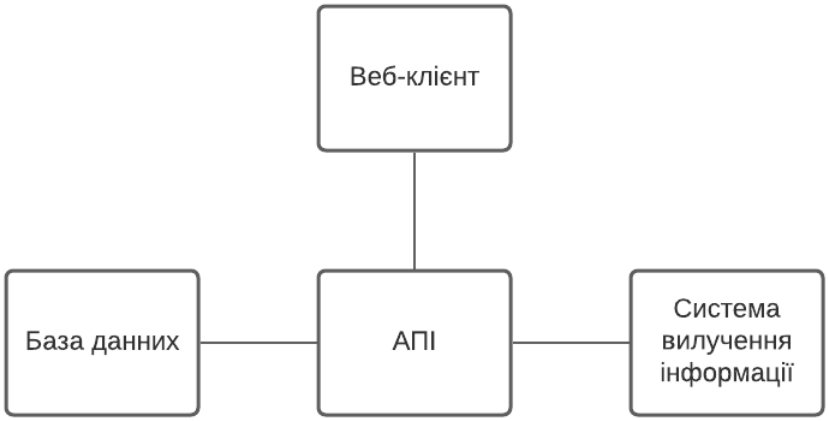


Рисунок 3.1 – Загальна структурна схема інтелектуальної технології пошуку інформації в тексті

3.3 Розробка основного алгоритму роботи інформаційної технології пошуку інформації в тексті

Використовує текстові та графічні описи ключових алгоритмів інтелектуальної технології, які шукають інформацію у тексті для відображення програмних циклів.

Кроки алгоритму:

1. Ідентифікація користувача.
2. Вибір дії: якщо користувач обирає свій завантажити новий текст перейти на крок 3, якщо обрати текст то крок 9.
3. Завантаження тексту на сервер.
4. Введення запитання.
5. Відправка запитання на сервер.
6. Аналіз тексту, пошук відповіді.

7. Отримання відповіді на запитання.
8. Візуалізація отриманого результату.
9. Вибір дії, якщо обрання тексту з вже завантажених , якщо ні перейти на крок 11.
10. Завантаження текстів з бази даних, перейти на крок 4.
11. Завершення роботи

Графічна інтерпретація описаного алгоритму наведена на рисунку 3.2.

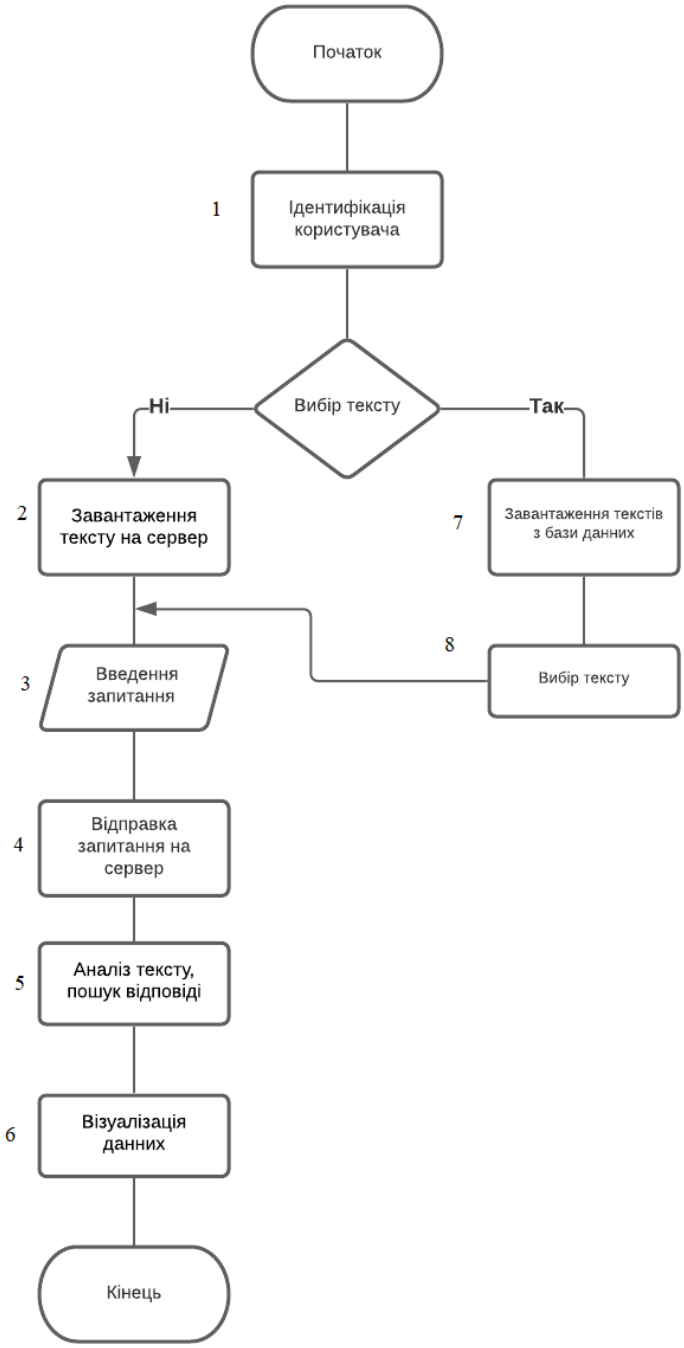


Рисунок 3.2 – Схема основного алгоритму роботи інтелектуальної технології пошуку інформації в текстах

3.3 Програмна реалізація компонентів інтелектуального додатку пошуку інформації в текстах

Розглянемо суть програмної реалізації інтелектуального додатка для пошуку інформації в тексті.

Головною частиною технології є серверна частина, яка забезпечує взаємодію між клієнтом і базою даних, а також виконує інтелектуальний пошук в текстах що завантажили користувачі.

Ініціалізація даних та ініціалізує прослуховувач запитів (маршрут). Потім усі запити, що вимагають автентифікації користувача, перевіряються на дійсність маркера JWT, який ідентифікує сеанс та користувача. Для самої аутентифікації використовуйте відповідний запит /auth. Для цього потрібний логін та пароль користувача. Пароль користувача зберігається у хешованому вигляді та розшифровується на сервері. Якщо дані збігаються, токен сеансу JWT повертається користувачеві. Це потрібно для подальшої роботи інтелектуального модуля. Фрагмент коду, що реалізує авторизацію користувачів:

```
export function* googleLoginSaga() {
  try {
    const userAuga: GeneratorFunction = yield call(
      injectSaga,
      'userAuthSaga',
    )

    const socialAuthService: ReturnType<typeof
injectSocialAuthService> =
      yield call(injectSocialAuthService)

    const sociaResponse: SocialAuthResponseModel = yield
call([
```

```
            socialAuthService,
            socialAuthService.handleGoogleLogin,
          ])

          yield call(userAuthSaga, {
            socialLogioken: socialAuthResponse.token,
            user: socialAuthRese.user,
            loginType: 'GOOGLE',
          })

          yield  put(sendAnalyticEvent({  name:  'login',  data:  {
method: 'Google' } }))
          yield put(googlenSuccess())
        } catch (e) {
          yield put(
            sendAnalyticsEvent({
              name: 'e_error_lin_fail',
              data: {
                error_category: 'loginfail_gg',
                error_code: 'loginfail',
                message: transformSignInErrorMe (e),
              },
            }),
          )
          yield put(googleLoginFailure(e as Error))
        }
      }
```

Коли користувач запитує виділений текст, ми шукаємо інформацію в цьому тексті. Усі отримані результати заносяться до бази даних з інформацією про тексті, після чого актуальні дані оновлюються на клієнті. Фрагмент коду, що реалізує інтелектуальний пошук інформації:

```
const findNewAnswer = async (q) => {
    const m = await q.load();
```

```
        const a = await model.findans(q, text);
        setAns(a)
    }
```

Розглянемо ключові моменти розробки клієнтської частини інтелектуальної програми для пошуку інформації в тексті. Для розробки інтерфейсу користувача виберіть бібліотеку Ant Design, яка надає доступ до багатьох компонентів інтерфейсу React, що спрощує процес розробки.

Щоб авторизувати користувачів, ви повинні переконатися, що дані, які вони запроваджують, відповідають вашим вимогам, особливо формат електронної пошти та мінімальна довжина пароля. Фрагмент коду, що виконує перевірку валідності полів авторизації:

```
        const authourazation = () => {
            firebase.auth().signInWithEmailAndPassword(email,
    password)
                .t((uC) => {
                    const user = uCl.user;
                    localStorage.setItem('token', user.uid);
                })
                .c((error) => {
                    var errorCode = error.code;
                    var errorMessage = error.message;
                });
        }

        return (
            <div>
                <div  span={10}>
                    <form
                        name="aaa"
                        initialValues={{
                            memory: true,
                        }}
                        onEnd={onEnd}
                        onFinishFaileddd={onFinishFailed}
                    >
                        <p>Sign In</p>
                        <Form.Item
                            label="Email"
                            name="email"
                            rules={[
                                {
                                    required: true,
```

```
                                message:    'Please    input    your
username!',
                            },
                        ]}
                    >
                        <Input    type={'email'}    onChange={e    =>
setEmail(e.target.value)} />
                    </Form.Item>

                    <Form.Item
                        label="Password"
                        name="password"
                        rules={[
                            {
                                required: true,
                                message:    'Please    input    your
password!',
                            },
                        ]}
                    >
                        <Input.Password       onChange={e       =>
setPassword(e.target.value)}/>
                    </Form.Item>
                    <Form.Item {...tailLayout}>
                        <Button    onClick={auth}    type="primary"
htmlType="submit">
                            Submit
        </Row>
    );
};
```

Приклад роботи перевірки полів наведено на рисунку 4.1.

Рисунок 3.3 – Приклад роботи перевірки полів авторизації на відповідність встановленим правилам

Обмін даними із сервером здійснюється за допомогою асинхронних запитів, реалізованих за допомогою бібліотеки redux-api-middleware. Це спрощує процес надсилання, обробки та зберігання даних, отриманих із сервера. Усі дані про клієнта зберігаються у централізованому сховищі.

3.4 Тестовий приклад роботи програми і аналіз результатів

Визначимо критерії, за якими будемо проводити тестування розробленої програми:

- веб-клієнт інтелектуального додатку доступний за відповідним посиланням;
- на відкриття сторінки авторизації відображаються відповідні поля авторизації, відбувається перевірка введених даних та виконується процес авторизації;
- список текстів користувача доступний на відповідній сторінці веб-клієнта;
- завантаження текстів користувача доступний на відповідній сторінці веб-клієнта;
- доступний голосовий і мануальний ввід запитання;
- при завершенні роботи видаляються дані про сесію.

