Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматики
Кафедра автоматизації та інтелектуальних інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Система пошуку та аналізу відгуків і коментарів»
(тема роботи)

Виконав: студент 2-го курсу гр. ІІСТ-22м
(шифр групи)
спеціальності 126 – Інформаційні системи та технології
(шифр та назва спеціальності)
Вільям ВОЙЦЕХОВСЬКИЙ
(ім'я ПРІЗВИЩЕ студента)
Керівник: к.т.н., доц. каф. АІІТ
Ілона БОГАЧ
(науковий ступінь, вчене звання / посада, ім'я ПРІЗВИЩЕ керівника)
«11» грудня 2023 р.
Опонент: к.т.н., доц.каф. КН
Олексій СІЛАГІН
(науковий ступінь, вчене звання / посада, ім'я ПРІЗВИЩЕ опонента)
«11» грудня 2023 р.

Допущено до захисту
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
«14» грудня 2023 р.

Вінниця ВНТУ – 2023 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій
Рівень вищої освіти II-ий (магістерський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність 126 – «Інформаційні системи та технології»
Освітня програма – Інформаційні технології аналізу даних та зображень.

ЗАТВЕРДЖУЮ
Завідувач кафедри АІІТ
д.т.н., проф. Бісікало О.В.
«20» 09 2023 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Войцеховському Вільяму Вільямовичу
(ПІБ автора повністю)

1. Тема роботи: Система пошуку та аналізу відгуків і коментарів Керівник роботи: к.т.н., доцент кафедри АІІТ І.В.Богач
Затверджені наказом ВНТУ від «13» 09 2023 року № 247.

2. Строк подання студентом роботи до «20» листопада 2023 року.

3. Вихідні дані до роботи: операційна система Manjaro Linux 23.0 і вище; оперативна пам'ять 2 гб; Intel Core i3 і вище; Java, Spring, PostgreSQL, Hibernate, Apache OpenNLP, Apache Lucene, PaLM.

4. Зміст текстової частини: вступ; дослідження існуючих технологій індексації та аналізу інформації та коментарів; обґрунтування вибору технології та методів розробки; проєктування архітектури системи; розробка та тестування системи; економічне обґрунтування; висновки; список використаних джерел.

5. Перелік ілюстративного (або графічного) матеріалу: use-case діаграми для розроблених сценаріїв взаємодій користувача, Блок-схема загальної роботи застосунку, Блок-схема загальної роботи плагіну

1. Консультанти розділів магістерської кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Ілона БОГАЧ, к.т.н., доцент каф. АПТ		
5	Володимир КОЗЛОВСЬКИЙ, к.е.н., професор каф. ЕПВМ		

2. Дата видачі завдання «20» 09 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Дослідження існуючих технологій індексації та аналізу інформації та коментарів	20.09.23-30.09.23	виконано
2	Обґрунтування вибору технологій та методів розробки	30.10.23-05.10.23	виконано
3	Проєктування архітектури системи	05.10.23-15.10.23	виконано
4	Розробка та тестування системи	15.10.23-12.11.23	виконано
5	Економічна частина	12.11.23-18.11.23	виконано
6	Оформлення пояснювальної записки і графічного матеріалу	18.11.23-21.11.23	виконано
7	Попередній захист роботи	21.11.23	виконано
8	Остаточний захист роботи	15.12.23	виконано

Студент _____ Вільям ВОЙЦЕХОВСЬКИЙ
(підпис) (прізвище та ініціали)

Керівник роботи _____ Ілона БОГАЧ
(підпис) (прізвище та ініціали)

ЗМІСТ

АНОТАЦІЯ

УДК 004.457

Войцеховський В.В. Система пошуку та аналізу відгуків і коментарів. Магістерська кваліфікаційна робота зі спеціальності 126 – Інформаційні системи та технології, освітньо-професійна програма – Інформаційні технології аналізу даних та зображень. Вінниця: ВНТУ, 2023. 112с.

На укр. мові. Бібліогр.: 34 назв; рис.: 42; табл.: 2.

У магістерській кваліфікаційній було досліджено підходи пошуку та аналізу комента на мові програмування Java, використовуючи Spring, Hibernate, PostgreSQL та Maven. Зручність системи полягає у можливості детального аналізу обраних коментарів і можлиовсті отримання рекомендацій що до роботи з цим коментарем. Розроблений модуль є гнучким у конфігурації, кроссплатформеним та відкритий для розширень.

Ключові слова: пошук, аналіз, коментарі, графічний інтерфейс, програмне забезпечення, бази даних.

ANNOTATION

Viliam V. V. Search and Analysis System for Reviews and Comments. Master’s qualification paper of a specialty 126 – Informational systems and technologies, educational program – Informational technologies of data and image analysis. Vinnytsia: VNTU, 2023. 112 p.

In Ukrainian language. Bibliography: 34 titles; fig.: 42; tabl.: 8.

In this master's thesis, a system for searching and analyzing comments was developed using the Java programming language, employing Spring, Hibernate, PostgreSQL, and Maven. This system is free, cross-platform, and equipped with a wide array of tools for comment search and analysis, open for expansions, offering flexibility and scalability.

In the master's qualification thesis, approaches to searching and analyzing comments were investigated using the Java programming language, with the utilization of Spring, Hibernate, PostgreSQL, and Maven. The convenience of the system lies in its ability to perform detailed analysis of selected comments and provide recommendations for working with each comment. The developed module is flexible in configuration, cross-platform, and open for extensions.

Keywords: search, analysis, comments, graphical interface, software, databases.

ВСТУП

Актуальність теми. Сучасне суспільство високо цінує та широко використовує інформацію, яка представлена у різних форматах та становить значну частину життя кожної людини [1]. Онлайн чи офлайн, в робочому оточенні, вдома або під час освіти, люди активно взаємодіють з файлами, текстами, зображеннями, відео та іншими формами даних. Цифровий формат даних, який існує менше століття, став необхідною складовою сучасного людського життя. Різноманітність інформації, що охоплює зображення, відео, документи, креслення, файли сирцевого коду та інші формати, є фундаментальною основою для сучасної цивілізації у цифровій епосі [2]. Для опису цих даних стали застосовувати метадані, які включають у себе різноманітні відгуки та коментарі. Ці елементи важливі як у професійній діяльності, так і в повсякденному житті, в освітньому процесі та культурному сприйнятті. Коментарі допомагають краще розуміти потенційні переваги товарів, виражають емоції від певних зображень, враження від місць, особисті враження від сервісів та багато іншого. Це важлива інформація, яка додає до контексту та збагачує розуміння предмета.

Аналіз коментарів дозволяє відслідковувати тенденції, розуміти уподобання та негативне ставлення, вивчати теми, що цікавлять людей у певний часовий проміжок. Робота з коментарями має велике значення як для бізнесу, так і для науки. Однак, актуальність цієї проблеми полягає у складності зручного пошуку та аналізу коментарів з різних джерел. Важливість полягає в тому, що коментарі та відгуки можуть мати різну структуру, вигляд, використовувати різні мови, що потребує їхньої обробки та представлення користувачам у зручному вигляді. Тому проблема ефективного використання коментарів та їх аналізу є вельми актуальною, а саме дослідження цього питання є затребуваним.

Багато веб-сайтів впроваджують засоби аналізу та пошуку по файлам, використовують фільтри, проте ці можливості часто виявляються недостатніми для отримання користувачем потрібної інформації. Коментарі можуть

розташовуватися на інших сайтах з використанням різних систем аналітики або взагалі можуть бути відсутніми. Потрібно мати універсальний та масштабований інструмент, який незалежно від джерела забезпечуватиме користувача необхідною інформацією.

Існують системи зручні у роботі з аналізом та обробкою інформації та коментарів, проте всі ці функції реалізовані окремо у різних системах. Часто доводиться комбінувати використання кількох платформ для роботи з відгуками з різних джерел, що не завжди є зручним для користувачів. Тому актуальність цієї проблеми вкрай висока.

Актуальним є необхідність розширення функціональних можливостей програмного забезпечення для проведення аналізу та пошуку коментарів, що суттєво прискорить та спростить процес аналізу та пошуку коментарів за рахунок використання сучасних технологій.

Мета роботи є розширення функціональних можливостей програмного забезпечення для проведення аналізу та пошуку коментарів. Основним завданням є створення системи аналізу коментарів з гнучкими налаштуваннями, що на відміну від існуючих прискорить та спростить процес аналізу та пошуку коментарів та забезпечить можливість впровадження системи на різних платформах.

Для досягнення поставленої мети потрібно розв'язати такі задачі дослідження:

- Дослідити існуючі технології індексації та аналізу інформації та коментарів; дослідити існуючі системи обробки та аналітики коментарів; провести порівняльний аналіз;
- Порівняти відомі методи аналізу інформації (текстової, зображення тощо);
- Обрати технології та методи розробки;
- Розробити загальну структуру системи для роботи з коментарями та інтерфейс користувача для цієї системи;

- Розробити програмне забезпечення, використовуючи мову. програмування Java;
- Провести тестування системи та порівняння з аналогами.

Об'єктом дослідження є процес аналізу коментарів.

Предметом дослідження є методи та засоби роботи з коментарями з різних джерел.

Методи дослідження: аналіз, класифікація, узагальнення, спостереження та методи представлення результату.

Джерела дослідження - підручники про роботу з даними та файлами їх обробку і зберігання. Також активно використовувались електронні ресурси з прикладами програм та результатами індексації в різних умовах і з використанням різних засобів.

Методи дослідження. У роботі використовуються методи дослідження, а саме аналіз, моделювання, класифікація, узагальнення, спостереження, прогнозування та експерименту; методи передачі даних та методі представлення результату.

Новизна: полягає в тому, що запропонована модифікація програмного забезпечення для проведення пошуку та аналізу коментарів, яка за рахунок використання мови програмування Java та фреймворку Spring разом із мовною моделлю PaLM на відміну від існуючих інструментів дозволяє розширити функціональні можливості програмного забезпечення, а саме надати гнучкість налаштування для різного типу систем, суттєво спростити пошук та аналіз коментарів, надавати інформацію про конкретний коментар на різних платформах та надати детальний опис та аналіз обраного коментаря.

Практична цінність роботи полягає в створенні програмного засобу системи пошуку та аналізу коментарів з використанням сучасних технологій розробки. Зручність даного модулю пошуку та аналізу коментарів значно покращить та пришвидшить збір необхідних даних для різного роду досліджень.

Апробація результатів та публікації. За результатами даної роботи опубліковано доповіді на Всеукраїнській конференції «Молодь в науці:

дослідження, проблеми, перспективи (МН–2022)» [11], «Молодь в науці: дослідження, проблеми, перспективи (МН–2023)» [29] та на LI Науково–технічній конференції підрозділів Вінницького національного технічного університету ВНТУ (м. Вінниця 2022) [6, 10]. Основний модуль роботи прийнятий на отримання свідоцтва авторського права на твір.

1 ДОСЛІДЖЕННЯ ІСНУЮЧИХ ТЕХНОЛОГІЙ ІНДЕКСАЦІЇ ТА АНАЛІЗУ ІНФОРМАЦІЇ ТА КОМЕНТАРІВ

Аналіз коментарів є складною задачею, для якої на сьогодні не існує універсального програмного забезпечення. Часто програми для цього призначення є спрямованими лише на конкретну сферу або створюються як частина веб-сайтів для їх власних потреб. Отже, необхідно розглядати два різних аспекти: індексацію та аналіз коментарів.

У сучасному світі існує багато різних систем індексації з різним функціоналом та ціновою політикою для кінцевого користувача. Проте, деякі з цих систем можуть не підходити через обмежену кросплатформеність або підтримку конкретних операційних систем. Часто виникає ситуація, коли для вирішення конкретних завдань доводиться одночасно використовувати декілька таких інструментів.

Оскільки відсутній загальний інструмент, який би поєднував індексований пошук коментарів та їх аналіз, доцільно розглянути системи для індексації та платформи, які мають вбудовані інструменти для роботи з коментарями. Такий підхід дозволить знайти оптимальне поєднання різних інструментів для вирішення відповідних завдань з аналізу та обробки коментарів.

1.1 Огляд існуючих систем індексації та аналітики інформації

Індексація є важливим етапом в роботі з інформацією, а у даному випадку з коментарями. Вона надає кілька ключових переваг:

- Швидкий пошук: індексація дозволяє створювати оптимізовані структури даних, які значно полегшують та прискорюють процес пошуку інформації. Замість перегляду всього набору даних, користувач може швидко звертатися до індексу для знаходження потрібної інформації. В ситуації даної системи, ця властивість стане корисною при пошуку серед коментарів, що спростить роботу із ними.

- Організація даних: індексація сприяє структуруванню та організації великих обсягів даних. Це полегшує розуміння зв'язків між різними елементами інформації, що робить процес використання даних більш систематизованим. Це дозволить мати чітку організованість серед великої кількості коментарів.

- Ефективне використання ресурсів: індексація дозволяє оптимізувати використання обчислювальних та мережевих ресурсів. Запити до вже індексованих даних можуть бути оброблені швидше, що зменшує час витрачений на пошук та обробку інформації, а значить і покращить користувацький досвід у використанні системи.

- Підтримка аналізу та прийняття рішень: індексація забезпечує можливість ефективного аналізу та використання даних для прийняття рішень. Індексація покращить можливості аналітики коментарів.

- Підтримка найпопулярніших форматів даних, а саме pdf та docx [3, 4].

Отже, індексація вважається ключовим елементом для створення ефективних та швидких систем обробки та використання інформації. Цей процес становить основу для подальшого розвитку та проведення інтелектуального аналізу даних.

У наступних розділах буде проведено аналіз найпопулярніших платформ для індексації, щоб визначити їхні особливості, переваги та можливості у контексті створення та оптимізації систем обробки інформації.

1.1.1 OpenKM

OpenKM є програмним забезпеченням для управління контентом підприємства, що часто відносять до систем управління документами (DMS). Ця платформа дозволяє підприємствам керувати виробництвом, зберіганням, управлінням та розподілом електронних документів, що сприяє підвищенню ефективності та можливості повторного використання інформації, а також контролю потоку документів.

Відповідно до рисунку 1.1, OpenKM має веб-версію програмного забезпечення, що у певній мірі вирішує проблему кросплатформеності. Проте для роботи цієї системи необхідний доступ до Інтернету. Подані нижче рисунки (1.2 та 1.3) вказують на значну кількість функцій та інструментів у OpenKM, проте інтерфейс може виглядати перенавантаженим та містити додаткові вікна.