Проведемо тестування відповідно до визначених критеріїв.

Відкриємо веб-клієнт інтелектуального додатку пошуку інформації. Отримаємо список з текстами користувача (рис. 3.4).

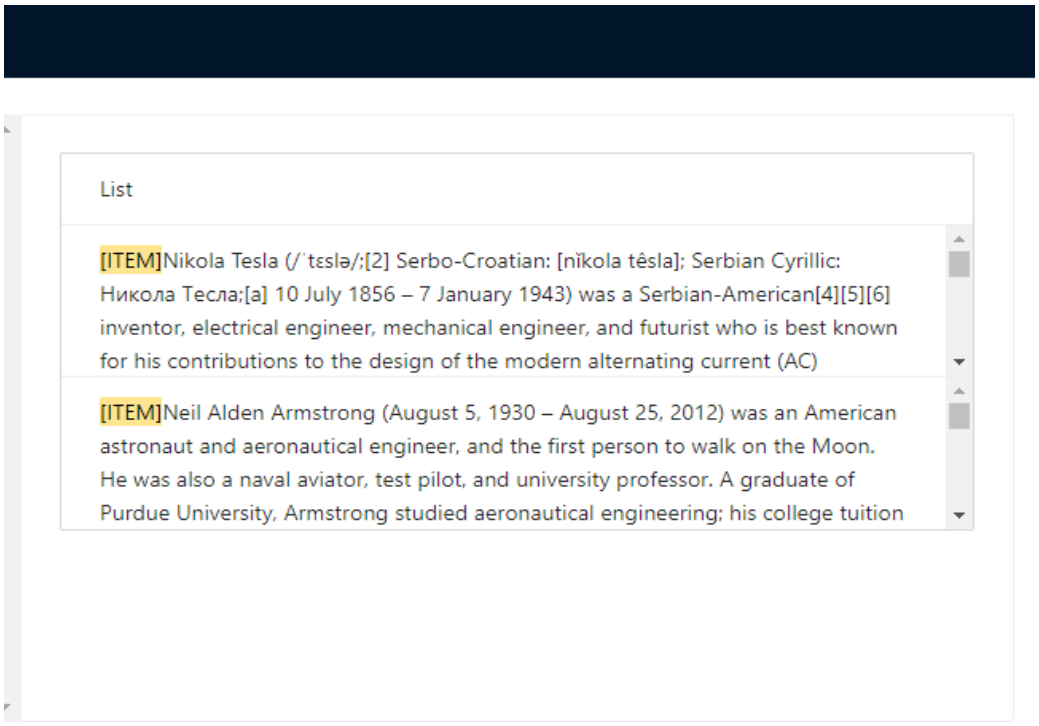


Рисунок 3.4 – Компонент з текстами користувача

Відкриємо сторінку авторизації. Заповнимо поля невідповідними даними, отримуємо повідомлення про помилку, завершити процес авторизації не є можливим (рис. 3.4). Введемо вірні дані та завершимо процес авторизації. Перевіримо запис даних про сесію користувача (рис. 3.5). Відкриємо головну сторінку, отримуємо форму для завантаження тексту (рис. 3.5)

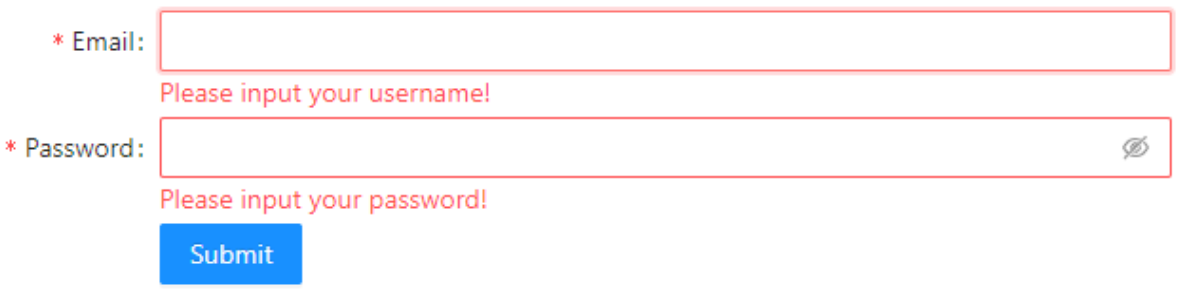


Рисунок 3.4 – Виведення помилок про невірні дані

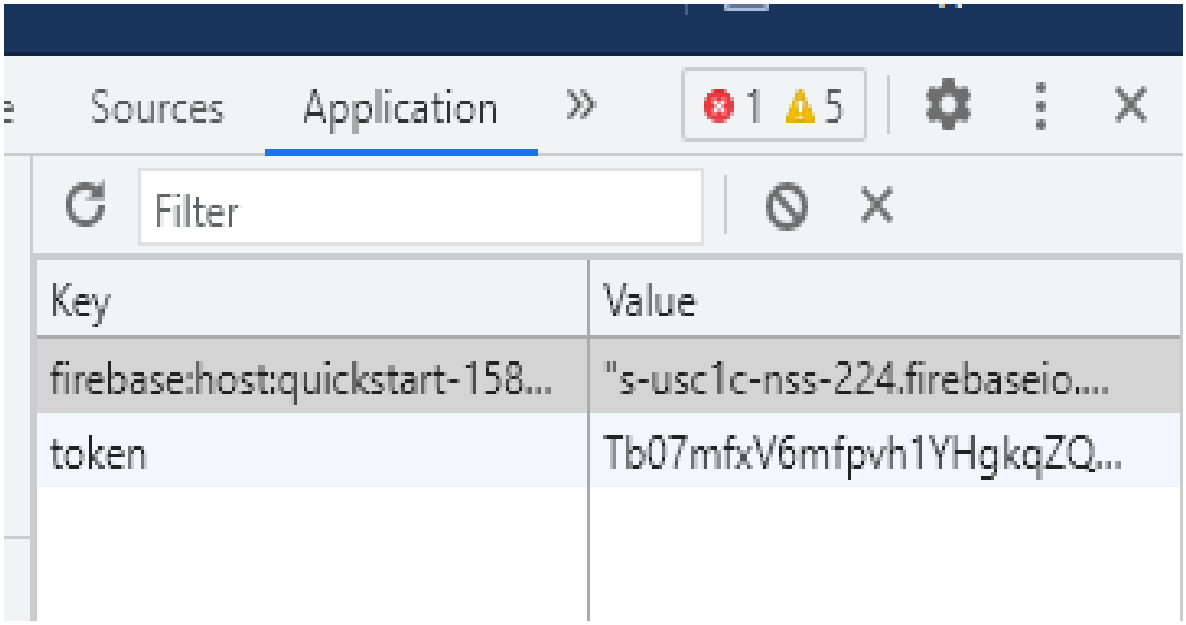


Рисунок 3.5 – Записані дані про сесію користувача

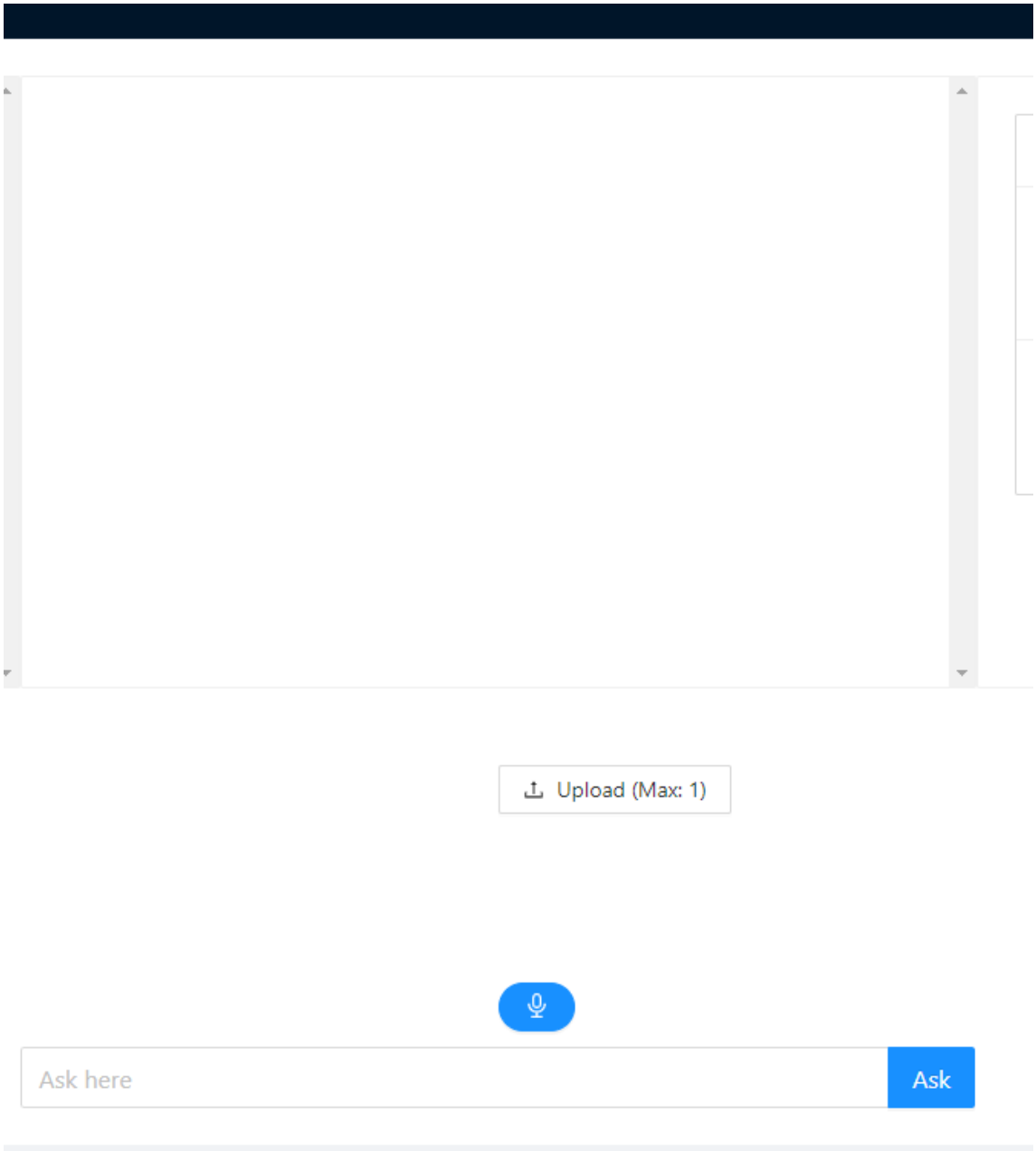


Рисунок 3.6 – Форма пошуку інформації в текстах

Завантажимо текст, для цього натиснемо кнопку завантажити. Результати сортування відображено на рисунку 3.7.