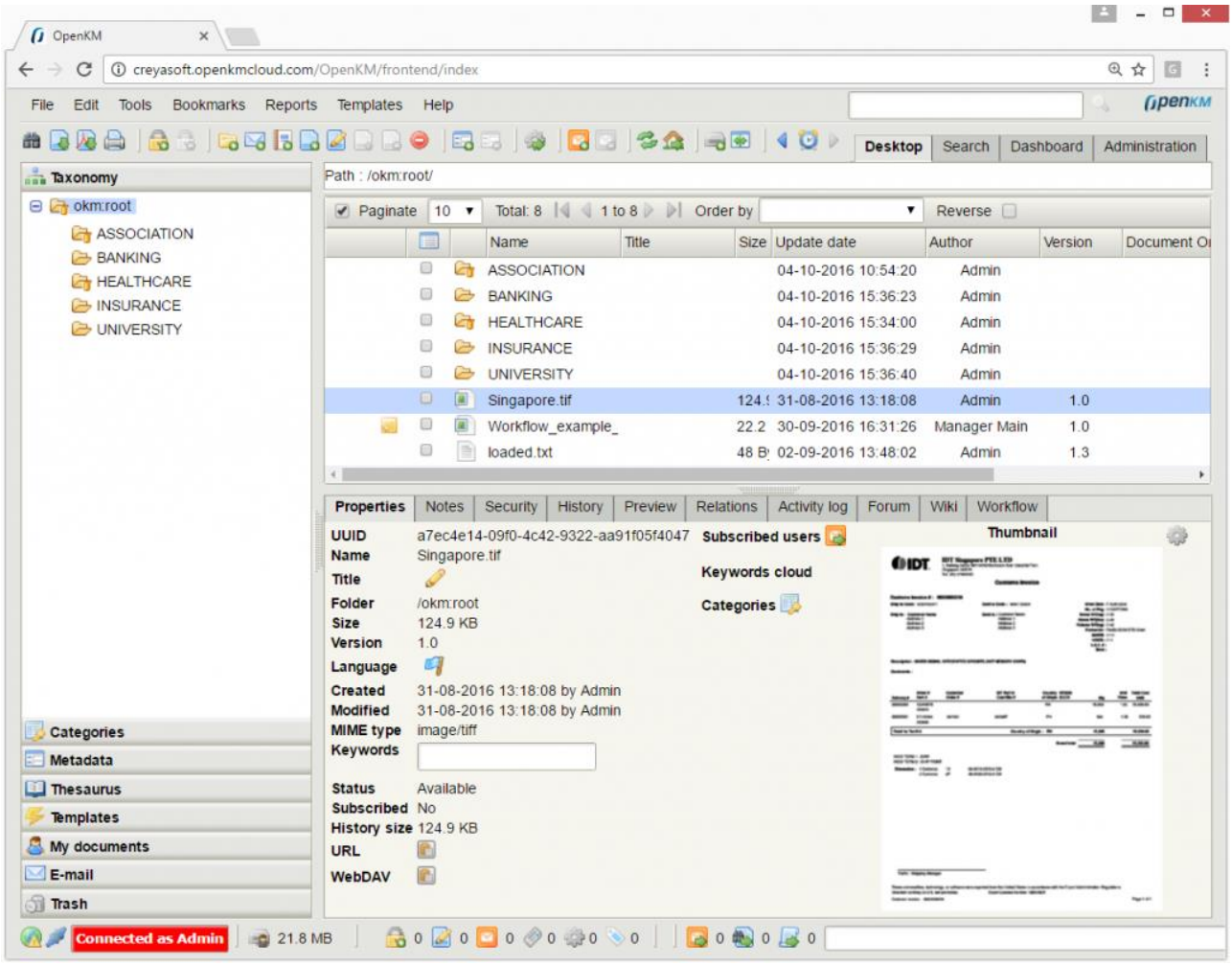

Рисунок 1.1 – Графічний інтерфейс OpenKM

Програмне забезпечення для керування документами OpenKM створює значущий репозиторій корпоративних інформаційних ресурсів з метою сприяти формуванню знань та поліпшенню бізнес-процесів. Це сприяє підвищенню ефективності робочих груп, збільшенню продуктивності підприємств через участь у спільних практиках, покращенню взаємозв'язків з клієнтами,

прискоренню циклів продажів, покращенню конкурентоздатності на ринку продукції та удосконаленню процесів прийняття управлінських рішень [5].

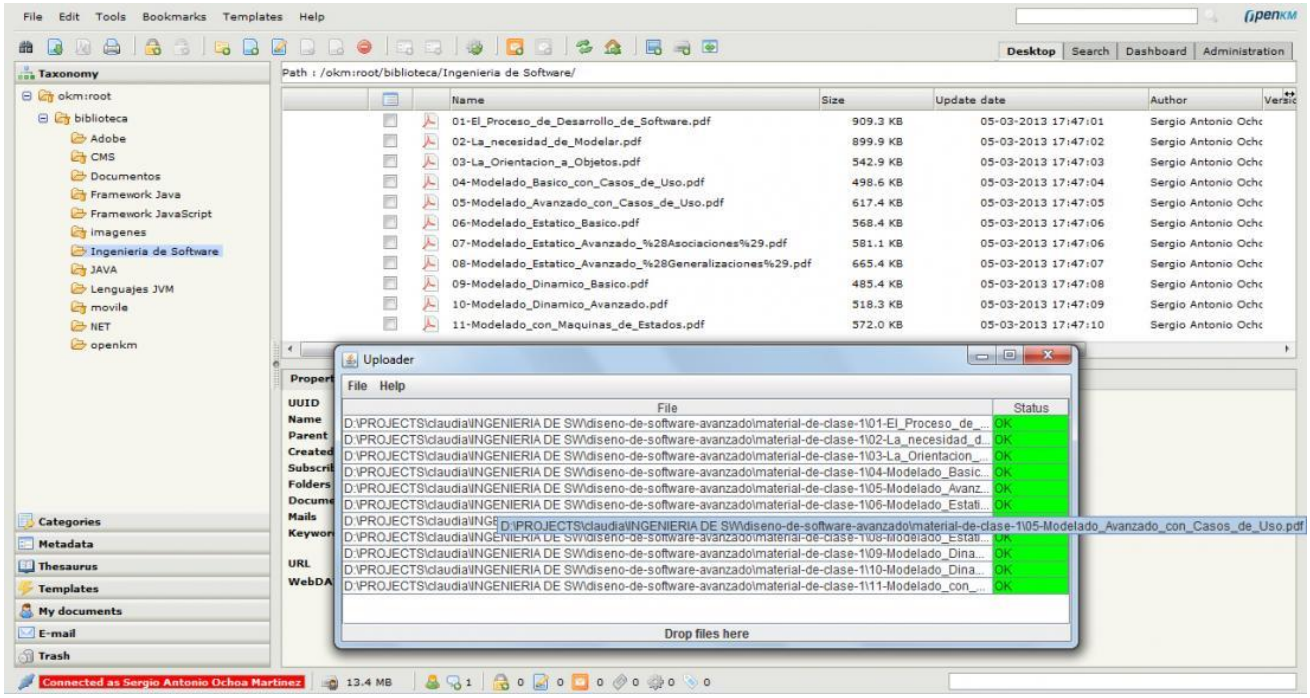

Рисунок 1.2 – Процес індексування в OpenKM

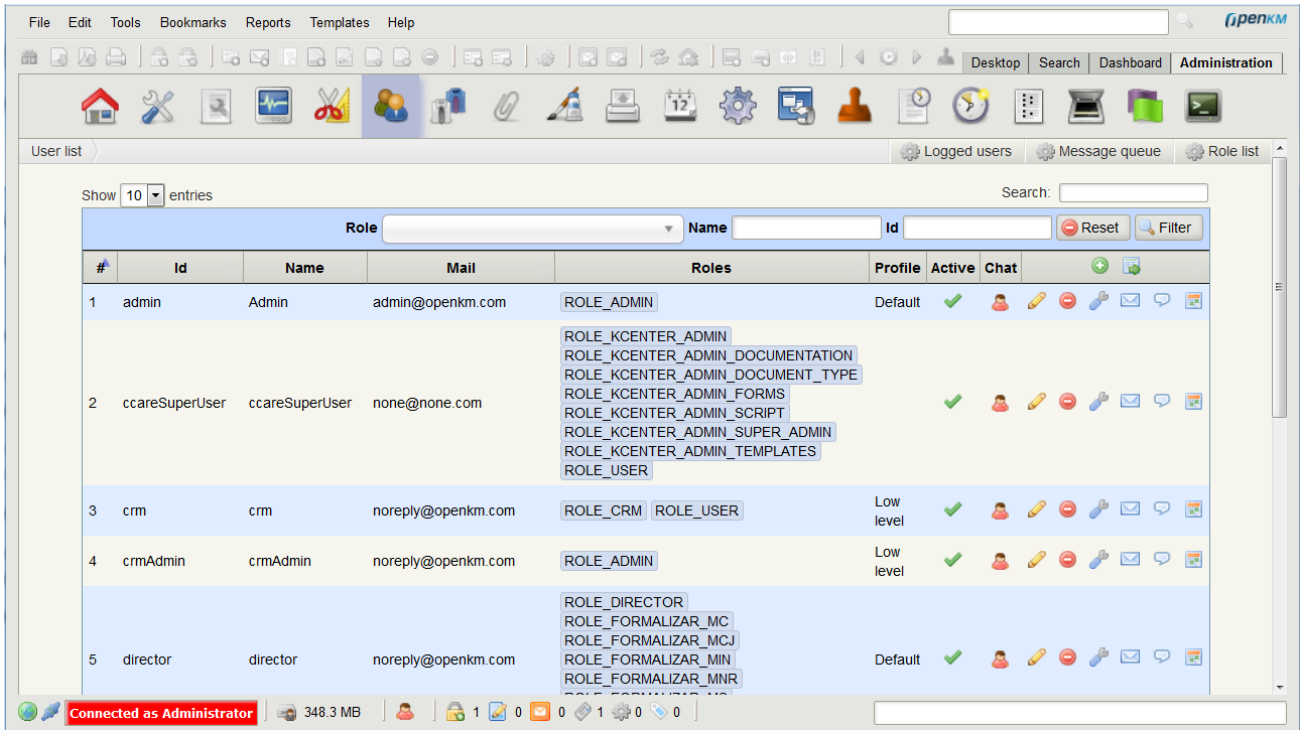

Рисунок 1.3 – Панель адміністратора в OpenKM

Переваги системи:

- Обширний функціонал.
- Можливість індивідуальної настройки окремих компонентів.
- Розгалужені можливості та зручні функції для великих проектів з численною кількістю співробітників.
- Підтримка користувачів.
- Постійне оновлення.

Недоліки системи:

- Надто громіздка та розширена система.
- Складний та перенавантажений інтерфейс користувача.
- Зручна лише для великих проектів.
- Безкоштовна версія має значні обмеження порівняно з платною професійною версією.

Згідно з проведеними дослідженнями, система виявилася досить складною для розуміння користувачами без достатнього досвіду та можливо не є оптимальним вибором для особистого використання. Також варто відзначити, що безкоштовна версія програми не надає всіх переваг, які має платна професійна версія.

SimpleSearch входить до всіх версій SimpleIndex та може бути окремо ліцензованим. Програма SimpleSearch відображає інтерфейс SimpleIndex у режимі «Вилучення», приховуючи всі меню та панелі інструментів, що зазвичай використовуються під час сканування. Замість цього вона використовує поля індексу для пошуку, які зазвичай призначені для введення значень.

За допомогою рисунка 1.4 можемо оцінити графічний інтерфейс програми SimpleSearch. Його зовнішній вигляд має деякі схожості з Microsoft Word, що робить його більш придатним для операційної системи Windows. Крім того, інтерфейс відображається досить зручно та інтуїтивно зрозуміло. Однак, його застарілість може вважатися основним недоліком.

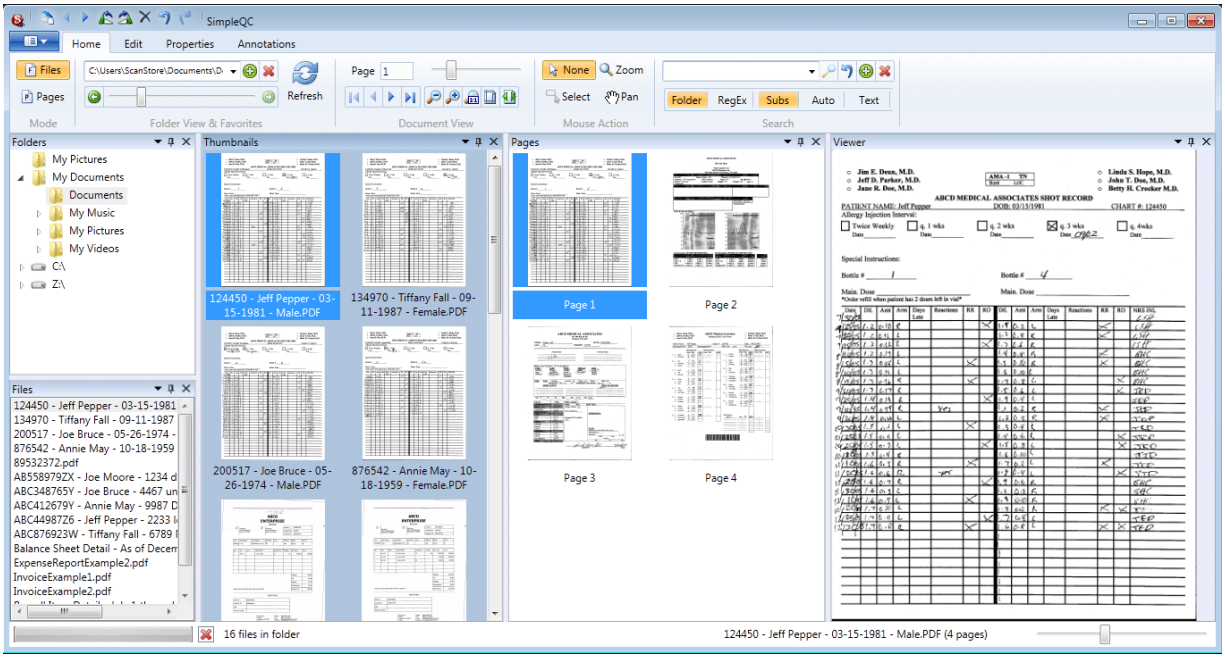

Рисунок 1.4 – Графічний інтерфейс SimpleSearch

В одному з функціональних елементів SimpleSearch, відображеному на рисунку 1.5, передбачено можливість зручного перегляду різних сторінок одного документа.

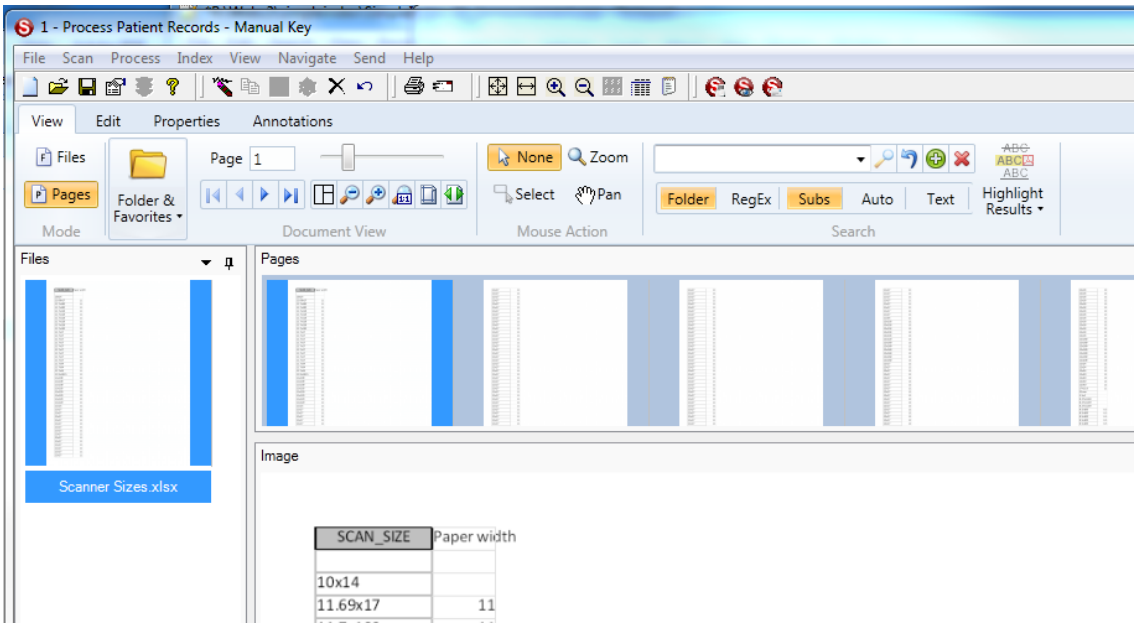

Рисунок 1.5 – Перегляд сторінок документу в SimpleSearch

Як ілюструється на рисунку 1.6, процес індексування виконується в тому ж вікні, де працює програма. Проте, відзначається недолік у візуальному

представленні, що призводить до неестетичного вигляду та ускладнює спостереження за процесом індексації.

1.1.2 SimpleSearch

SimpleSearch має можливість використовувати вбудовану базу даних для здійснення пошуку або підключення до будь-якої іншої бази даних, включаючи існуючі бізнес-додатки. Користувачі можуть легко вводити значення індексів, які їм потрібно знайти, і SimpleIndex відображає відповідності. Пошук підтримує як часткову відповідність, так і повнотекстовий пошук. Після знаходження документів користувачам надається можливість роздрукувати їх, відправити електронною поштою або відкрити у відповідних програмах, таких як Word, Adobe Reader, Excel та інші. Це спрощує процес обробки знайдених документів та їх подальшого використання залежно від потреб користувача.

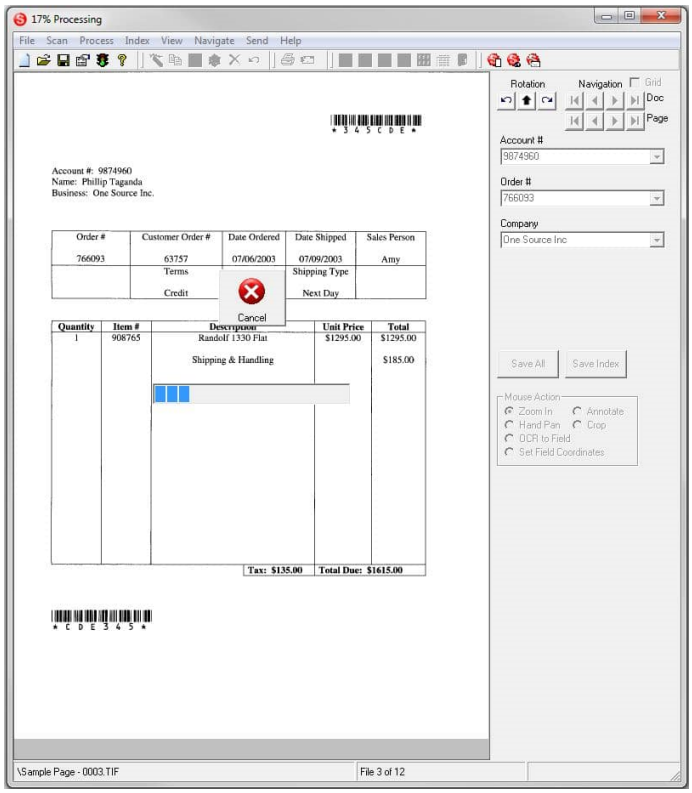

Рисунок 1.6 – Процес індексації в SimpleSearch

За допомогою SimpleSearch можна преглядати різноманітні формати зображень і файли у форматі PDF, а також може здійснювати попередній перегляд документів для будь-якої програми, яка підтримує OLE (наприклад, MS Office, програми Adobe, AutoCad та інші) [6].

Переваги системи включають:

- Розширена підтримка різноманітних форматів документів.
- Підтримка користувачів.
- Регулярні оновлення.

Однак система має декілька недоліків:

- Громіздкість
- Старомодний інтерфейс користувача.
- Обмеження лише під платформу Windows.
- Відсутність безкоштовної версії. Доступний лише безкоштовний пробний період.

Дане дослідження показало, що серед недоліків системи є застарілість інтерфейсу, громіздкість, обмеженість тільки під однією платформою і відсутність безкоштовної версії.

1.1.3 LogicalDOC

LogicalDOC має ключову особливість - повнотекстове індексування всіх документів з метою надання миттєвих результатів пошуку на основі вмісту файлів і метаданих. Важливо зазначити, що це програмне забезпечення доступне лише у веб-версії, що частково вирішує питання кросплатформеності. Проте це також означає необхідність доступу до Інтернету для його функціонування.

Аналізуючи рисунки 1.7, 1.8 та 1.9, можна зауважити, що у LogicalDOC присутній сучасний та естетичний інтерфейс, що відображається в його дизайні.

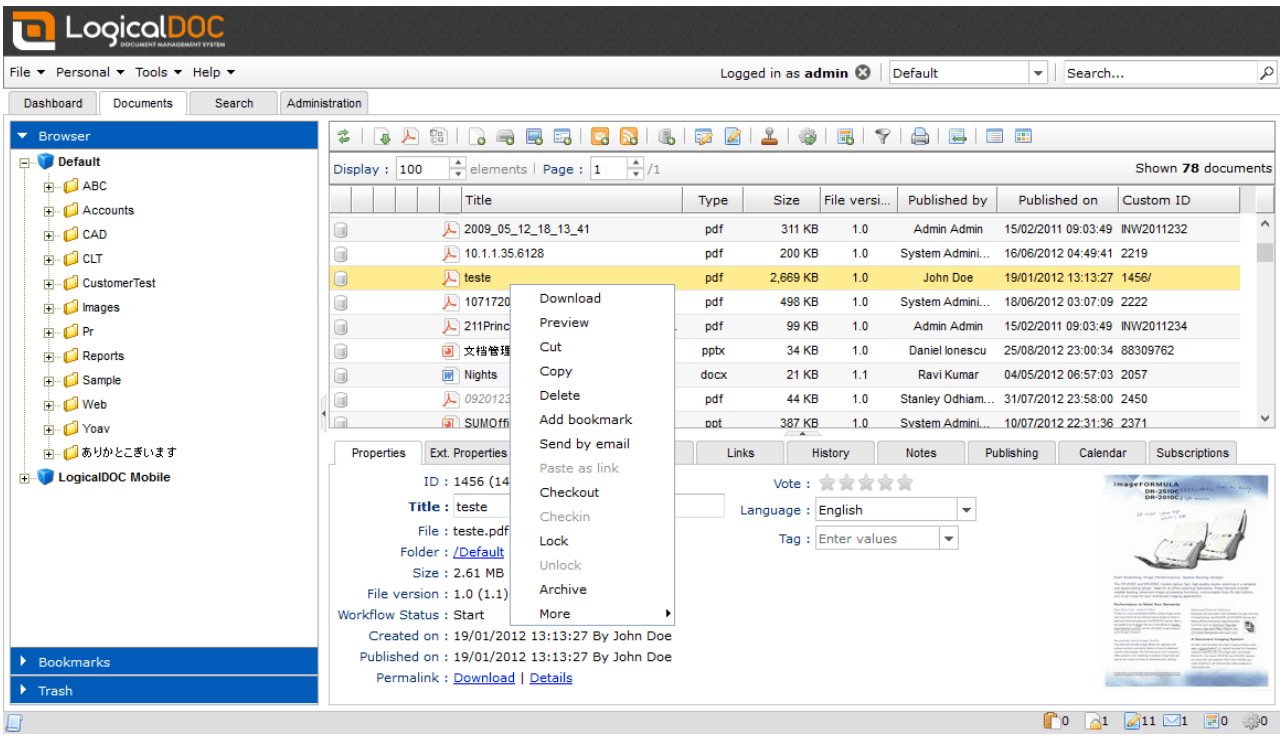

Рисунок 1.7 – Графічний інтерфейс LogicalDOC

На основі даних, поданих на рисунку 1.8, можна спостерігати за існуванням функції, яка відображається як досить зручна, оскільки дозволяє вибирати одночасно кілька або всі документи, що спрощує процес роботи з програмою.

Поверхневий аналіз інтерфейсу на рисунку 1.9 вказує на ряд переваг. Наприклад, присутня інформаційна панель індексації, яка містить список індексованих документів та відображає позначки, що свідчать про завершення процесу індексації. Також пристутній індикатор процесу, який відображає час, що залишився для завершення індексації файлу. Ця функціональність корисна для уявлення часових рамок, необхідних для завершення процесу індексації.

В рамках аналізу властивостей LogicalDOC варто звернути увагу на його здатність автоматичного індексування повного змісту документів, які знаходяться в його сховищі. Оптимізація продуктивності й підвищення рівня паралельності досягаються за допомогою асинхронного процесу індексування з використанням налаштованої політики планування. У цьому контексті варто зазначити, що в залежності від мови документа використовуються різні

алгоритми, що робить пошук відповідно адаптованим та здатним визначати варіанти слів, які характерні для конкретної мови [7].

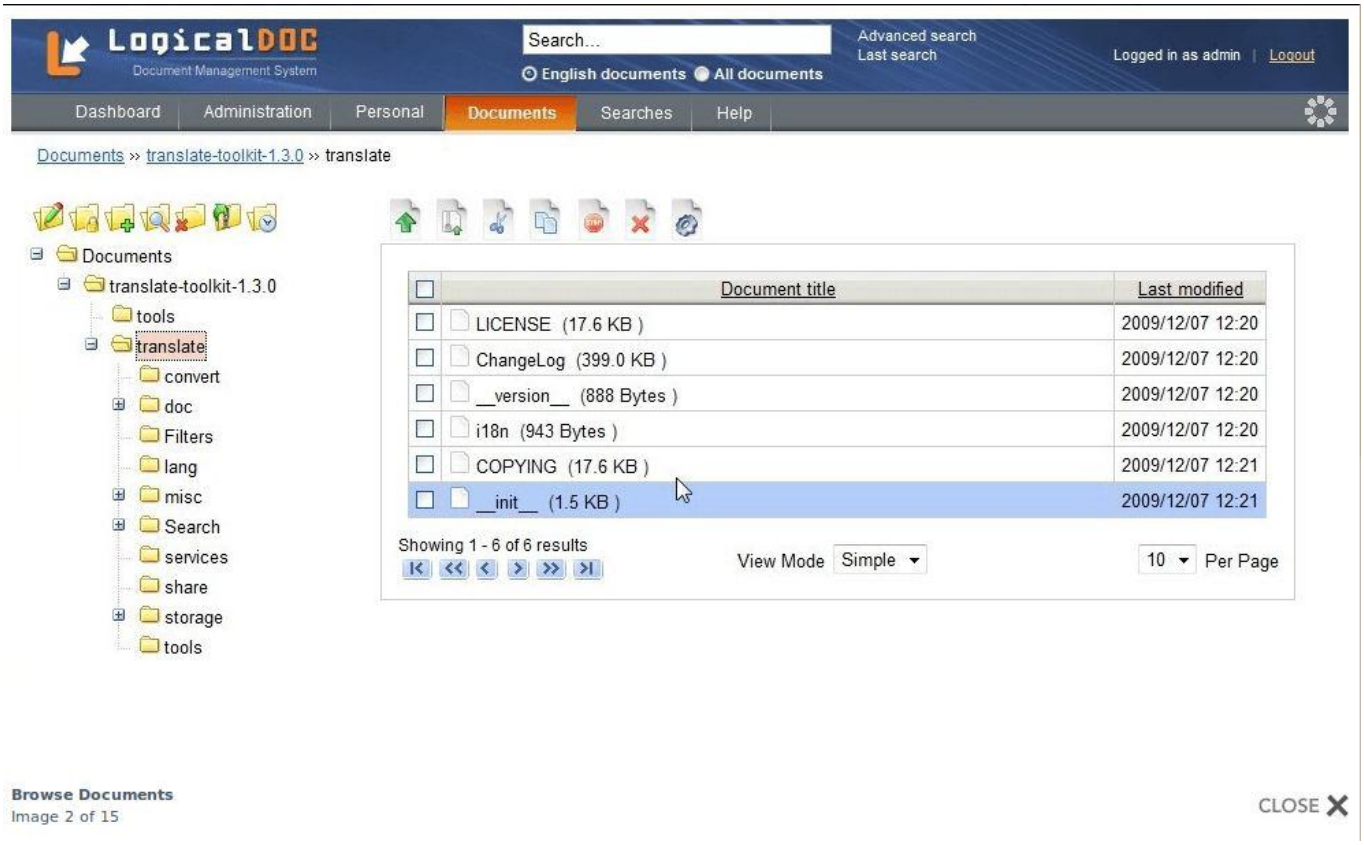

Рисунок 1.8 – Меню вибору декількох документів в LogicalDOC

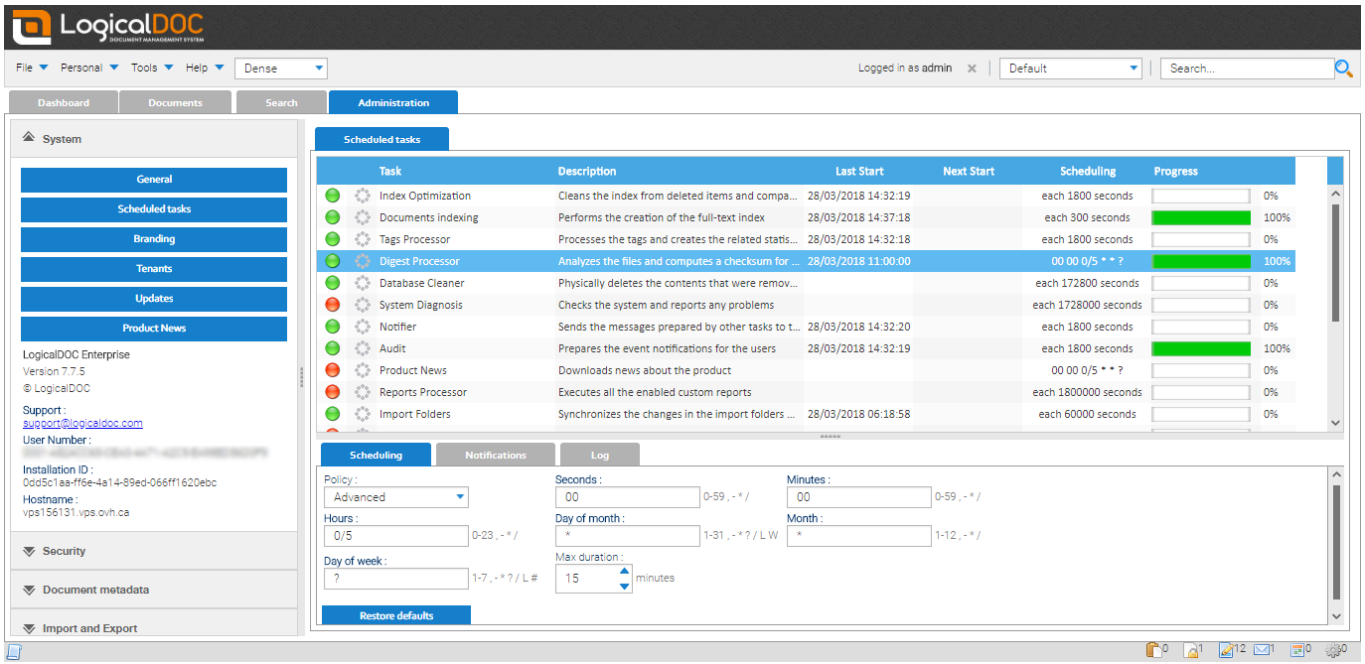

Рисунок 1.9 – Процес індексації LogicalDOC

Переваги системи LogicalDOC:

- Адаптація до змісту документів та їх мови.
- Висока швидкість та ефективність роботи при обмежених ресурсах.
- Наявність на популярних платформах.
- Сучасний та зрозумілий інтерфейс.