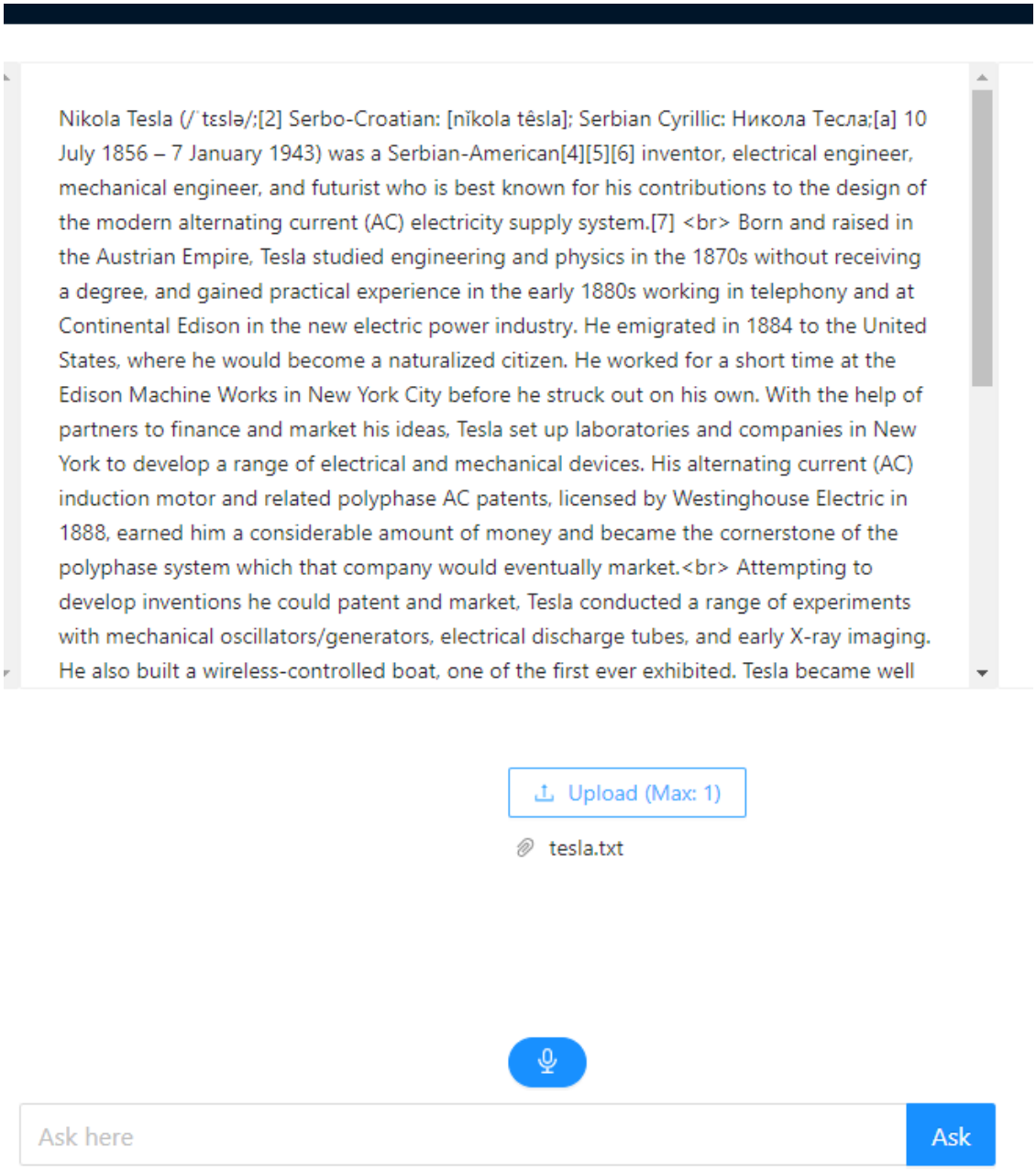


Рисунок 3.7 – Форма з завантаженим текстом

Напишемо запитання у поле для запитання. Натиснемо кнопку задати запитання. В результаті отримаємо відповідь або повідомлення про помилку (рис. 3.8).

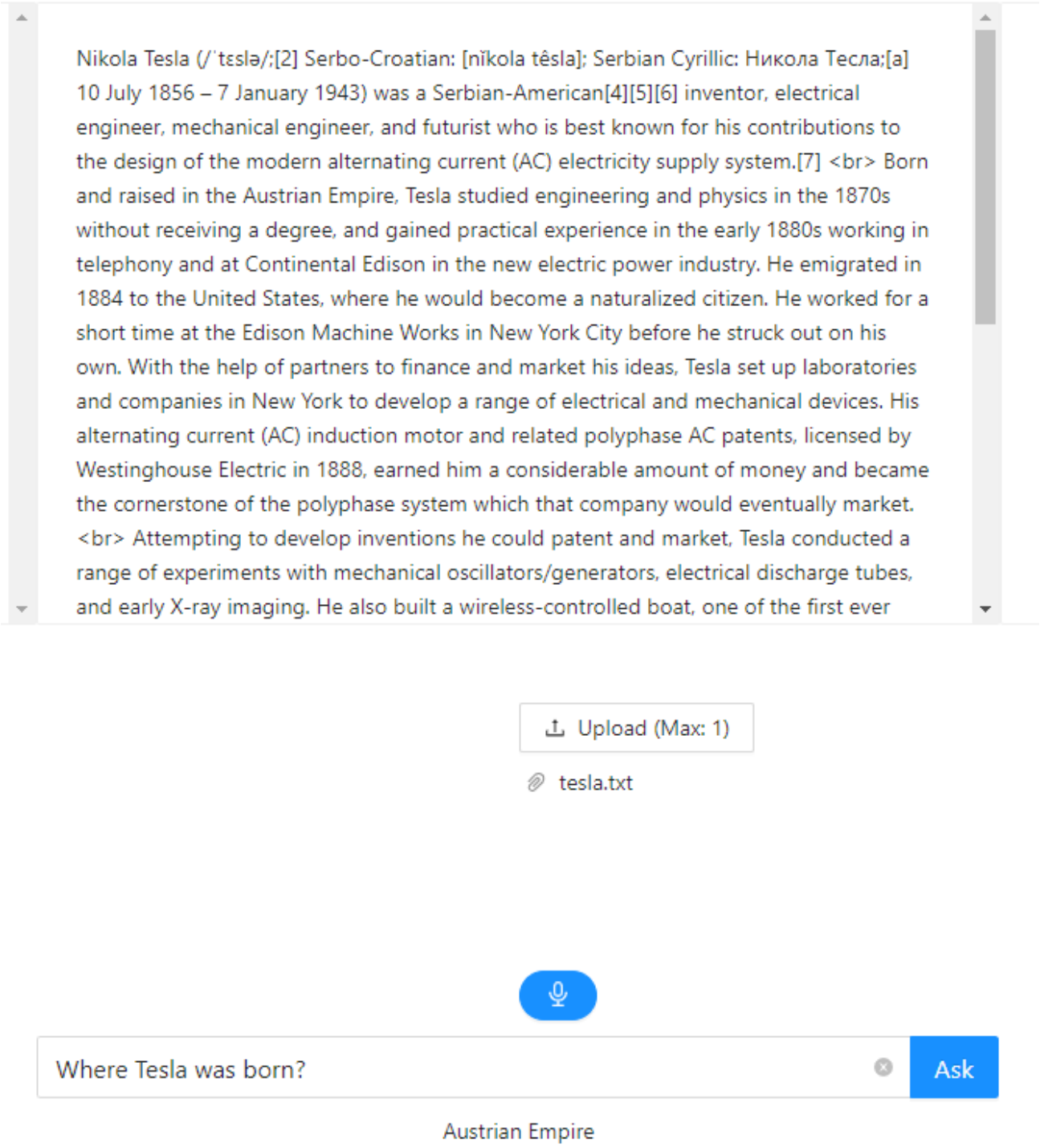


Рисунок 3.8 – Результати запиту пошуку інформації

Перейдемо на сторінку завантаження текстів на облікового запису користувача. Вставимо текст у поле завантаження і натиснемо кнопку

завантажити на сервер (Рис. 3.8). Після оновлення інформації на сторінці списку збережених текстів повинен відобразитися щойно доданий новий текст у списку (Рис. 3.9).