Недоліки системи LogicalDOC:

- Необхідність пройти процес реєстрації для отримання програми.
- Відсутність безкоштовної версії, за винятком пробного періоду.

За результатами проведених аналізів системи можна відзначити, що серед недоліків присутні відсутність безкоштовної версії та необхідність проходження реєстрації для отримання програмного продукту.

1.2 Огляд існуючих систем обробки та аналітики коментарів

Коментарі грають важливу роль у процесі прийняття рішень покупцями, діючи як вагомий механізм для поліпшення якості товарів і надання клієнтам якісного обслуговування. Аналіз коментарів стає невід'ємним інструментом для сприйняття відгуків клієнтів стосовно продуктів чи послуг і оптимізації бізнес-процесів. Цей аналітичний інструмент дозволяє здійснити оцінку настроїв суспільства та виявити актуальні тенденції, які привертають увагу споживачів через маркетплейси, соціальні мережі та інші платформи. Нижче наведено деякі особливості аналізу коментарів на маркетплейсах, соціальних мережах та інших платформах:

- Відслідковування трендів: аналіз коментарів дозволяє вам виявляти популярні теми серед людей. Можна виявити часті позитивні або негативні тенденції і реагувати на них.

- Оцінка задоволеності: спостереження за ступенем задоволеності людей за їхніми коментарями може допомогти вам зрозуміти, наскільки ефективні продукти, обслуговування, реклама чи якісь події.

- Виявлення проблем: шукаючи конкретні ключові слова або фрази, ви можете виявити можливі проблеми або слабкі місця у продуктах, обслуговуванні, рекламних кампаніях тощо.

- Покращення продуктів, обслуговування, контенту, реклами, маркетингу: аналіз коментарів надає можливість збирати конструктивний фідбек, який може бути використаний для вдосконалення продуктів чи процесів обслуговування.

- Взаємодія з користувачами: реагуючи на коментарі, можна покращити взаємодію з користувачами, надаючи додаткову інформацію, вирішуючи проблеми або висловлюючи подяку за позитивні коментарі.

- Маркетинг та реклама: Позитивні коментарі та відгуки можуть служити основою для маркетингових кампаній. Є можливість використовувати ці відгуки у рекламі для підвищення довіри клієнтів.

- Порівняння з конкурентами: аналіз коментарів та відгуків порівнюючи з коментарями та відгуками конкурентів може допомогти визначити конкурентні переваги та недоліки в послугах чи контенті.

- Пошук можливостей для розширення: в коментарях можуть виявлятися нові ідеї або запити від користувачів, які можуть вказувати на можливості для розширення продуктів, послуг чи контенту. Збір та аналіз цих даних може вимагати використання спеціальних інструментів для моніторингу соціальних мереж чи аналізу тексту.

Отже, відгуки та коментарі відіграють значну роль у сприйнятті продуктів та послуг користувачами. Деякі платформи надають можливість використовувати ці коментарі для вдосконалення своїх послуг, товарів або контенту. Також можна звернути увагу як виглядають коментарі на різних платформах і які особливості мають. Розглянемо деякі з таких платформ.

1.2.1 Платформа Repustate

Платформа Repustate спеціалізується на обробці коментарів у соціальних мережах, використовуючи ряд інструментів для аналізу природної мови (NLP) та надання корисних аналітичних даних. Це дозволяє користувачам отримувати інформацію про актуальні тенденції.

Repustate пропонує різноманітні інструменти та рішення для опрацювання текстів у вигляді природної мови (NLP), спрямовані на допомогу підприємствам у видобутті цінної інформації з неструктурованих текстових даних [8]. Деякі ключові аспекти, пов'язані із платформою Repustate, включають:

- Аналіз настроїв (Sentiment Analysis): Repustate дозволяє визначати тон та настрій тексту, допомагаючи бізнесам зрозуміти, як сприймають їхні клієнти або користувачі.

- Текстовий аналіз: Платформа надає засоби для аналізу текстової інформації, включаючи розпізнавання ключових слів, екстракцію іменованих сутностей та інші аспекти обробки тексту.

- Мовні моделі та API: Repustate може надавати мовні моделі та програмні інтерфейси (API), які дозволяють інтегрувати їхні функціональності в різноманітні додатки та системи.

- Аналіз соціальних мереж: Здатність використовувати Repustate для моніторингу та аналізу відгуків у соціальних мережах, оглядів продуктів, коментарів тощо.

Приклад інтерфейсу зображений на рисунку 1.10.

Узагальнюючи, дана платформа виступає як ефективний засіб для аналізу, однак вона обмежена універсальністю, коли йдеться про аналіз текстів, зокрема коментарів. Це програмне рішення більш підходить для загальної оцінки настроїв у суспільстві та рекламних цілей, ніж для детального аналізу коментарів. Важливо відзначити, що найнижча вартість підписки, яку запропонувала компанія – це 199 доларів на місяць, є відносно високою, особливо для індивідуального використання або малих груп користувачів.

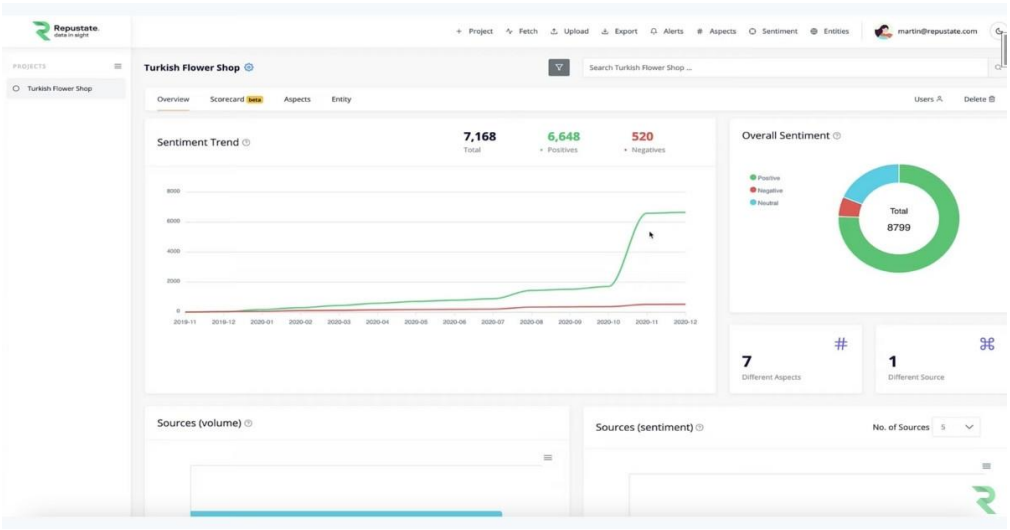

Рисунок 1.10 – Інтерфейс Repustate

1.2.2 Робота з коментарями на сайтах Rozetka, Hotline, Amazon

На основі популярних маркетплейсів можна розглянути методику обробки та візуалізацію аналітики коментарів, хоча самі ці платформи не спеціалізуються на функціоналі для пошуку та аналізу відгуків. Вони пропонують обмежені можливості для роботи з коментарями, зокрема щодо певних товарів чи магазинів, та можуть слугувати прикладом способів, методів та візуалізації аналітики відгуків. Цей тип коментарів допомагає встановлювати загальні тенденції та популярність конкретних товарів або магазинів серед користувачів.

"Rozetka" - це найбільший інтернет-магазин в Україні, який пропонує широкий асортимент товарів, включаючи електроніку, побутову техніку, товари для дому, моду, косметику, і багато інших категорій. Заснований у 2005 році, Rozetka став популярним місцем для онлайн-шопінгу серед українців. Магазин відомий своєю надійністю та репутацією як одного з провідних маркетплейсів в електронній торгівлі в Україні [9]. Він пропонує широкий вибір товарів від різних брендів та часто проводить різноманітні акції та знижки для своїх клієнтів. Способи оплати та доставки різноманітні, що робить покупки більш зручними для користувачів.

Hotline - це інтернет-платформа в Україні, яка надає інформацію про товари та послуги, а також порівнює їх ціни в різних магазинах. Основною

функцією Hotline є створення зручної системи для споживачів, які шукають інформацію перед покупкою.

Основні характеристики та функції Hotline включають:

1. Порівняння цін: Hotline надає можливість порівнювати ціни на товари в різних інтернет-магазинах, що допомагає покупцям знайти найвигіднішу пропозицію.

2. Відгуки та оцінки: Користувачі можуть залишати відгуки та оцінки товарів, що допомагає іншим споживачам приймати інформовані рішення.

3. Каталог товарів: Hotline пропонує широкий каталог товарів, включаючи електроніку, побутову техніку, одяг, взуття та інші категорії.

4. Новини та рецензії: Платформа також надає новини, статті та рецензії про нові товари та технології.

Hotline допомагає споживачам зекономити час і гроші, дозволяючи їм знаходити оптимальні пропозиції на ринку.

Amazon – це міжнародна торговельна компанія та одна з найбільших онлайн-платформ у світі. Заснована в 1994 році Джеффом Безосом, Amazon спочатку стартувала як книжковий інтернет-магазин, а згодом розширила свою діяльність, ставши однією з найрізноманітніших онлайн-платформ для покупок [10].

Основні характеристики та послуги Amazon включають:

- Онлайн-Маркетплейс: Amazon пропонує величезний асортимент товарів, включаючи електроніку, одяг, книги, продукти харчування, побутову техніку та багато іншого. Крім того, компанія надає можливість покупцям та продавцям взаємодіяти на їхньому маркетплейсі.

- Amazon Prime: Сервіс Amazon Prime надає підписникам ряд переваг, таких як безкоштовна доставка для багатьох товарів, доступ до великої бібліотеки стрімінгового відео та музики, а також ексклюзивні акції та пропозиції.

- Amazon Web Services (AWS): Компанія є провідним провайдером хмарних послуг, надаючи широкий спектр послуг інфраструктури, аналітики, штучного інтелекту та багато іншого для бізнесу та розробників.

- Kindle: Amazon виробляє і продає сучасні електронні книги, так звані "Kindle", із доступом до широкого асортименту електронних книг.

Amazon відомий своєю акцентованістю на зручність, широкий вибір товарів та послуг, а також інноваційність у великій кількості сегментів електронної комерції і технологій.

Сайти, такі як Rozetka, Hotline, Amazon та інші, пропонують деякі можливості для роботи з коментарями через їхній веб-інтерфейс. Нижче наведено загальні можливості, які часто доступні на цих майданчиках:

- Фільтрація та сортування: зазвичай існує вкладка "відгуки" або "коментарі" на деяких маркетплейсах можна використовувати різні фільтри та параметри сортування. Наприклад, можна вибрати відгуки з певного рейтингу, переглянути тільки позитивні або негативні відгуки тощо. Можна використовувати поле пошуку, щоб знайти конкретні слова чи фрази у відгуках. Це допомагає швидко виявляти відгуки, що містять конкретні терміни або вирази.

- Категорії та продукти: відгуки можна фільтрувати за категоріями товарів чи конкретними продуктами. Це дозволяє швидко оцінити реакцію користувачів на певну тему чи категорію.

- Відповіді на відгуки: на коментарі можуть бути відгуки, їх можна розглядати в якості додаткових даних.

Варто відзначити, що Rozetka та Amazon пропонують одні з найбільш розвинутих інструментів для пошуку та аналізу коментарів, де користувач може вибрати відгуки за певною оцінкою. Amazon надає можливість вибирати відгуки за ключовими словами або фразами, що значно спрощує пошук. На Hotline також присутня ефективна візуалізація коментарів. Додатково, на всіх цих маркетплейсах можна помітити, що кожен коментар має свій рейтинг серед інших коментаторів, що полегшує оцінку важливості самого відгуку.

Приклади інтерфейсів та можливостей роботи з коментарями проілюстровані на рисунках 1.11, 1.12, 1.13 та 1.14.

Хоча функціонал можна розширити за допомогою властивостей браузера для пошуку по тексту, це не завжди є зручним для користувачів, оскільки має значні обмеження. Такий функціонал застосовується в основному для оцінки товарів або магазинів тільки на одному з маркетплейсів і не може бути використаний для інших коментарів, окрім існуючих для конкретного товару чи магазину.