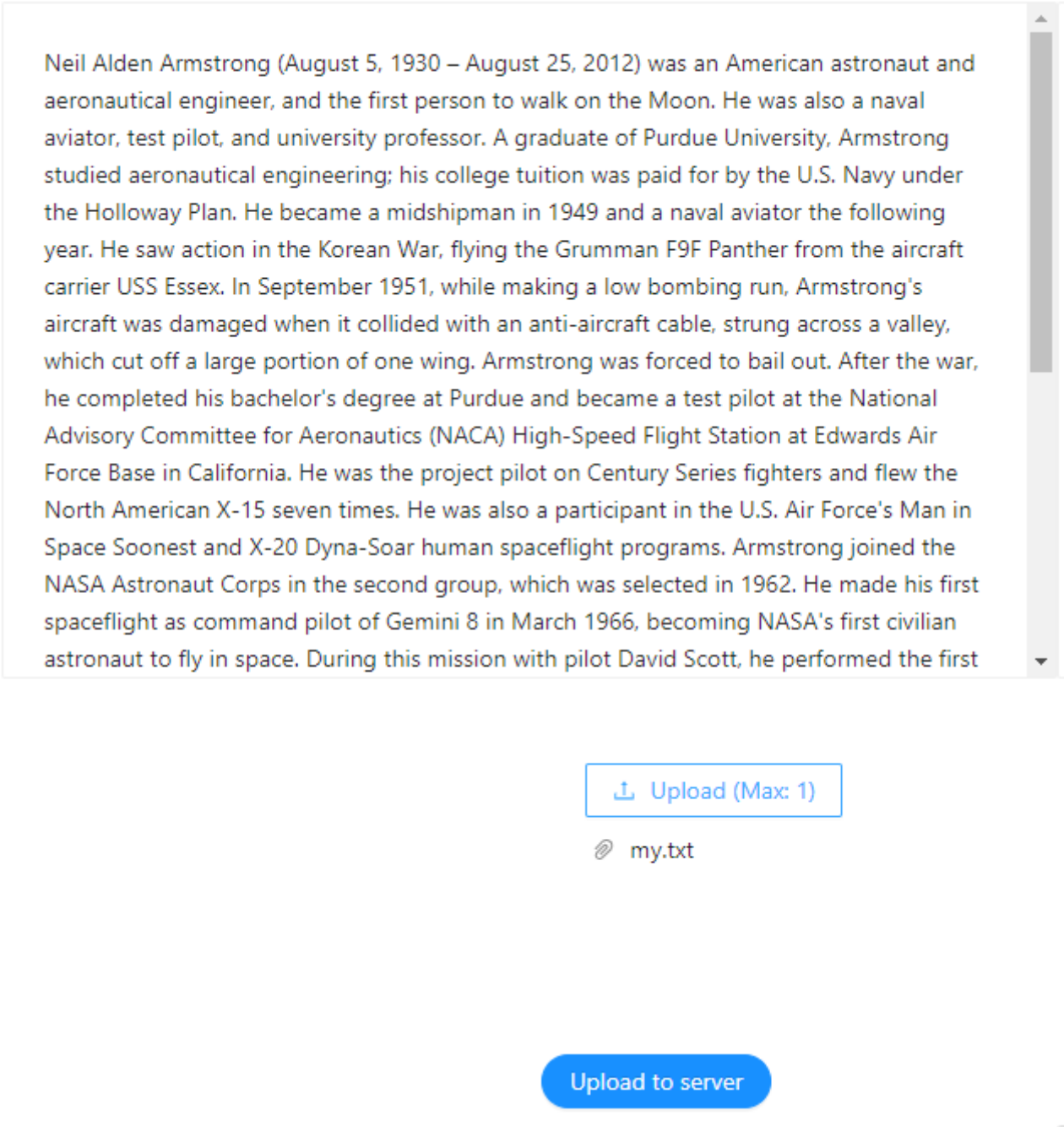


Рисунок 4.9 – Сторінка завантаження тексту в базу даних

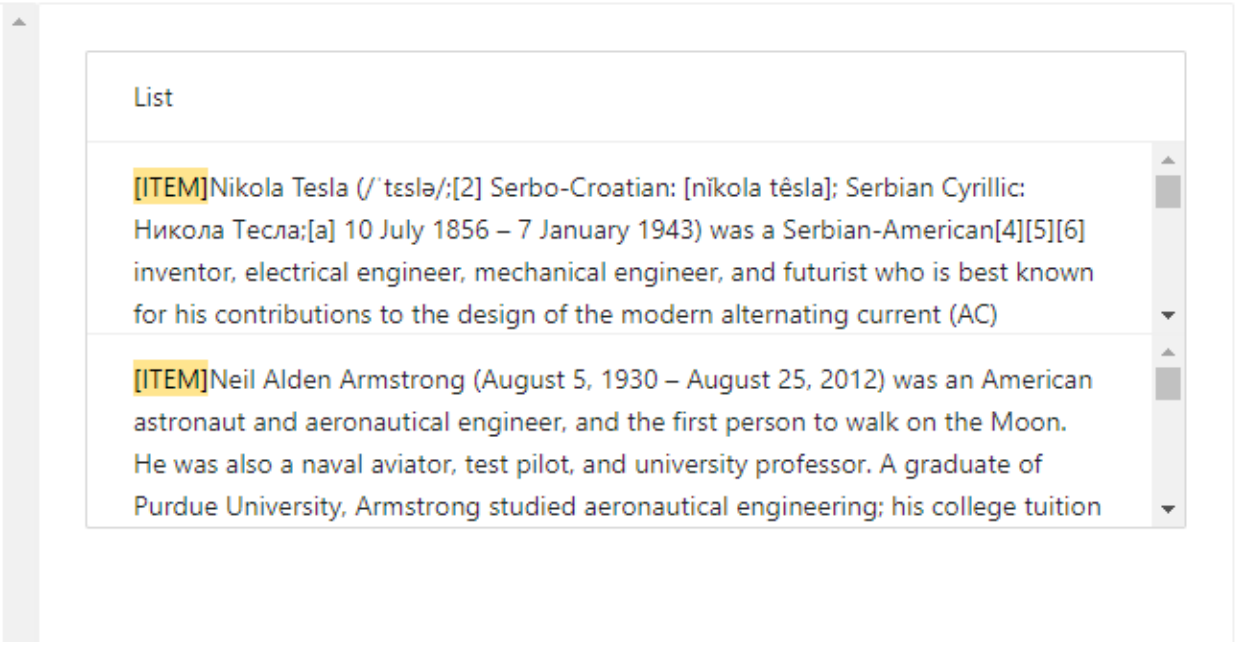


Рисунок 4.10 – Оновлена сторінка списку текстів користувача

Вийдемо зі свого облікового запису, усі дані про сесію видалено (рис. 4.11).

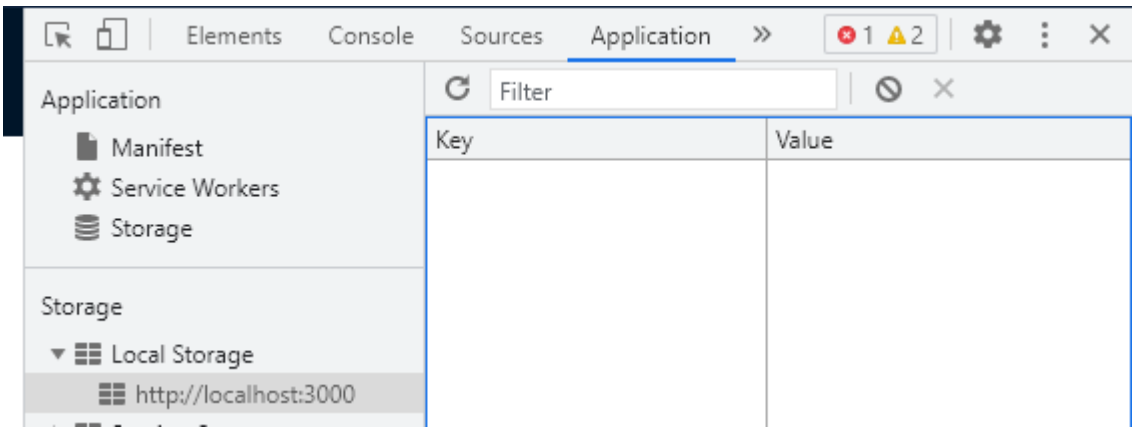


Рисунок 4.11 – Відсутність даних про сесію після завершення роботи із обліковим записом

Тому при тестуванні створеної програми її поведінка перевірялася відповідно до встановлених стандартів. Усі компоненти працювали згідно з поставленим завданням та виконували всі функції, на які були розраховані.

3.4 Висновок до розділу 3

Тому в цьому розділі ми надамо різні мови програмування та технології реалізації програм інформаційних технологій для пошуку інформації в тексті, мову програмування TypeScript, React.js, бібліотеки Express, фреймворк Node.js, базу даних MongoDB, розглянуті та порівняні. Система управління та бібліотека вибрали Tensorflow як основний алгоритм. З використанням вибраних технологій була розроблена програма, що реалізує функції та завдання, описані в попередньому розділі. Тестування розробленої програми проводилося за критеріями, заснованими на завданнях дослідження та досягненні цілей дослідження, а саме удосконалення у сфері пошуку текстової інформації.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Комерційний та технологічний аудит науково-технічної розробки

Метою даного розділу є проведення технологічного аудиту, в даному випадку нового програмного продукту інформаційної технології пошуку інформації в текстах. Особливістю інформаційної технології є те, що вона дозволяє за запитом природною мовою знаходити і показувати інформацію з текста.

Аналогом може бути TextAnalyst ціна якої 3000 грн.

Для проведення комерційного та технологічного аудиту залучають не менше 3-х незалежних експертів. Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, у відповідності із табл. 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздат-ність продук-кту в реаль-них умовах
2	Багато аналогів на малому ринку	Ринкові п Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку

Продовження табл. 4.1

Ринкові переваги					
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною .	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практик на здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так із комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Продовження табл. 4.1

11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Усі дані по кожному параметру занесено в таблиці 4.2

Таблиця 4.2 – Результати оцінювання комерційного потенціалу розробки

Критерії оцінювання	ПІБ експертів		
	Експерт 1	Експерт 2	Експерт 3
	Бали		
Технічна здійсненність концепції	4	3	4
Наявність аналогів на ринку	4	4	3
Цінова політика	4	4	3
Технічні та споживчі властивості виробу	3	3	4
Експлуатаційні витрати	3	4	4
Ринок збуту	3	3	3
Конкурентоспроможність	3	4	4
Фахівці з технічної і комерційної реалізації	4	3	3
Фінансування	3	4	3
Матеріально-технічна база	3	4	3
Термін реалізації ідеї	4	4	3
Супровідна документація	4	3	4
Сума	42	43	41
Середньоарифметична сума балів	(42+43+41) / 3 = 42		

За даними таблиці 4.2 можна зробити висновок щодо рівня комерційного потенціалу даної розробки. Для цього доцільно скористатись рекомендаціями, наведеними в таблиці 4.3.

Таблиця 4.3 - Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

Як видно з таблиці, рівень комерційного потенціалу розроблюваного нового програмного продукту є високим, що досягається за рахунок того, що інформаційна технологія відрізняється від існуючих тим, що вона дозволяє за запитом природною мовою знаходити і показувати інформацію з текста.

4.2 Прогнозування витрат на виконання науково-дослідної (дослідно-конструкторської) роботи

4.2.1 Основна заробітна плата розробників, яка розраховується за формулою:

$$З_о = \frac{M}{T_р} \cdot t \,, \tag{4.1}$$

де М – місячний посадовий оклад конкретного розробника (дослідника), грн.;

$T_р$ – число робочих днів в місяці, 21 днів;

t – число днів роботи розробника (дослідника).

Результати розрахунків зведемо до таблиці 4.1.

Таблиця 4.1 – Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	37000	1761,90	30	52857,143
Інженер	33000	1571,43	30	47142,857
Всього				100000,00

Так як в даному випадку розробляється програмний продукт, то розробник виступає одночасно і основним робітником, і тестувальником розроблюваного програмного продукту.

4.2.2 Додаткова заробітна плата розробників, які приймали участь в розробці обладнання.

Додаткова заробітна плата прийнято розраховувати як 15 % від основної заробітної плати розробників та робітників:

$$З_д = З_о \cdot 15\ \% / 100\ \% \qquad (4.2)$$

$З_д = (100000{,}00 \cdot 15\ \% / 100\ \%) = 15000{,}00$ (грн.)