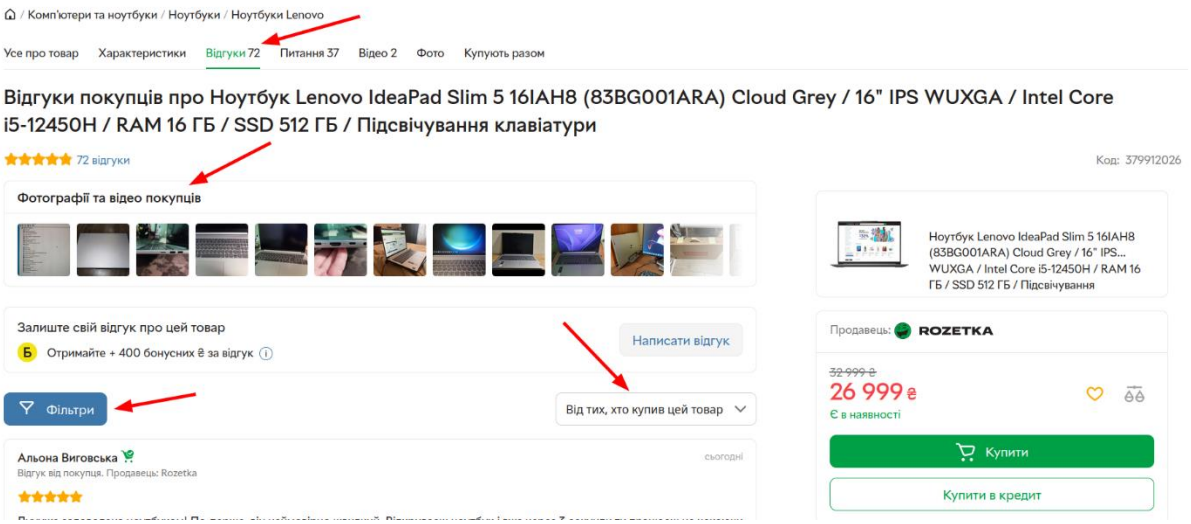

Рисунок 1.11 – Робота з коментарями на маркетплейсі Rozetka

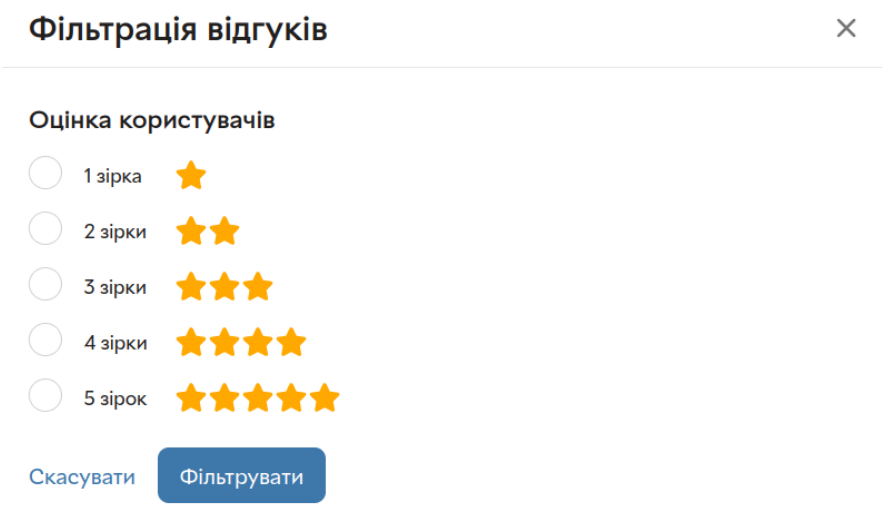

Рисунок 1.12 – Фільтрація коментарів по оцінці на маркетплейсі Rozetka

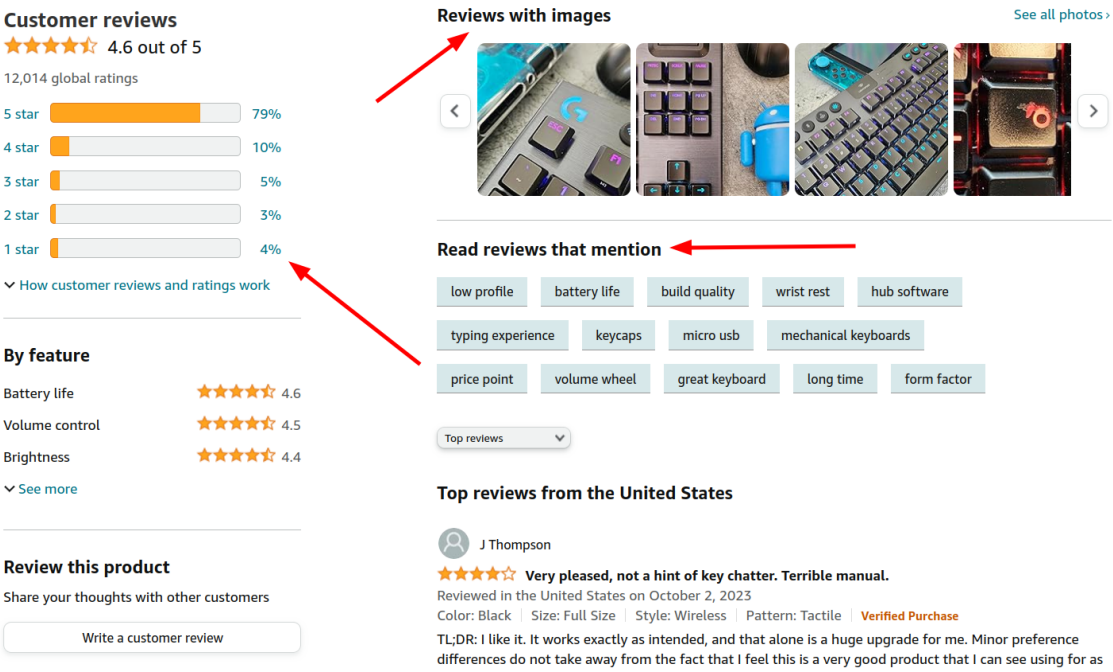

Рисунок 1.13 – Робота з коментарями на маркетплейсі Amazon

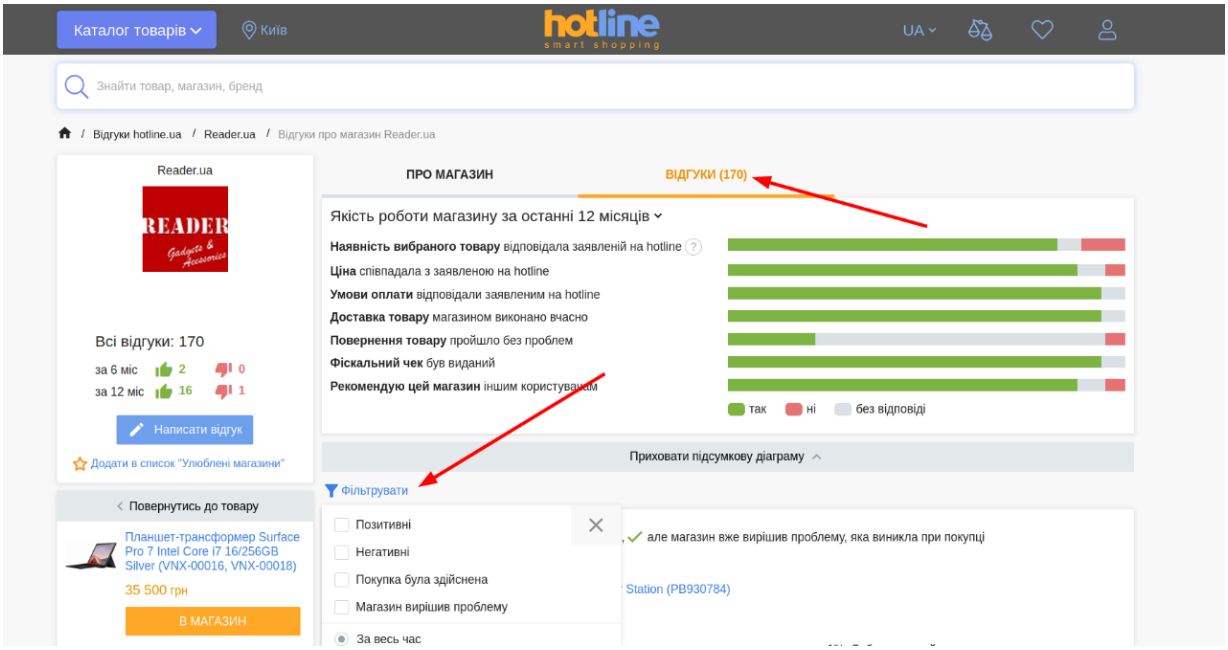

Рисунок 1.14 – Робота з коментарями на маркетплейсі Hotline

Така концепція, узагальнено, працює та надає базовий набір функціоналу для роботи з коментарями, а також можливість розширення функціоналу за допомогою браузера, наприклад, функції пошуку по тексту. Проте вона обмежується доступністю цих інструментів лише на власних сайтах маркетплейсів, що ускладнює користувачам роботу з відгуками з інших джерел.

Хоча вона забезпечує можливість обробки існуючих коментарів на конкретних платформах.

1.2.3 Робота з коментарями на сайтах Reddit, Steam та Youtube

Також великим джерелом коментарів є соціальні мережі різних типів, розглянемо різні варіанти соціальних мереж, а саме Reddit, Steam та Youtube.

Reddit - це великий веб-сайт, який функціонує як платформа для обговорення тем, спільнот та розділів, відомих як "сабреддіти" (subreddits). Заснований в 2005 році Стівеном Хаффманом та Алексом Огін'яном, Reddit вийшов за межі звичайного форуму, пропонуючи унікальну модель обговорення та взаємодії.

Основні риси та характеристики Reddit включають:

- Сабреддіти: Reddit розділений на тисячі тематичних спільнот, відділених за інтересами, галузями, хобі та багато іншого. Користувачі можуть приєднатися до сабреддітів, щоб обговорювати та ділитися контентом відповідно до їхніх інтересів.

- Голосування: Reddit використовує систему голосування "upvote" та "downvote", яка дозволяє користувачам визначати популярність вмісту. Пости та коментарі з більшою кількістю "upvotes" вище піднімаються в ленті новин.

- Розміщення змісту: Користувачі можуть публікувати текстові пости, зображення, посилання та інші види змісту. Велика частина контенту виникає через внесок спільноти.

- AskMe: Інші користувачі можуть задавати питання та отримувати відповіді від інших членів спільноти на підсайтах, таких як "r/AskReddit".

- АмА (Ask Me Anything): Відомий формат інтерв'ю, де особистості або експерти відповідають на питання від користувачів.

Reddit став важливим ресурсом для обміну інформацією, обговорення новин, вираження думок та взаємодії з іншими людьми з усього світу. Свобода

висловлення та широкий спектр тем зробили Reddit важливою частиною культури Інтернету [11].

Steam є визнаною цифровою дистрибуційною платформою для відеоігор, розробленою компанією Valve Corporation. Зокрема, в контексті соціальної мережі, Steam пропонує низку особливостей, що сприяють взаємодії гравців та сприяють формуванню спільноти.

Основні особливості Steam як соціальної мережі включають:

- Профілі та друзі: користувачі можуть створювати особисті профілі, додавати друзів та взаємодіяти з ними через обмін повідомленнями, коментарі та відгуки.

- Групи та заходи: користувачі можуть приєднуватися до груп за інтересами, спільнот, а також брати участь у заходах та змаганнях, що підсилює взаємодію та конкуренцію.

- Стрімінг та захоплення (Broadcasting and Highlights): гравці можуть ream-іти свою гру або ділитися відеооглядами через функції стрімінгу та захоплень, що дозволяє гравцям демонструвати свої вміння та враження від гри.

- Відгуки та оцінки: користувачі можуть залишати відгуки та ставити оцінки іграм, що допомагає іншим гравцям зробити інформований вибір.

- Маркетплейс: існує можливість обміну, купівлі та продажу ігор, предметів та іншого внутрішнього вмісту через торговельну площадку.

- Комунікації під час гри: Steam пропонує можливості голосового та текстового спілкування під час гри, що полегшує комунікацію між гравцями.

Загалом, Steam створює інтерактивне середовище, де гравці можуть спілкуватися, співпрацювати та взаємодіяти як у межах ігор, так і в рамках більш широкої спільноти.

YouTube - це велика відео-платформа, яка включає в себе елементи соціальної мережі, особливо у відношенні до спільнот та взаємодії між користувачами. Основні особливості YouTube як соціальної мережі включають:

- Канали та Підписка: Користувачі можуть створювати свої власні канали, де публікують відео, а інші користувачі можуть підписатися на ці канали. Це

дозволяє створювати власні спільноти вокруг конкретних тематик чи творчого контенту.

- Коментарі та Взаємодія: Кожне відео має розділ для коментарів, де глядачі можуть виражати свою думку, задавати питання та спілкуватися з авторами та іншими глядачами. Це сприяє активній взаємодії та обговоренням.

- Лайки та Дизлайки: Користувачі можуть виражати своє ставлення до відео, ставлячи лайки або дизлайки. Це дозволяє визначити популярність та сприйняття контенту.

- Спільноти та Групи: YouTube дозволяє створювати спільноти та групи, де глядачі можуть обговорювати теми, ділитися враженнями та взаємодіяти поза межами конкретного відео.

- YouTube Live та Чат: Функції YouTube Live дозволяють проводити стріми в реальному часі, а чат надає можливість глядачам спілкуватися під час трансляції.

- Рекомендації та Персоналізація: YouTube використовує алгоритми рекомендацій, що враховують історію перегляду та попередні взаємодії, щоб пропонувати глядачам контент, який може їх зацікавити.

Загалом, YouTube створює віртуальне середовище, де користувачі можуть не лише споживати вміст, але й взаємодіяти з ним та іншими глядачами, що робить його соціальною мережею в розумінні обміну контентом і спілкування.

Сайти, такі як Reddit, Steam, YouTube та інші, надають певні засоби для роботи з коментарями через веб-інтерфейс. Нижче подано загальні функції, які бувають доступні у деяких соціальних мереж:

- Фільтрація та сортування: можна використовувати різноманітні фільтри та параметри для сортування. Наприклад, є можливість переглядати тільки позитивні або негативні відгуки та інші параметри. Застосування поля пошуку дозволяє виявити конкретні слова чи фрази у відгуках. Це сприяє швидкому виявленню коментарів, які містять певні терміни чи вирази.

- Відповіді на коментарі: на коментарі доволі часто бувають відповіді у вигляді інших коментарів, які часом можуть утворювати цілі обговорення. Є

певні графічні, або текстові підказки які показують до якої теми відносяться коментарі.

На рисунках 1.15, 1.16 та 1.17 зображено вигляд коментарів та інтерфейсу роботи з ними на платформах Reddit, Steam та YouTube відповідно.

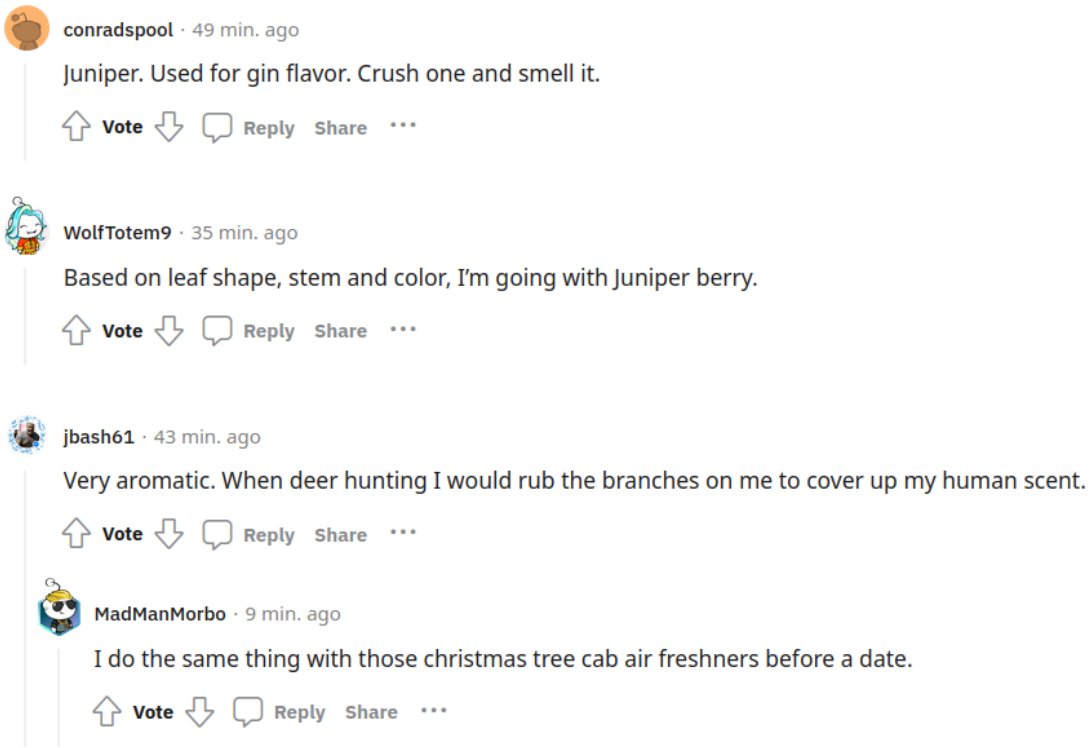

Рисунок 1.15 – Коментарі на Reddit

Як можна помітити, у маркетплейсів був дещо зручніший спосіб перегляду коментарів і набагато ширший інструментарій. Але варто зазначити що на соціальних платформах існує більша увага на кожного користувача, а не на товар чи сервіс. Тому такий інструментарій не реалізовано.

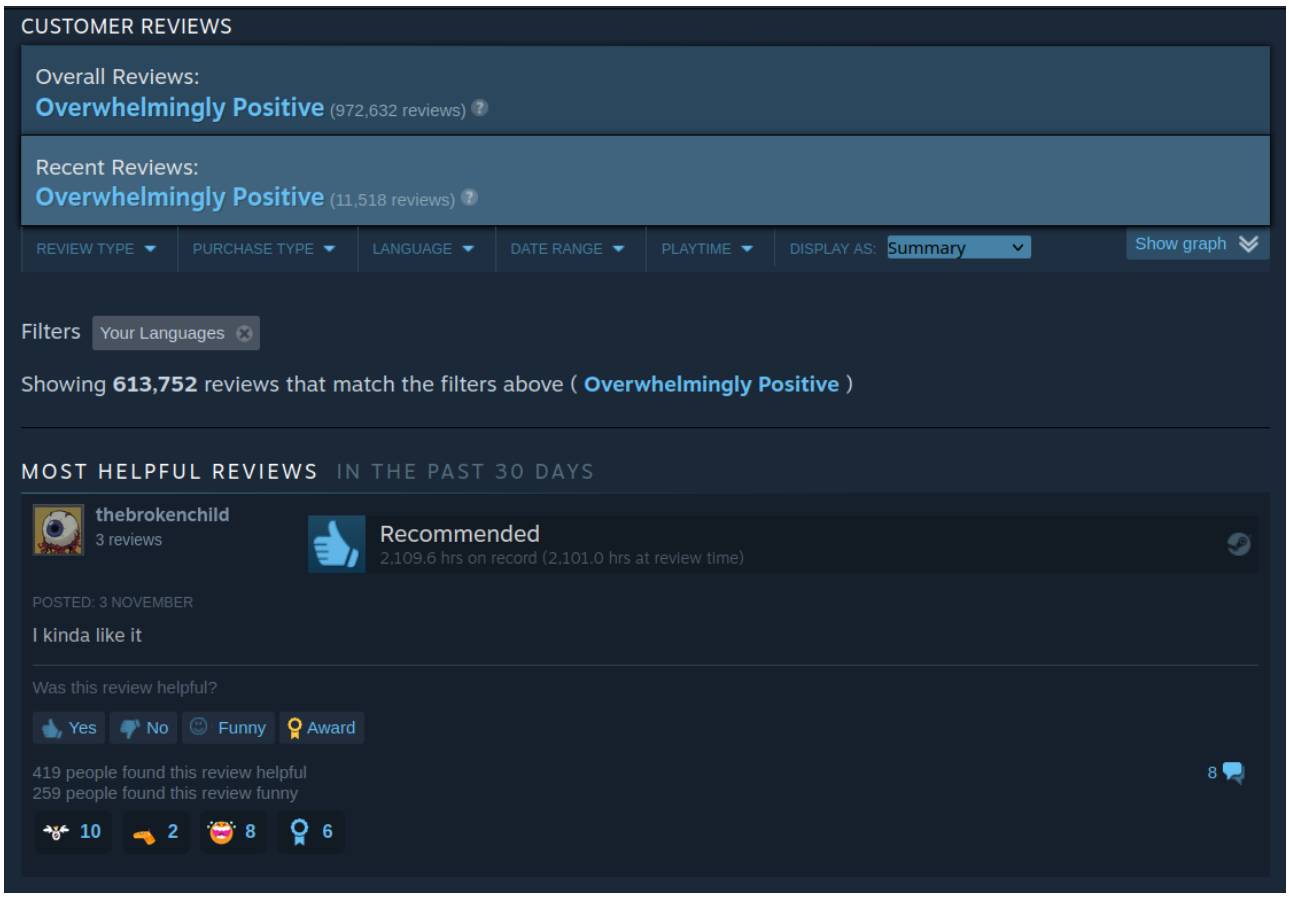

Рисунок 1.16 – Коментарі на Steam

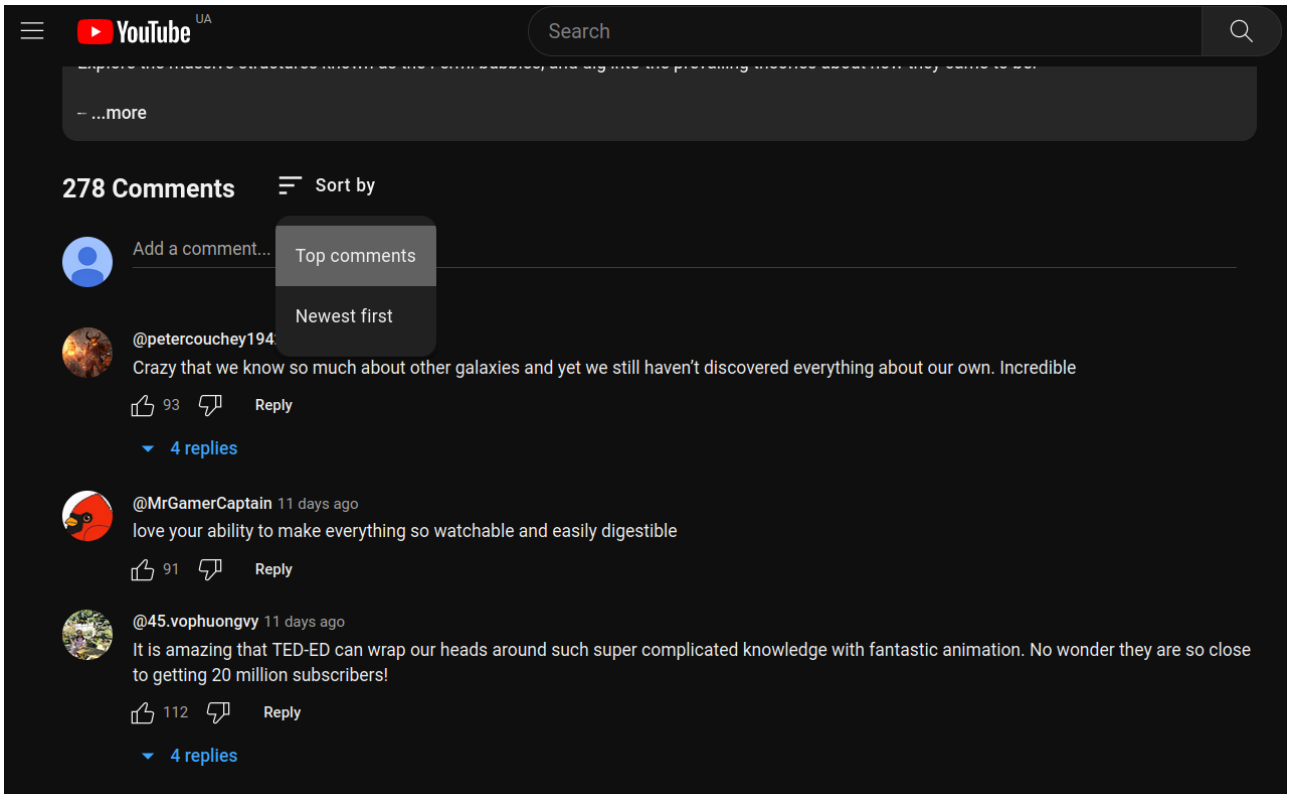

Рисунок 1.17 – Коментарі на Reddit

З точки зору функціоналу можна сказати те саме що і про маркетплейси, що реалізація перегляду коментарів не може слугувати як повноцінний інструмент для пошуку та аналізу коментарів. Також є обмеження що коментарі знаходяться на різних платформах і неможливо переглядати їх на одному сайті.

1.3 Огляд існуючих концепцій застосунків індексації інформації

Мета індексування – швидкий пошук потрібної інформації за певним пошуковим текстом або запитом. Без використання індексування потрібно було б перевіряти і сканувати кожен документ чи файл, що є дуже затратним і не зручним, а також вимагає великої кількості часу і обчислювальної потужності комп'ютеру. Для прикладу, коли із використанням індексування перегляд декількох тисяч документів можна описати в межах мілісекунд, то послідовний перегляд кожного слова міг би зайняти години. Звісно на сам процес виділяється додаткова пам'ять для зберігання індексів документів, але це все компенсується значним зменшенням часу на пошук потрібних файлів. Одна з структур даних що використовується – пошуковий індекс. Він містить потрібну для пошуку документів інформацію та використовується в різних пошукових системах. Індексування – це в першу чергу процес збору потрібної інформації, їх відповідного сортування та зберігання у вигляді даних з метою повторного використання для швидкого та точного доступу до потрібної інформації [12, 13].

На рисунку 1.11 зображено типовий спосіб доступу до індексованої інформації. Користувач (User) робить запит (Query). В таблиці індексів (Index) перевіряється чи містить заданий запит уже проіндексовані слова які по індексу доступні в таблиці яка містить необхідну інформацію (Table). Якщо існують такі данні то всі, які підходять під пошуковий запит, показуються користувачу.

Відомі два способи індексування:

- вільне індексування: із змісту документа або файлу безпосередньо обирають слова які є ключовими без врахування змін їх форм і різних відношень між цими словами;

- контрольоване індексування: в пошуковий образ файлу додаються слова, які присутні в словнику ключових слів, а у ньому в свою чергу вказані їх асоціативні, синонімічні та родо-видові відношення.

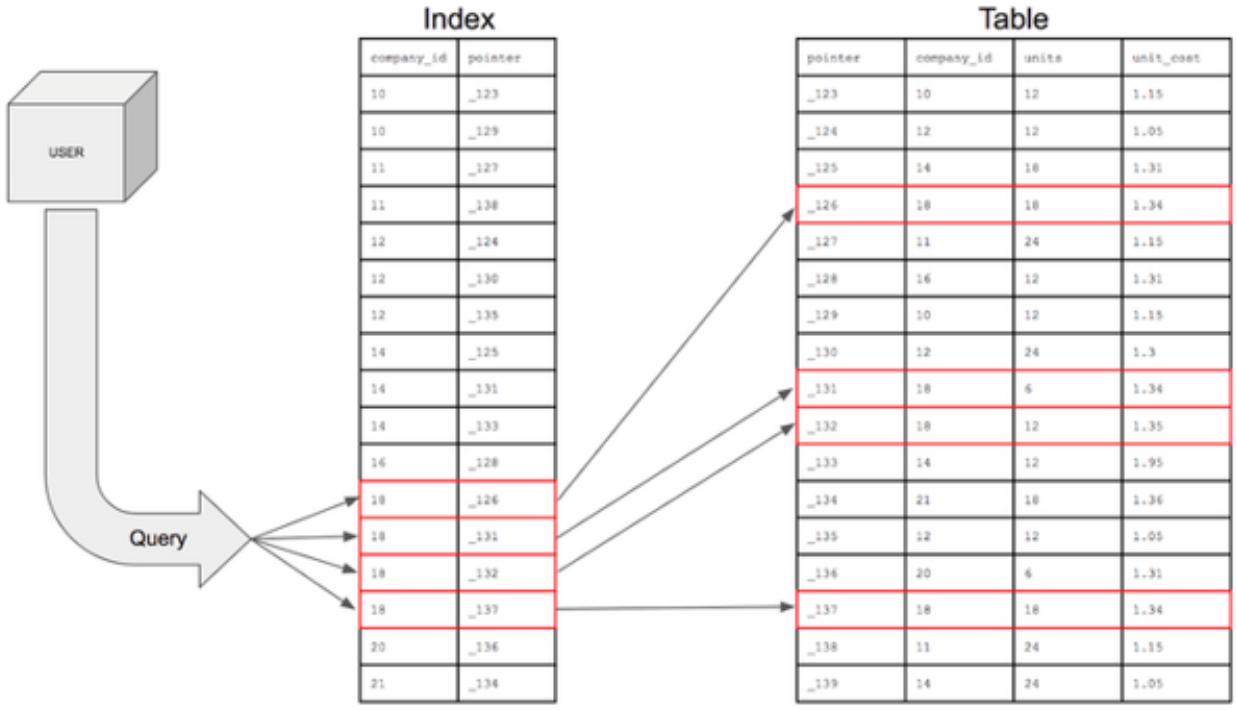

Рисунок 1.18 – Приклад роботи індексації

Також доволі часто використовується така структура даних як інвертований індекс. В якій для кожного слова у наборі з документів у відповідному списку перераховані всі присутні документи у списку — в яких є задане слово. Інвертований індекс використовується для пошуку за текстом. [14]

Щоб показати як працює інвертований індекс розв’яжемо задачу.

Для прикладу у нас є корпус з трьох різних текстів:

X_0 = "it is what it is"

X_1 = "what is it"

X_2 = "it is a banana"

Інвертований індекс буде мати наступний вигляд:

```
banana: {2}
a:      {2}
it:     {0, 1, 2}
is:     {0, 1, 2}
```

```
what:   {0, 1}
```

Цифри в дужках позначають конкретні номери текстів, у яких міститься відповідний елемент. Тоді при пошуковому запиті для тексту "what is it" результат буде наступним:

```
{0, 1} ∪ {0, 1, 2} ∪ {0, 1, 2} = {0, 1}.
```

З результату можна побачити, що нам підходять лише X_0 і X_1, так як у даних текстових блоках присутнє шукане словосполучення.

Є два варіанти інвертованого індексу:

- містить лише список документів для кожного слова;
- додатково включає позицію слова в кожному документу.

Визначити якість індексування можна за двома показниками:

- детальністю;
- глибиною.

Детальність індексування описує точний пошуковий вид відображення змісту файлу або документу. А точність такого відображення визначається за смисловою схожістю ключових слів, доданих до уявної анотації, слів інформаційно-пошукових мов, що утворюють відповідний пошуковий образ.

Глибина в свою чергу описує повноту розкриття змісту файлу або документа в наданому йому пошуковому виді. Оцінити глибину індексування можна приблизно за кількістю слів в інформаційно-пошукових мовах, доданих за індикатором до відповідного пошукового образу.

1.4 Огляд існуючих концепцій обробки природної мови

Обробка природної мови (Natural Language Processing, NLP) – це галузь штучного інтелекту, яка зосереджена на взаємодії між комп'ютерами та людською мовою. Основна мета поля – надавати можливість машинам розуміти, тлумачити та генерувати людську мову способом, що є якісним та контекстуально відповідним. Основними етапами для даної системи є токенізація та лематизація.

Токенізація в NLP – це процес розбиття тексту на менші одиниці, так звані токени. Токен може бути словом, фразою, реченням або навіть окремим символом. Основна мета токенізації полягає в розбитті тексту на зручні для обробки та аналізу одиниці. Наприклад, розглянемо речення "Токенізація – це важливий етап в обробці тексту." Токенізація цього речення може видати такий набір токенів: ["Токенізація", "–", "це", "важливий", "етап", "в", "обробці", "тексту", "."]. Токенізація, один із ключових етапів NLP, включає в себе розбиття тексту на менші одиниці, які називають токенами. Токен може представляти собою слово, фразу, речення або навіть окремий символ. Цей процес є важливим для подальшої обробки та аналізу тексту [15].

Лематизація, інший етап, спрямований на визначення базової форми слова, що називається лемою, допомагає нормалізувати граматичні варіації та спрощує подальший аналіз. Враховуючи словозміни, лематизація сприяє однорідності в розумінні тексту та полегшує подальшу обробку. Наприклад, лематизація слова "бігаючи" може повернути лему "бігати". Це допомагає враховувати різні граматичні форми слова як одиницю для аналізу. Лематизація сприяє зменшенню розміру словника та покращенню якості аналізу тексту. Порівняно з токенізацією, лематизація виходить за рамки простого розбиття тексту на частини. Вона враховує мовні особливості, такі як словозміни, та старається визначити базові форми слів для подальшого аналізу. Обидві ці процедури є важливими етапами підготовки тексту для більш складних завдань в NLP, таких як визначення сутностей, визначення тональності або побудова моделей машинного навчання для обробки природної мови [16].

Ці процеси становлять основу для більш складного використання в NLP, таких як аналіз настрою, визначення іменованих сутностей та машинний переклад [17]. Розвиток NLP відкриває нові можливості для взаємодії машин із людьми та застосування в різних галузях, від технологічних до медичних та багатьох інших. Стандартний сценарій роботи NLP зображений на рисунку 1.16 і більш детальний зображений у додатку Б на рисунку Б.1.