4.2.3 Нарахування на заробітну плату розробників.

Згідно діючого законодавства нарахування на заробітну плату складають 22 % від суми основної та додаткової заробітної плати.

$$Н_з = (З_о + З_д) \cdot 22\ \% / 100\% \qquad (4.3)$$

$Н_з = (100000{,}00 + 15000{,}00) \cdot 22\ \% / 100\ \% = 25300{,}00$ (грн.)

4.2.4. Оскільки для розроблювального пристрою не потрібно витрачати матеріали та комплектуючи, то витрати на матеріали і комплектуючі дорівнюють нулю.

4.2.5 Амортизація обладнання, яке використовувалось для проведення розробки.

Амортизація обладнання, що використовувалось для розробки в спрощеному вигляді амортизація обладнання, що використовувалась для розробки розраховується за формулою:

$$A = \frac{Ц}{Тв} \cdot \frac{t_{вик}}{12} \text{ [грн.].} \quad (4.4)$$

де Ц – балансова вартість обладнання, грн.;

Т – термін корисного використання обладнання згідно податкового законодавства, років

$t_{вик}$ – термін використання під час розробки, місяців

Розрахуємо, для прикладу, амортизаційні витрати на комп'ютер балансова вартість якого становить 20000 грн., термін його корисного використання згідно податкового законодавства – 2 роки, а термін його фактичного використання – 1,43 міс.

$$A_{обл} = \frac{20000}{2} \times \frac{1{,}43}{12} = 1190{,}476\, грн.$$

Аналогічно визначаємо амортизаційні витрати на інше обладнання та приміщення. Розрахунки заносимо до таблиці 4.2. Для розрахунку амортизації нематеріальних ресурсів використовується формула:

$$A_{н.р.} = Ц_{н.р.} * Н_а * \frac{t_{вик.}}{12} \quad (4.5)$$

Але ліцензійна ОС та спеціалізовані ліцензійні нематеріальні ресурси - Tanserflow є безкоштовними.

Таблиця 4.2 – Амортизаційні відрахування матеріальних і нематеріальних ресурсів для розробників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортиза-ційні відрахуван-ня, грн.
Комп'ютер та комп'ютерна переферія	20000	2	1,43	1190,476
Офісне обладнення	25000	4	1,43	744,048
Приміщення	1075000	20	1,43	6398,810
Всього				8333,33

5.2.6 Тарифи на електроенергію для непобутових споживачів (промислових підприємств) відрізняються від тарифів на електроенергію для населення. При цьому тарифи на розподіл електроенергії у різних постачальників (енергорозподільних компаній), будуть різними. Крім того, розмір тарифу залежить від класу напруги (1-й або 2-й клас). Тарифи на розподіл електроенергії для всіх енергорозподільних компаній встановлює Національна комісія з регулювання енергетики і комунальних послуг (НКРЕКП). Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п, \qquad (4.6)$$

де В – вартість 1 кВт-години електроенергії для 1 класу підприємства, В = 6,2 грн./кВт;

П – встановлена потужність обладнання, кВт. П = 0,7 кВт;

Ф – фактична кількість годин роботи обладнання, годин.

$К_п$ – коефіцієнт використання потужності, $К_п$ = 0,9.

$B_е = 0,9 \cdot 0,7 \cdot 8 \cdot 30 \cdot 6,2 = 937,44$ (грн.)

5.2.7 Інші витрати та загальновиробничі витрати.

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками. Витрати за статтею «Інші витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.7)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

$I_в = 100000,00 * 110\% / 100\% = 110000$ (грн.)

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$Н_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.8)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$H_{нзв}$ = 100000,00 * 140 % / 100 % = 140000 (грн.)

5.2.9 Витрати на проведення науково-дослідної роботи.

Сума всіх попередніх статей витрат дає загальні витрати на проведення науково-дослідної роботи:

$B_{заг}$ = 100000,00+15000,00+25300,00+8333,33+937,44+110000+140000 =
= 399570,77 грн.

5.2.11 Розрахунок загальних витрат на науково-дослідну (науково-технічну) роботу та оформлення її результатів.

Загальні витрати на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються *ЗВ*, визначається за формулою:

$$ЗВ = \frac{B_{заг}}{\eta} \text{ (грн)}, \qquad (5.9)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи.

Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η=0,1; технічного проектування, то η=0,2; розробки конструкторської документації, то η=0,3; розробки технологій, то η=0,4; розробки дослідного зразка, то η=0,5; розробки промислового зразка, то η=0,7; впровадження, то η=0,9. Оберемо η = 0,5, так як розробка, на даний момент, знаходиться на стадії дослідного зразка:

ЗВ = 399570,77 / 0,5 = 799142 грн.

4.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнювальним позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї розробки.

Для того, щоб розрахувати можливе зростання чистого прибутку у потенційного інвестора від можливого впровадження науково-технічної розробки необхідно:

а) вказати, з якого часу можуть бути впроваджені результати науково-технічної розробки;

б) зазначити, протягом скількох років після впровадження цієї науково-технічної розробки очікуються основні позитивні результати для потенційного інвестора (наприклад, протягом 3-х років після її впровадження);

в) кількісно оцінити величину існуючого та майбутнього попиту на цю або аналогічні чи подібні науково-технічні розробки та назвати основних суб'єктів (зацікавлених осіб) цього попиту;

г) визначити ціну реалізації на ринку науково-технічних розробок з аналогічними чи подібними функціями.

При розрахунку економічної ефективності потрібно обов'язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій до отримання прибутку минає чимало часу. При оцінюванні ефективності інноваційних проектів передбачається розрахунок таких важливих показників:

− абсолютного економічного ефекту (чистого дисконтованого доходу);

− внутрішньої економічної дохідності (внутрішньої норми дохідності);

− терміну окупності (дисконтованого терміну окупності).

Аналізуючи напрямки проведення науково-технічних розробок, розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором можна об’єднати, враховуючи визначені ситуації з відповідними умовами.

4.3.1 Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

$$\Delta П_i = (\pm \Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (4.10)$$

де $\pm\Delta Ц_o$ – зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу;

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки;

$Ц_o$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, $Ц_o = Ц_б \pm \Delta Ц_o$;

$Цб$ – вартість програмного продукту у році до впровадження результатів розробки;

ΔN – збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик;

λ – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт $\lambda = 0{,}8333$.

p – коефіцієнт, який враховує рентабельність продукту;

ϑ – ставка податку на прибуток, у 2022 році ϑ =18%.

Припустимо, що при прогнозованій ціні 1800 грн. за одиницю виробу, термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти її ціну на *100* грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року – на 5000 шт., протягом другого року – на 7500 шт., протягом третього року на 10000 шт. До моменту впровадження результатів наукової розробки реалізації продукту не було:

*$\Delta П_1$ = (0*100 + (1800 + 100)*5000)* 0,8333* 0,35) * (1 - 0,18) = 2152499,914 грн.*
*$\Delta П_2$ = (0*100 + (1800 + 100)*(5000+7500)* 0,8333* 0,35) * (1 - 0,18) = 5680208,106 грн.*
*$\Delta П_3$ = (0*100 + (1800 + 100)*(5000+7500+10000)* 0,8333* 0,35) * (1 - 0,18) = 10224374,591 грн.*

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 18057082,61 грн.

4.3 Розрахунок ефективності вкладених інвестицій та періоду їх окупності.

Розраховуємо приведену вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої науково-дослідної (науково-технічної) роботи, грн;

Т – період часу, протягом якою виявляються результати впровадженої науково-дослідної (науково-технічної) роботи, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0,05...0,15$;

t – період часу (в роках).

Збільшення прибутку ми отримаємо починаючи з першого року:

ПП= $(2152499,914/(1+0,1)^1)+(5680208,106/(1+0,1)^2)+(10224374,591/(1+0,1)^3)$ = 1956818,10 + 4694386,865 + 7681723,96 = 14332928,93 грн.

Далі розраховують величину початкових інвестицій *PV*, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} * ЗВ, \quad (4.12)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}=2...5$, але може бути і більшим;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$PV = 2 * 799142 = 1598283,09$ *грн.*

Тоді абсолютний економічний ефект $E_{абс}$ або чистий приведений дохід *(NPV, Net Present Value)* для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV, \quad (4.13)$$

$E_{абс}$ = 14332928,93 – 1598283,09 = 12734645,83 грн.

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів даної науково-дослідної (науково-технічної) роботи може бути доцільним.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність або показник внутрішньої но-рми дохідності *(IRR, Internal Rate of Return)* вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього використаємо формулу:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (4.14)$$

$T_ж$ – життєвий цикл наукової розробки, роки.

$E_в = \sqrt[3]{(1 + 12734645,83/1598283,09} - 1 = 1,078$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \quad (4.15)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2022 році в Україні d = (0,09…0,14);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5).

$$\tau_{\min} = 0,14 + 0,05 = 0,19.$$

Так як $E_в > \tau_{min}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в}, \qquad (4.16)$$

$T_{ок}$ = 1 / 1,078 = 0,93 р.

Оскільки $T_{ок}$ < 3-х років, а саме термін окупності рівний 0,93 роки, то фінансування даної наукової розробки є доцільним.

Висновки до розділу: економічна частина даної роботи містить розрахунок витрат на розробку нового програмного продукту, сума яких складає 799142 гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт за ціною дешевший за аналог і є висококонкурентоспроможним. Період окупності складе близько 0,93 роки.

ВИСНОВКИ

При виконанні магістерської кваліфікаційної роботи розв’язано задачу розробки інформаційної технології та програмного забезпечення пошуку інформації в текстах.

1. Проаналізовано існуючі рішення та аналоги інтелектуального додатку пошуку інформації в текстах.
2. Обґрунтовано вибір методу розв’язання задачі.
3. Розроблено загальну структурну схему інформаційної технології.
4. Розроблено основний алгоритм роботи інформаційної технології.
5. Змодельовано інформаційну технологію із використанням мови UML.
6. Обрано мову програмування, бібліотеки, фреймворки та систему управління базами даних для програмної реалізації інформаційної технології.
7. Розроблено програмну реалізацію компонентів інформаційної технології

Мета дослідження – розширення функціональних можливостей пошуку – досягнута за рахунок введення нового середовища використання технології, а саме розширення браузера і створення бази даних, де зареєстровані користувачі можуть зберігати свої дані.

Під час програмної реалізації інтелектуальної технології пошуку інформації в текстах було обгрунтовано мову програмування та технології для розробки. Для реалізації технології було обрано мову програмування TypeScript, так як вона є найкращою для будування веб-застосунків.