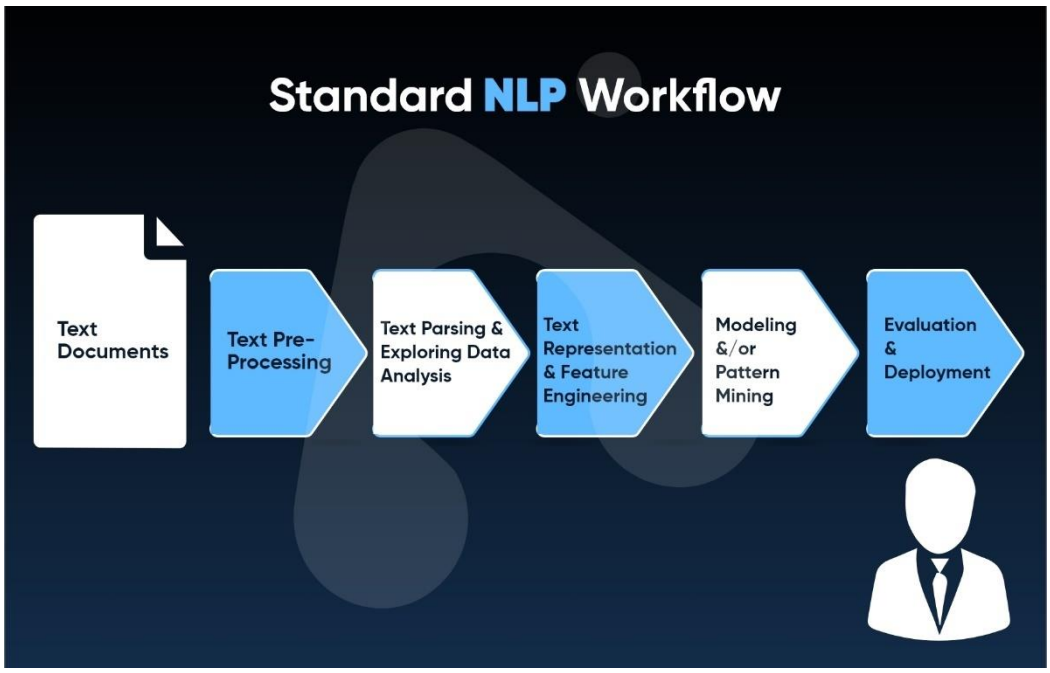

Рисунок 1.19 – Стандартний сценарій роботи NLP

Також NLP включає такі аспекти та способи застосування:

– Обробка тексту та розмовного мовлення: NLP передбачає обробку як письмового тексту, так і сказаного слова. Це включає в себе завдання, такі як розуміння тексту, аналіз настрою, розпізнавання мовлення та переклад мови.

– Визначення частин мови: це включає в себе визначення та позначення частин мови (іменників, дієслів, прикметників і т.д.) для кожного слова в реченні. Це допомагає розуміти граматичну структуру тексту.

– Визначення іменованих сутностей (NER): NER передбачає визначення та класифікацію сутностей, таких як імена людей, організації, місця, дати та інші в тексті.

– Синтаксис та синтаксичний розбір: Синтаксичний аналіз та розбір передбачає розуміння граматичної структури речень. Це важливо для витягнення значущої інформації з даного тексту.

– Семантичний аналіз: NLP спрямована на розуміння значення слів та їх взаємозв'язків. Семантичний аналіз допомагає розуміти призначене значення речення або документа.

– Машинний переклад: NLP використовується для автоматичного перекладу тексту або мовлення з однієї мови на іншу.

– Аналіз настрою: Аналіз настрою або відсутність спрямований на визначення настрою, вираженого в тексті. Це часто використовується для оцінки громадської думки за різними темами.

– Чатботи та віртуальні асистенти: NLP використовується у розробці чатботів та віртуальних асистентів, які можуть розуміти та реагувати на запитання чи команди користувачів природною мовою.

– Системи відповідей на запитання: NLP використовується для побудови систем відповідей на запитання, які можуть розуміти питання користувача та надавати відповіді на основі наявної інформації.

1.5 Оцінка дослідженого програмного забезпечення

Було розглянуто найбільш популярні та функціональні системи індексування та пошуку даних. Також були розглянуті популярні маркетплейси і платформи які надають мінімальний інструментарій для роботи з коментарями. Були досліджені існуючі концепції які використовуються у наведених програмах та платформах. Проведена їх оцінка та виведено основні недоліки: відсутність або обмеженість безкоштовної версії, складність структури, перевантажений інтерфейс.

Сьогодні існує багато систем для індексації інформації, які мають унікальний інтерфейс та алгоритми. Деякі з них використовують власні способи індексації та інтерфейсі адаптовані для певних користувачів. Також ці програми створюються для різних цілей. Проте багато з цих програм мають застарілий інтерфейс або взагалі не підтримують певні операційні системи. Більше того деякі з них мають обмежений або застарілий функціонал і мають платну версію, яка не завжди може себе окупити, якщо потрібно виконувати невеликий обсяг роботи.

Щодо існуючих програм для аналізу коментарів, варто зауважити, що їх кількість надзвичайно обмежена, а деякі взагалі не надають необхідних можливостей. Оглянуті платформи є лише частковим рішенням і не спеціалізуються на аналізі коментарів, а просто мають певний функціонал у цьому напрямку. Тому існує актуальна необхідність розробки нового програмного забезпечення, спеціально зорієнтованого на ці цілі.

2 ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЇ ТА МЕТОДІВ РОЗРОБКИ

Планування є однією з найважливіших задач під час розробки програмного забезпечення. Саме на цьому етапі буде вказаний шлях розвитку програми, її можливості та функції. Також, під час планування приймаються архітектурні рішення які є надзвичайно важливими і визначають на скільки система буде гнучкою та масштабованою.

Для розробки загальної структури та реалізації необхідного функціоналу на стадії розробки потрібно обрати технології та обґрунтувати доцільність їх використання. Необхідно розробити відповідну структуру та описати алгоритми роботи відповідно до поставлених задач. Потрібно описати та довести доцільність використання кожного компоненту системи і правильно інтегрувати його з іншими компонентами програми.

2.1 Основні вимоги до розроблюваного програмного забезпечення

Необхідно обрати потрібні технології та інструменти виходячи із задач нашої програми, а саме:

- кросплатформеність;
- простота підтримування та доопрацювання системи в майбутньому;
- підтримка створення інтерфейсу користувача;
- швидкість роботи;
- надійність та безпечність роботи.

Також варто виділити інструменти які можуть змінюватись, але при цьому є не менш важливими:

- Середовище розробки;
- Операційна система для розробки;
- Допоміжні застосунки та програми;
- Стороні бібліотеки та API.

2.2 Вибір мови програмування

Від вибору мови програмування буде визначити доступність конкретних інструментів розробки, які оптимізовані для певних мов. Наприклад, для одних мов існують потужні інтегровані середовища розробки (IDE), що полегшують процес програмування та відладки, тоді як інші мови можуть користуватися менш розвинутими інструментами.

Крім того, вибір мови визначає доступність різноманітних бібліотек, фреймворків та інших ресурсів, які можуть значно полегшити розробку, зменшити час витрачений на написання коду та забезпечити більшу функціональність.

Мова програмування також впливає на масштабованість проекту та його здатність до майбутнього розширення. Деякі мови мають вбудовану підтримку для паралельного програмування чи обробки великої кількості даних, що може бути критичним для великих проектів.

Таким чином, вибір мови програмування – це стратегічне рішення, яке може значно впливати на ефективність розробки, підтримку та розширюваність програмного забезпечення. Правильний вибір дозволяє оптимально використовувати ресурси та забезпечує успішну реалізацію проекту.

Розглянемо та порівняємо одні з найпопулярніших і найбільш відповідних до наших задач мови програмування, а саме Java, C# та Python.

Java – це високорівнева, об'єктно–орієнтована мова програмування, яка вперше була представлена компанією Sun Microsystems у 1995 році. Однією з ключових особливостей Java є її платформонезалежність, що означає, що програми, написані на Java, можуть виконуватися на будь–якому пристрої, що підтримує відповідну віртуальну машину Java (JVM). Java широко використовується в розробці веб–застосунків, мобільних додатків (особливо на платформі Android), корпоративних систем та великих мережевих додатків. Java використовується у багатьох галузях і залишається однією з найпопулярніших мов програмування в світі [18]. Вона володіє великою екосистемою бібліотек і

фреймворків, що полегшують розробку та підтримку програмного забезпечення. Також слід зауважити велику кількість фреймворків, таких як Spring, Hibernate та інших, які зарекомендували себе і продовжуть розвиватись. Java показує себе ефективно в різних задачах, є доволі зрозумілою і швидкою в розробці. [19].

C# – це мова програмування, розроблена компанією Microsoft, яка була випущена в 2000 році. Вона є об'єктно–орієнтованою мовою та частиною технологій Microsoft .NET Framework. C# призначена для розробки різноманітних додатків, включаючи веб–застосунки, десктопні програми, ігри, мобільні додатки та корпоративні системи. Однією з ключових особливостей C# є його інтеграція з платформою .NET, що забезпечує велику бібліотеку класів та фреймворків для розробки різних застосунків. C# підтримує безпеку типів, автоматичне управління пам'яттю та високий рівень абстракції, що сприяє продуктивній розробці програмного забезпечення. Ця мова використовується як основна мова розробки для платформи Microsoft, включаючи розробку додатків для операційних систем Windows та платформи Azure для хмарних обчислень. C# також використовується у розробці ігор за допомогою технологій, таких як Unity.

Python – це високорівнева, інтерпретована, об'єктно–орієнтована мова програмування з акцентом на простоту читання коду та ефективність розробки. Розробленою Гвідо ван Россумом, Python вперше був випущений у 1991 році і швидко завоював популярність через свою зручність та універсальність.Однією з головних переваг Python є його чіткий та лаконічний синтаксис, який полегшує розробку коду та зменшує кількість потрібних рядків для вираження ідей. Python також відомий своєю розширюваністю та великою кількістю бібліотек, які покривають різні галузі, включаючи науку про дані, штучний інтелект, веб–розробку та інше. Мова використовується для розробки різноманітних проектів, включаючи веб–застосунки, автоматизацію завдань, наукові дослідження, обробку даних, штучний інтелект та інше. Python також є популярним в середовищі машинного навчання та аналізу даних завдяки бібліотекам таким як NumPy, pandas, TensorFlow та PyTorch.

Всі запропоновані мови мають garbage collector (збирач сміття), який є необхідним та надзвичайно зручним інструментом, для забезпечення надійності, швидкості та безпечності роботи.

Оцінки виставимо в таблиці 2.1. по 5–бальній шкалі де 1 це дуже погано, а 5 це дуже добре.

Таблиця 2.1 – Порівняння мов програмування

Мова програмування	Java	C#	Python
Швидкість та ефективність	5	4	2
Кросплатформеність	5	3	4
Масштабованість та розширюваність	5	3	4
Кількість бібліотек та інструментів	4	3	5
Зручність та швидкість розробки	4	4	5
Універсальність та адаптивність під різні задачі	4	3	5
Безпека та надійність	5	5	3
Загальна кількість балів	32	25	28

Отже Java є найкращим варіантом для вказаних вимог і потреб проекту. Опишемо більш детально кілька аспектів, які роблять Java гарним вибором:

– Найкраща кросплатформеність: Програми, написані на Java, можуть виконуватися на різних платформах, оскільки вони транслюються в байт–код, який виконується на віртуальній машині Java (JVM). Це робить Java ідеальним вибором для розробки кросплатформових додатків.

– Гарна безпека та надійність: Java відома своєю вбудованою безпекою та надійністю. Вона використовує систему управління пам'яттю, що дозволяє

уникнути багатьох типів помилок, пов'язаних із витоками пам'яті та неправильними вказівниками.

– Широкий вибір бібліотек та фреймворків: Java має величезну екосистему бібліотек і фреймворків, які полегшують розробку. Спільнота Java дуже активна, і це призводить до появи нових інструментів та рішень.

– Об'єктно–орієнтоване програмування: Java використовує об'єктно–орієнтований підхід, що полегшує розробку, тестування та підтримку коду.

– Масштабованість: Java підтримує масштабованість, що робить її ідеальним вибором для великих корпоративних проектів та високозавантажених систем.

– Підтримка великих проектів та команд: Java має добре визначені практики розробки, а також інструменти для управління залежностями та конфігурацією, що сприяє роботі великих команд розробників.

Але варто зауважити що Python теж є гарним варіантом, хоча і в дечому поступається Java, але для деяких задач він також може застосуватись в майбутньому для окремих модулів. Наприклад, Python можна застосувати для веб-скрапінгу коментарів для даної системи.

2.3 Обґрунтування вибору інструментів розробки

Основним середовищем розробки було обрано IntelliJ IDEA, яка зарекомендувала себе як багатофункціональна, надійна та швидка система [20]. Приклад її інтерфейсу можна побачити на рисунку 2.1.

IntelliJ IDEA – це потужна інтегрована середовища розробки (IDE) для мов програмування Java, Kotlin, інших JVM–мов, а також деяких інших технологій.

Короткий огляд переваг IntelliJ IDEA:

– Зручний редактор коду: IntelliJ IDEA надає швидкий та інтуїтивно зрозумілий редактор коду з підтримкою автодоповнення, рефакторингу та великою кількістю інших функцій, що полегшують написання та редагування коду.

– Вбудовані функції і аналітика для рефакторингу: інтелектуальні інструменти рефакторингу дозволяють легко вносити зміни до коду, підтримуючи його читабельність та ефективність.

– Аналіз коду та виявлення помилок: IntelliJ IDEA автоматично аналізує код, надаючи рекомендації, виявляючи помилки та допомагаючи забезпечити високу якість програмного забезпечення.