Було визначено задачі подальшого дослідження, зокрема тестування розробленого програмного забезпечення для підтвердження досягнення поставленої мети і задач магістерської кваліфікаційної роботи, а також необхідність розрахунку економічних показників комерційного застосування розробленої інформаційної технології.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Matthew Michelson and Craig A. Knoblock. Creating Relational Data from Unstructured and Ungrammatical Data Sources. In Journal of Articial Intelligence Research 31 (2008), pages 543- 590, 2008.

2. Everything You Must Know About Searching Algorithms [Електронний ресурс] – Режим доступу: https://codete.com/blog/everything-you-must-know-about-searching-algorithms

3. Пошук інформації в інтернет. Засоби інтелектуалізації пошуку інформації [Електронний ресурс] – Режим доступу: https://ua.kursoviks.com.ua/metodychni_vkazivky/article_post/924-glava-10-poshuk-informatsii-v-internet-zasobi-intelektualizatsii-poshuku-informatsii-z-distsiplini-suchasni-informatsiyni-sistemi-i-tekhnologii-ksu-kiivskiy-slavistichniy-un

4. TextAnalist [Електронний ресурс] – Режим доступу: http://www.analyst.ru/index.php?lang=eng&dir=content/

5. Business Objects Text Analysis [Електронний ресурс] – Режим доступу: https://blogs.sap.com/2008/02/25/business-objects-text-analysis/

6. Chattermill Reviews & Product Details [Електронний ресурс] – Режим доступу: https://www.g2.com/products/chattermill/reviews

7. PWA [Електронний ресурс] – Режим доступу: https://cases.media/article/chto-takoe-pwa-i-stoit-li-ispolzovat-ego-pri-sozdanii-internet-magazina

8. MongoDB [Електронний ресурс] – Режим доступу: https://searchdata.com

9. M. Iyyer, V. Manjuhatha, J. Boyd-Graber, Hal Daume II. Deep Unordered Composition Rivals Syntactic Methods for Text Classification,

10. M. Iyyer, V. Manjuhatha, J. Boyd-Graber, Hal Daume III. Deep Unordered Composition Rivals Syntactic Methods for Text Classification.

11. The Mobile App Architecture Guide for 2023 [Електронний ресурс] – Режим доступу: https://www.netsolutions.com/insights/mobile-app-architecture-guide/

12. Microservices [Електронний ресурс] – Режим доступу: https://www.clickittech.com/devops/microservices-vs-monolith/

13. SPA (Single-page application) [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Glossary/SPA

14. Загальна характеристика UML [Електронний ресурс] – Режим доступу: http://www.informicus.ru/default.aspx?SECTION=6&id=73&subdivisionid=2.

15. Whats UML [Електронний ресурс] – Режим доступу: https://www.lucidchart.com/pages/what-is-UML-unified-modeling-language

16. C++ [Електронний ресурс] – Режим доступу: https://www.guru99.com/cpp-tutorial.html

17. Advantages of Firebase [Електронний ресурс] – Режим доступу: https://www.linkedin.com/pulse/advantages-disadvantages-firebase-nav-adalyn/?trk=portfolio_article-card_title

18. React JS Pros and Cons [Електронний ресурс] – Режим доступу: https://www.javatpoint.com/pros-and-cons-of-react

19. Д. Фленаган. JavaScript. Подробное руководство.

20. MongoDB [Електронний ресурс] – Режим доступу: https://www.mongodb.com/.

21. MySQL – Documentation [Електронний ресурс] – Режим доступу: https://dev.mysql.com/doc/.

22. Redis – Documentation [Електронний ресурс] – Режим доступу: https://redis.io/documentation.

23. ReactJS - Docs [Електронний ресурс] – Режим доступу: https://reactjs.org/docs/.

24. Vue.js - Guide [Електронний ресурс] – Режим доступу: https://vuejs.org/v2/guide/.

25. Angular - Docs [Електронний ресурс] – Режим доступу: https://angular.io/docs.

26. Express.js - guide [Електронний ресурс] – Режим доступу: https://expressjs.com/en/guide.

27. Introduction to GraphQL [Електронний ресурс] – Режим доступу: https://graphql.org/learn.

28. Civil Research Data. Collection of user-generated online news comments [Електронний ресурс] – Режим доступу: https://figshare.com/articles/dataset/data_json/7376747.

29. Deliver Better Digital Experiences with Composable Architectures – [Електронний ресурс] – Режим доступу: https://hygraph.com/blog/better-digital-experiences-with-composable-architectures

30. What is the Difference Between SPAs, SSGs, and SSR? – [Електронний ресурс] – Режим доступу: https://hygraph.com/blog/difference-spa-ssg-ssr

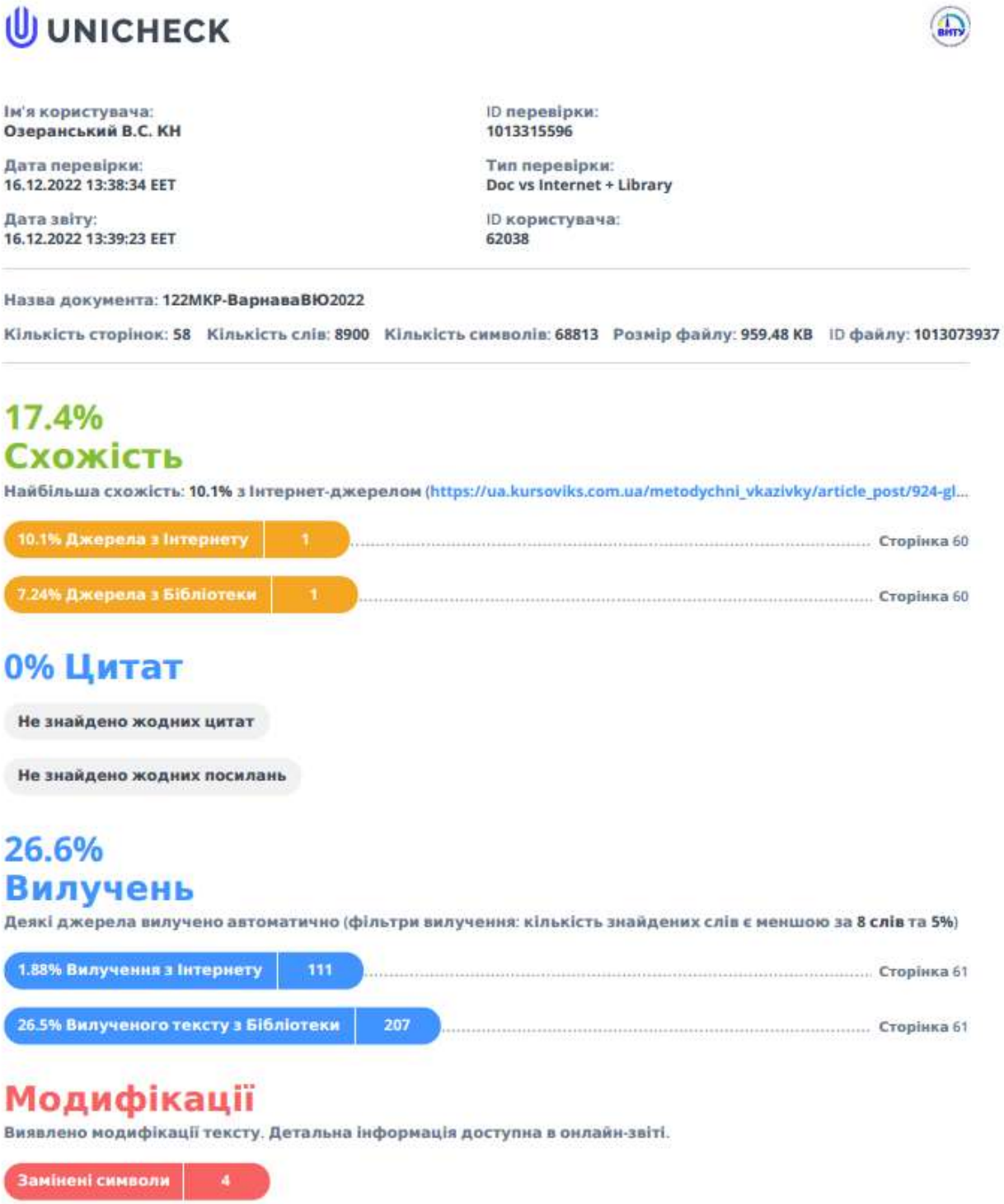
31. Microservices vs. monolithic architecture – [Електронний ресурс] – Режим доступу: https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith

32. Understanding client-side JavaScript frameworks – [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks

ДОДАТКИ

Додаток А

Протокол перевірки МКР на наявність текстових запозичень



UNICHECK

Ім'я користувача:
Озеранський В.С. КН

ID перевірки:
1013315596

Дата перевірки:
16.12.2022 13:38:34 EET

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
16.12.2022 13:39:23 EET

ID користувача:
62038

Назва документа: 122МКР-ВарнаваВЮ2022

Кількість сторінок: 58 Кількість слів: 8900 Кількість символів: 68813 Розмір файлу: 959.48 KB ID файлу: 1013073937

17.4%
Схожість

Найбільша схожість: 10.1% з Інтернет-джерелом (https://ua.kursoviks.com.ua/metodychni_vkazivky/article_post/924-gl...

10.1% Джерела з Інтернету 1 ..... Сторінка 60

7.24% Джерела з Бібліотеки 1 ..... Сторінка 60

0% Цитат

Не знайдено жодних цитат

Не знайдено жодних посилань

26.6%
Вилучень

Деякі джерела вилучено автоматично (фільтри вилучення: кількість знайдених слів є меншою за 8 слів та 5%)

1.88% Вилучення з Інтернету 111 ..... Сторінка 61

26.5% Вилученого тексту з Бібліотеки 207 ..... Сторінка 61

Модифікації

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

Замінені символи 4

Додаток Б

Лістинг програми

```
import {useEffect} from "react";
import firebase from "firebase";
import {useHistory} from "react-router-dom";
import {message} from "antd";

const FirebaseAuth = ({ setUser }) => {
    let history = useHistory();
    useEffect(() => {
        firebase.auth().onAuthStateChanged((user) => {
            if (user) {
                // User is signed in, see docs for a list of available properties
                // https://firebase.google.com/docs/reference/js/firebase.User

                console.log('====> user onAuthStateChanged', user, history)
                setUser(user)

                message.success(`Signed in successfully.`);
                history.push('/')
                // ...
            } else {
                // User is signed out
                // ...
                setUser(null)
                message.success(`Signed out successfully.`);
                history.push('/signin')

            }
        });

    }, [])

    return null
}

export default FirebaseAuth


import {Form, Input, Button, Checkbox, Row, Col, Typography, message} from 'antd';
import {useEffect, useState} from "react";
import firebase from 'firebase'


const { Title } = Typography;

const SignUp = () => {
    const [email, setEmail] = useState()
    const [password, setPassword] = useState()
    const onFinish = (values) => {
        console.log('Success:', values);
```

```
    };

    const onFinishFailed = (errorInfo) => {
        console.log('Failed:', errorInfo);
    };

    const auth = () => {
        firebase.app().auth().createUserWithEmailAndPassword(email, password)
            .then((userCredential) => {
                // Signed in
                const user = userCredential.user;

                firebase.database().ref('users/' + user.uid).set({
                    email: user.email,
                    uid: user.uid,
                });

                localStorage.setItem('token', user.refreshToken);

                // ...
            })
            .catch((error) => {
                var errorCode = error.code;
                var errorMessage = error.message;
                console.log('===> error', error)
                // ..
            });
    }

    return (
        <Row style={{ marginTop: 100 }} justify="center">
            <Col span={10}>
                <Form
                    {...layout}
                    name="basic"
                    initialValues={{
                        remember: true,
                    }}
                    onFinish={onFinish}
                    onFinishFailed={onFinishFailed}
                >
                    <Title>Sign Up</Title>
                    <Form.Item
                        label="Email"
                        name="email"
                        rules={[
                            {
                                required: true,
                                message: 'Please input your username!',
                            },
                        ]}
                    >
                        <Input type={'email'} onChange={e =>
setEmail(e.target.value)} />
                    </Form.Item>
```

```
                    <Form.Item
                        label="Password"
                        name="password"
                        rules={[
                            {
                                required: true,
                                message: 'Please input your password!',
                            },
                        ]}
                    >
                        <Input.Password onChange={e => setPassword(e.target.value)}/>
                    </Form.Item>
                    <Form.Item {...tailLayout}>
                        <Button onClick={auth} type="primary" htmlType="submit">
                            Submit
                        </Button>
                    </Form.Item>
                </Form>
            </Col>
        </Row>
    );
};

export default SignUp

export const FileInput = ({ setFileText }) => {

    const props = {
        name: 'file',
        multiple: false,
        action: 'https://www.mocky.io/v2/5cc8019d300000980a055e76',
        onChange(info) {
            const { status } = info.file;
            if (status !== 'uploading') {
                const reader = new FileReader()
                reader.onload = async (e) => {
                    const text = (e.target.result)
                    setFileText(text)
                };
                reader.readAsText(info.file.originFileObj)
            }
            if (status === 'done') {
                message.success(`${info.file.name} file uploaded successfully.`);
            } else if (status === 'error') {
                message.error(`${info.file.name} file upload failed.`);
            }
        },
        // listType:"picture",
        maxCount:1,
    };


    return (
        <Upload
            {...props}
        >
            <Button icon={<UploadOutlined />}>Upload (Max: 1)</Button>
```

```
        </Upload>
    )



function MainPage() {


    useEffect(() => {

        if (userId) {
            firebase.database().ref('/users/' + userId).once('value').then((snapshot)
=> {
                console.log('===> snapshot.val()', snapshot.val())

                setUser(snapshot.val())
            });
        }

    }, [])

    const getList = () => {
        const list = user?.files && Object.keys(user.files).map(id => ({
            id,
            value: user.files[id].file
        }))

        if (!list) {
            return []
        }
        return list

    }

    const findAnswer = async (question) => {
        const model = await qna.load();
        const answers = await model.findAnswers(question, fileText);
        setAnswers(answers)
    }

    const uploadFile = async () => {

        if (!fileTextToUpload) return null

        const fileId = '_' + Math.random().toString(36).substr(2, 9);

        await firebase.database().ref('users/' + userId + '/files/' + fileId).set({
            file: fileTextToUpload
        });

        firebase.database().ref('/users/' + userId).once('value').then((snapshot) =>
```

Додаток В

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ПОШУКУ ІНФОРМАЦІЇ В ТЕКСТАХ

Виконав: студент 1-го курсу,
групи 1КН-21м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
Варнава В.Ю.
(прізвище та ініціали)

Керівник: д.т.н., доцент каф. КН
Сілагін О.В.
(прізвище та ініціали)
«15» 12 2022 р.

Вінниця ВНТУ - 2022 рік

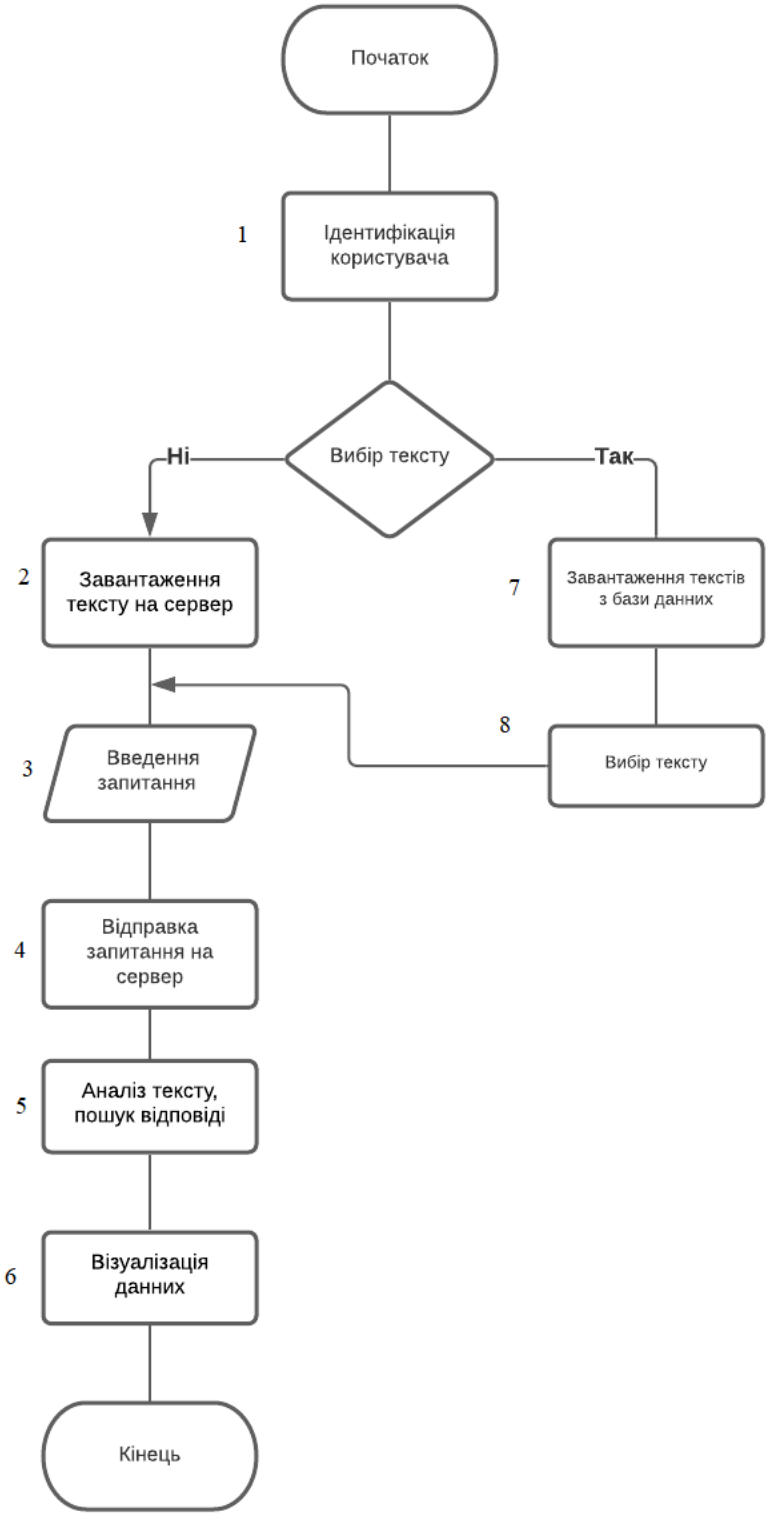


Рисунок В.1 – Алгоритм інформаційної технології роботи інформаційної технології пошуку інформації в текстах

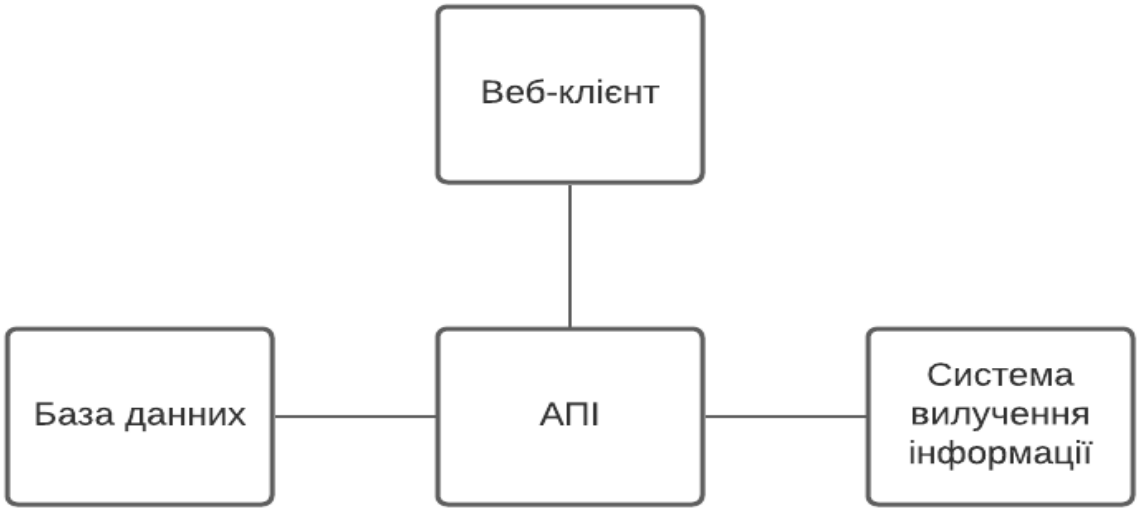


Рисунок В.2 – Загальна структурна схема інформаційної технології пошуку інформації в тексті

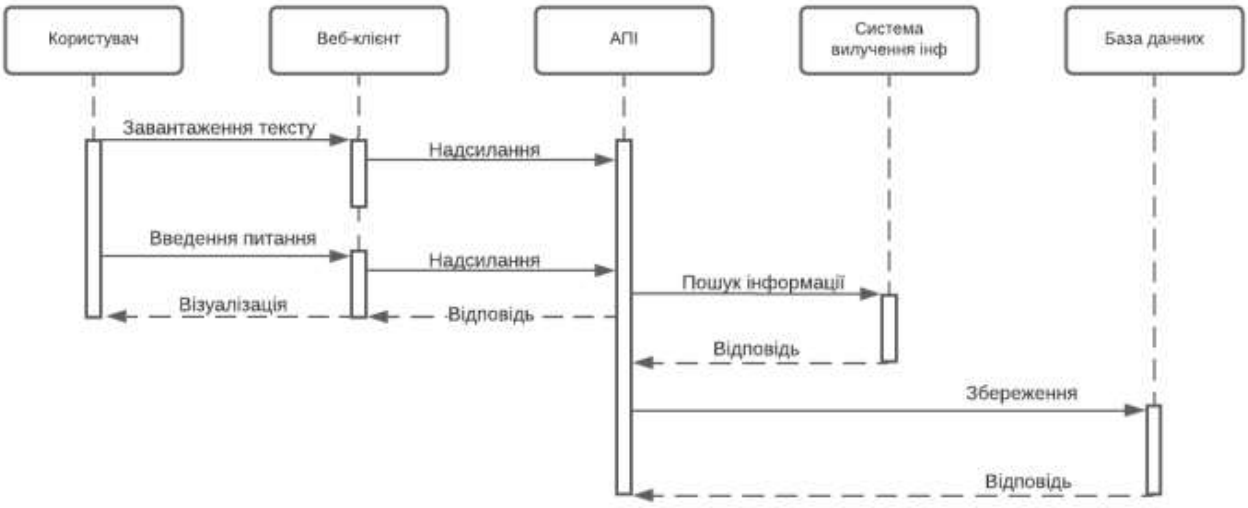


Рисунок В.3 – UML-діаграма послідовностіінформаційної технології пошуку інформації в тексті

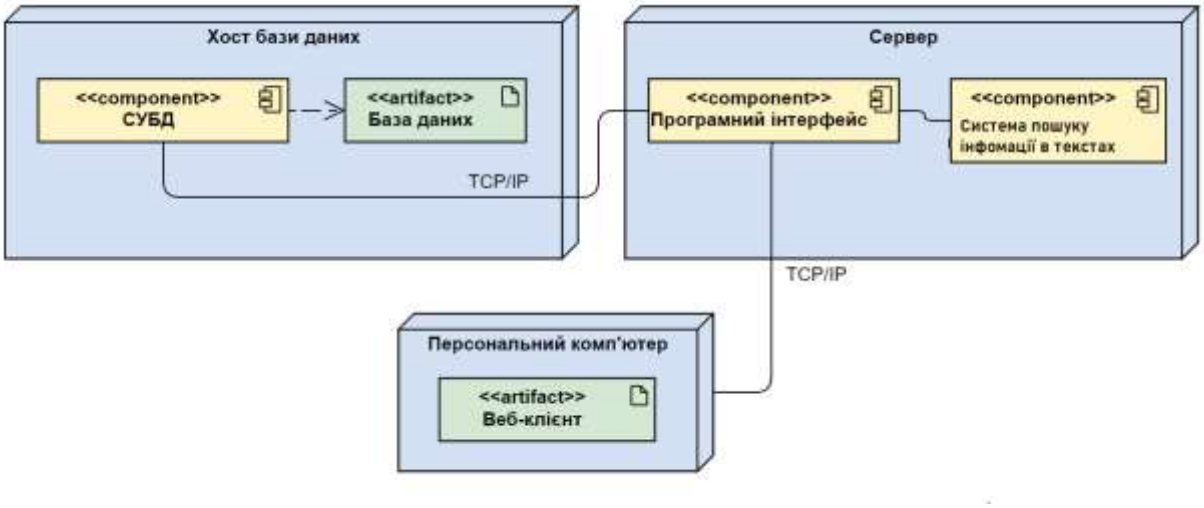


Рисунок В.4 – UML- розгортування послідовностіінформаційної технології пошуку інформації в тексті

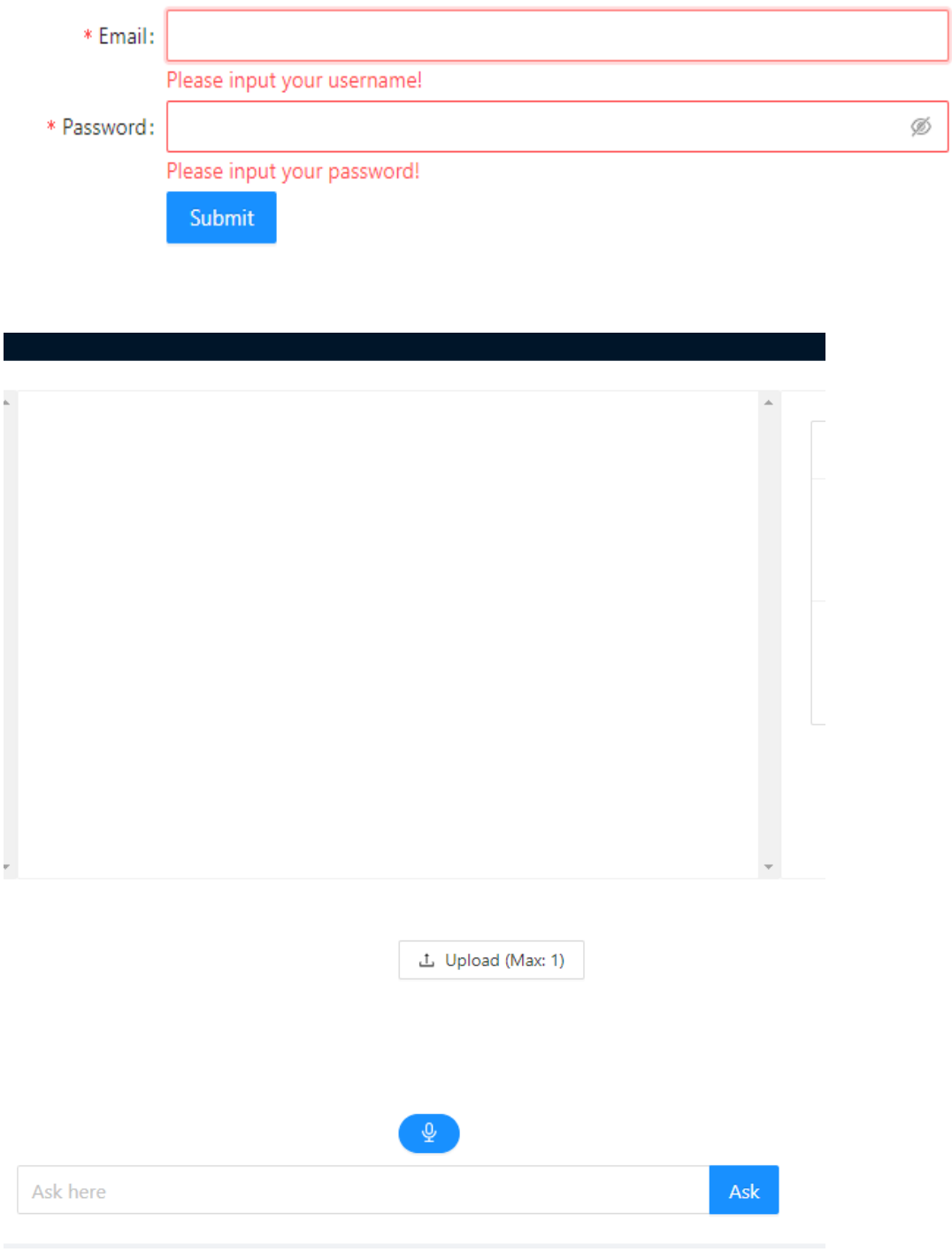


Рисунок В.4 – Робочі вікна інформаційної технології пошуку інформації в тексті

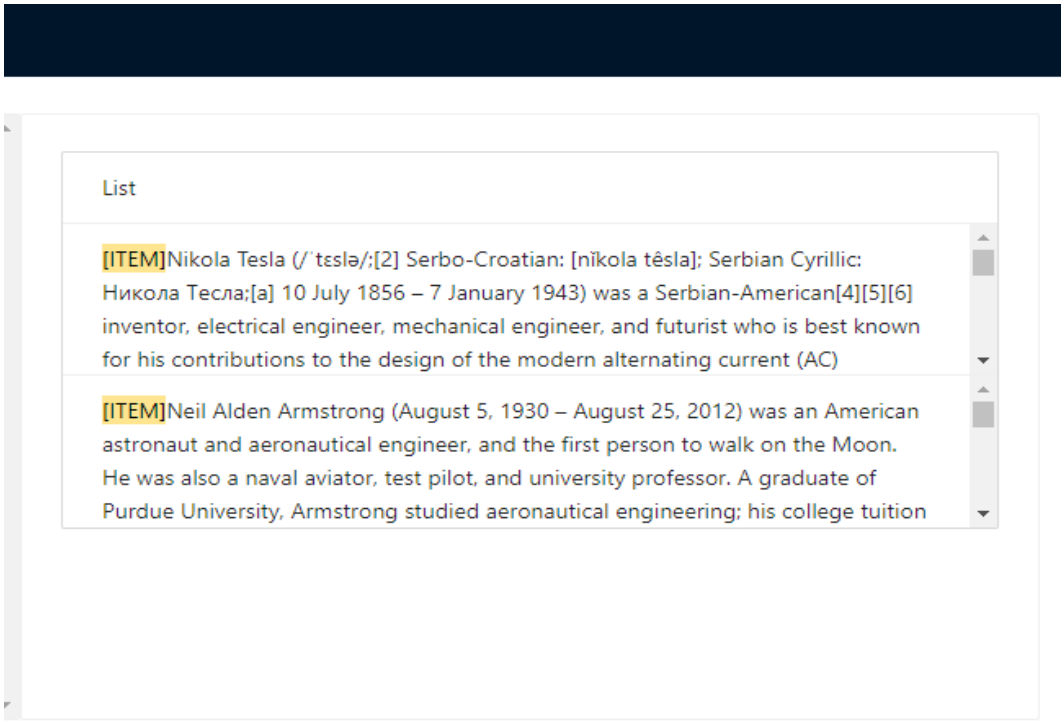


Рисунок В.5 – Результат інформаційної технології пошуку інформації в тексті