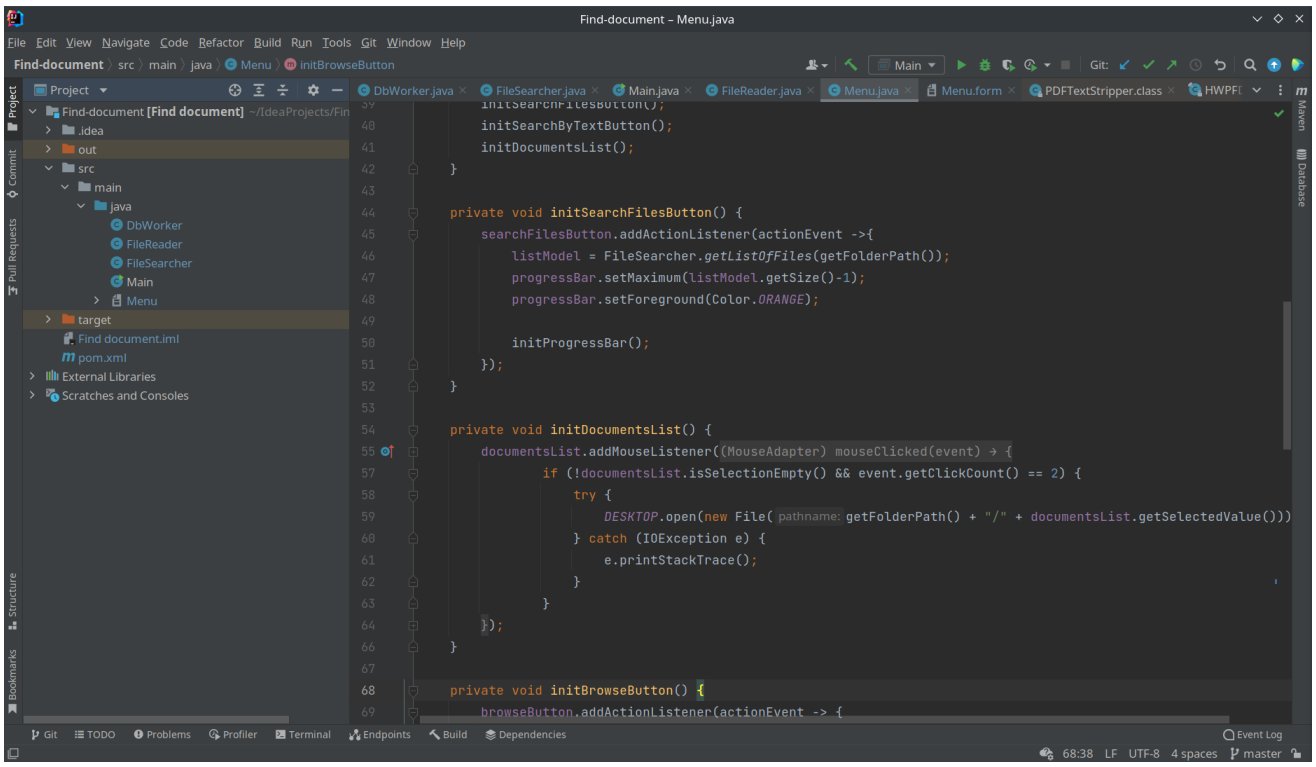

Рисунок 2.1 – Вигляд інтерфейсу Intellij IDEA

– Інтеграція з іншими інструментами та фреймоврками: підтримка широкого спектру інструментів розробки, тестування, систем контролю версій, фреймворків і бібліотек дозволяє легко інтегрувати IntelliJ IDEA в різні процеси.

– Вбудовані інструменти для тестування: IntelliJ IDEA надає зручні засоби для написання, запуску та відлагодження тестів, сприяючи практиці тест–драйву та забезпечуючи високу якість коду.

– Кросплатформеність: IntelliJ IDEA підтримується на різних операційних системах, включаючи Windows, macOS і Linux.

Ці переваги роблять IntelliJ IDEA популярним вибором серед розробників Java та інших JVM–мов для зручного та продуктивного програмування.

2.4 Обґрунтування вибору архітектурних методів та підходів розробки

Вибір мови Java означає, що також обирається ООП (об’єктно орієнтоване програмування). Одна з найпопулярніших парадигм програмування. Вона розглядає програму як множину об'єктів, які можуть взаємодіяти між собою і таким чином утворювати структуру.

Основні принципи ООП складають: інкапсуляція; успадкування; поліморфізм; абстракція.

Великою перевагою ООП є краща модульність програмного забезпечення (для прикладу, тисячу рядків коду для функцій із процедурної мови, в ООП можна замінити всього кількома десятками класів із своїми методами) [21, 22].

Також інкапсуляція, успадкування, поліморфізм та абстракція полегшують і прискорюють розробку програмного забезпечення. Дану парадигму широко використовують для розробки програмного забезпечення, так як вона є перевіреною та зручною у використані. Приклад реалізації ООП на практиці зображено на рисунку 2.2.

Якщо підсумовувати переваги і особливості ООП, то будемо мати такий список:

– Модульність і розширюваність: класи дозволяють групувати дані та методи в єдиний об'єкт. Це сприяє модульності, що полегшує розробку і обслуговування програм. Можливість спадкування дозволяє створювати нові класи на основі вже існуючих, зберігаючи їхні властивості. Це полегшує розширення функціоналу і внесення змін без модифікації вже наявного коду.

– Поліморфізм: Параметризований поліморфізм дозволяє використовувати однаковий інтерфейс для різних типів об'єктів, поліпшуючи читабельність коду та зменшуючи його обсяг. Спадковий поліморфізм дозволяє

використання спадкування і перевизначення методів дозволяє об'єктам конкретних класів вести себе по–різному за допомогою спільного інтерфейсу.

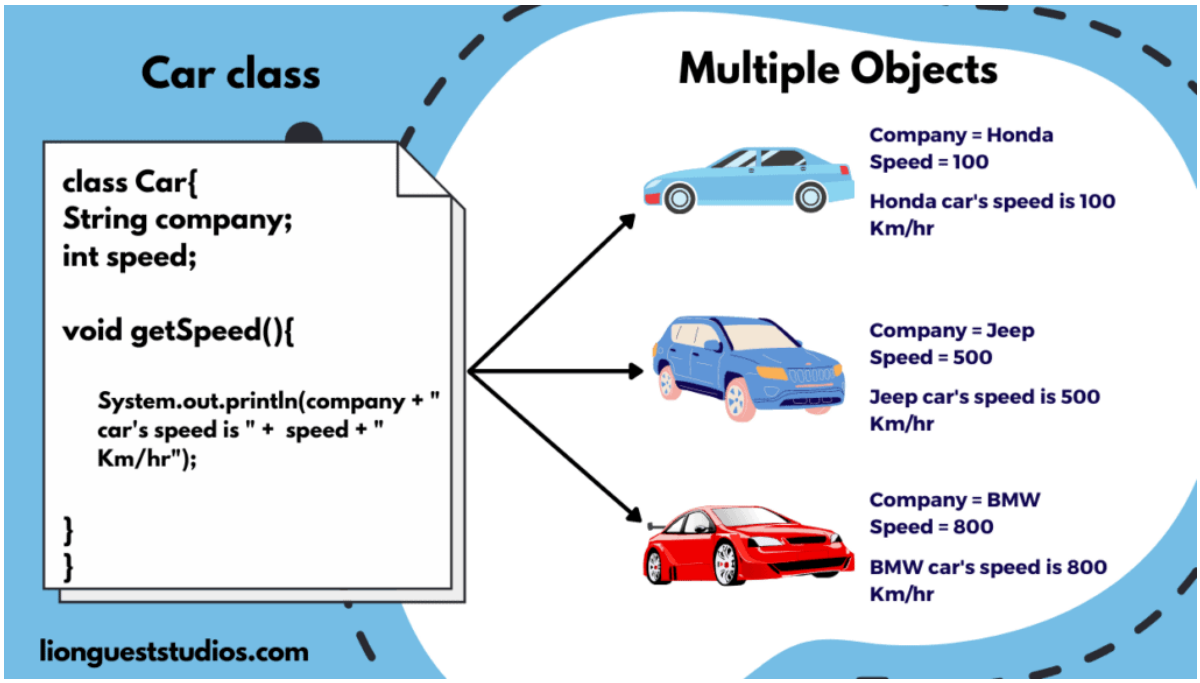

Рисунок 2.2 – Приклад реалізації ООП з класом Car

– Інкапсуляція: забезпечує захист внутрішньої реалізації класів та об'єктів від зовнішнього втручання. Зовнішній код може взаємодіяти лише з публічним інтерфейсом класу, не вдаваючись в його деталі реалізації.

– Підтримка повторного використання коду: можливість визначення класів і створення бібліотек дозволяє легко повторно використовувати код у різних частинах програми або навіть у різних програмах.

– Зменшення складності: Забезпечення високого рівня абстракції дозволяє програмістам концентруватися на важливих аспектах проблеми, мінімізуючи деталі реалізації.

– Моделювання реальних об'єктів: ООП дозволяє програмістам моделювати об'єкти та взаємодії між ними, що сприяє кращому розумінню та аналізу проблем.

Загалом, ООП допомагає покращити структуру програм, зменшити складність коду, полегшити його обслуговування та розширення, що робить його популярним в багатьох галузях програмування.

2.5 Обґрунтування вибору систем управління базами даних

Наша система буде використовувати базу даних для збереження індексів та проіндексованої інформації. Найпопулярнішими СУБД є MySQL та PostgreSQL. І та і та система мають свої переваги і недоліки і можуть задовольнити виконання задач програми. Але найкращим рішенням у цьому випадку буде виступати PostgreSQL через більшу гнучкість та адаптивність [23].

Більш детально про особливості та переваги PostgreSQL:

- Високий ступінь безпеки, надійності та стабільності: PostgreSQL має добре визначену систему контролю цілісності даних, яка дозволяє уникати помилок та зберігає цілісність даних навіть у випадку виникнення непередбачуваних ситуацій. Підтримка транзакцій забезпечує консистентність бази даних, і в разі виникнення помилки можна повернути зміни. PostgreSQL має різні механізми безпеки, включаючи ролі, автентифікацію, відмовостійкість та шифрування даних, що дозволяє забезпечити надійний захист інформації.

- Розширені можливості SQL: PostgreSQL володіє потужним та розширеним SQL–діалектом, який дозволяє використовувати різноманітні операції та запити для взаємодії з базою даних. Підтримка геоданих, робота з JSON та інші розширені типи даних дозволяють ефективно взаємодіяти з різноманітними форматами інформації.

- Розширюваність: Можливість розширення функціоналу бази даних за допомогою власних функцій, типів та індексів робить PostgreSQL високоякісним рішенням для великих та складних проектів. Велика кількість розширень та плагінів сприяє вирішенню різноманітних завдань.

- Підтримка ACID: PostgreSQL дотримується принципів ACID (Атомарність, Консистентність, Ізольованість, Довершеність), що гарантує надійність та цілісність даних навіть у випадку ситуацій відмов та виключень.

- Підтримка реплікації та висока продуктивність: PostgreSQL надає можливості для налаштування реплікації, що покращує продуктивність та забезпечує високу доступність. Підтримка паралельного виконання запитів та оптимізації запитів сприяє високій продуктивності системи.

- Активна спільнота і відкритий код: PostgreSQL є вільним та відкритим програмним забезпеченням (Open Source). Велика активна спільнота розробників та користувачів сприяє швидкому виправленню помилок та розвитку нових функціональних можливостей.

Ці переваги роблять PostgreSQL привабливим вибором для розробників та організацій, що шукають потужну та надійну СУБД для своїх проектів.

Разом у поєднанні з PostgreSQL планується використовувати популярний та потужний фреймоврк для роботи з базами даних, а саме Hibernate.

Hibernate – це фреймворк для роботи з базами даних в середовищі мови програмування Java. Основною його метою є полегшення роботи з реляційними базами даних через об'єктно–реляційне відображення (ORM). Нижче наведено короткий огляд характеристик та переваг Hibernate:

– Об'єктно–реляційне відображення (ORM): Hibernate надає механізми для зіставлення об'єктів Java з записами в реляційній базі даних. Це полегшує роботу з даними, оскільки розробникам не потрібно працювати напряму з SQL–запитами.

– Незалежність від системи управління базами ваних (DBMS): Hibernate забезпечує абстракцію бази даних, що дозволяє розробникам працювати з об'єктами, незалежно від конкретної реляційної СУБД.

– Мова HQL (Hibernate Query Language): Hibernate використовує мову HQL, яка є об'єктно–орієнтованою мовою запитів, схожою на SQL, але оперує об'єктами, а не таблицями.

– Кешування даних: Hibernate підтримує можливості кешування, що може значно покращити продуктивність, особливо при частих операціях читання з бази даних.

– Управління сесіями та транзакціями: Hibernate надає управління сесіями, що дозволяє взаємодіяти з базою даних, а також керування транзакціями для забезпечення цілісності даних.

– Підтримка асоціацій та наслідування: Hibernate забезпечує можливості роботи з асоціаціями між об'єктами та наслідуванням в об'єктній моделі.

– Підтримка Java EE та Spring: Hibernate може інтегруватися з різними технологіями та фреймворками, такими як Java EE і Spring, що робить його популярним в екосистемі Java.

– Відкритий код та активна спільнота: Hibernate є проектом з відкритим вихідним кодом, і він має активну спільноту розробників, яка постійно вдосконалює та підтримує його.

Hibernate використовується для спрощення взаємодії з базами даних у Java–проектах, забезпечуючи високий рівень абстракції та продуктивність розробки.

Hibernate в свою чергу використовує JDBC (Java DataBase Connectivity) — прикладний програмний інтерфейс Java, який визначає методи, за допомогою яких можна працювати з базою даних.

Java DataBase Connectivity – це кросплатформений та сучасний промисловий стандарт для взаємодії з різноманітними СУБД. Це все реалізовано у вигляді пакета для java.sql, який в свою чергу входить до складу Java SE [24].

Схему роботи JDBC наведено у додатку Б на рисунку Б.2.

2.6 Обґрунтування вибору інструментів автоматичної збірки проєктів

Додатково будуть використовуватись інструменти автоматичної збірки проектів які автоматизовують роботу з проектом та допомогають оновлювати необхідні бібліотеки.

Найпопулярнішими такими інструментами для використання разом із Java є Gradle та Maven. Вони мають увесь необхідний функціонал для роботи з проектами, є зручними та надійними. Також плюсом даних інструментів є популярність, кількість інформації та навчальних ресурсів які можна знайти в мережі Інтернет.

Gradle – це інструмент автоматизації збірки та управління залежностями, який може використовуватися для будь–яких проектів розробки програмного забезпечення, але часто використовується в Java–проектах. Основна мета Gradle – надати гнучкість та ефективність у збиранні проектів та керуванні їх залежностями.

Apache Maven – це інструмент для управління проектами та залежностями в середовищі розробки програмного забезпечення на мові програмування Java. Основна мета Maven – це спростити процес збирання, створення та управління проектами Java, а також автоматизувати багато стандартних завдань розробки.

Основні характеристики Maven:

- Система збирання проектів: Maven використовує POM (Project Object Model) для опису проекту та його залежностей. Система збирання Maven автоматично обробляє задачі збирання, тестування та пакування проекту.

- Централізоване управління залежностями: Maven дозволяє описувати та керувати залежностями проекту в одному файлі POM. Залежності автоматично завантажуються з центрального репозитарію Maven чи інших зовнішніх репозитаріїв.

- Стандартні структура каталогів: Maven накладає стандартну структуру каталогів на проекти, що сприяє стандартизації та спрощує обмін проектами між розробниками.

- Життєвий цикл збирання (Build Lifecycle): Maven визначає життєвий цикл збирання, який включає етапи від компіляції та тестування до пакування та розгортання.

- Плагіни: Maven підтримує велику кількість плагінів, які дозволяють розширити функціоналість інструменту для різних задач розробки.

- Автоматичне керування завданнями (Task Automation): Maven автоматизує багато завдань розробки, таких як генерація документації, обробка ресурсів, запуск тестів та інше.

- Інтеграція з IDE: Maven може інтегруватися з популярними середовищами розробки, такими як Eclipse та IntelliJ IDEA, для спрощення роботи з проектами.

Apache Maven є потужним інструментом для розробки на Java, забезпечуючи стандартизацію та автоматизацію процесів розробки програмного забезпечення.

Основні характеристики Gradle:

- Декларативні Залежності: Gradle автоматично вирішує та завантажує залежності проекту з центральних репозиторіїв, так само як інші інструменти управління залежностями.

- Життєвий Цикл Збирання (Build Lifecycle): Gradle надає гнучкий життєвий цикл збирання, де користувач може налаштовувати етапи збирання за своїми потребами.

- Інкрементальна Збірка: Gradle підтримує інкрементальну збірку, що дозволяє зменшити час збирання та використовувати результати попередніх збірок.

- Плагіни та Розширення: Gradle підтримує широкий вибір плагінів для різних завдань розробки. Також можна визначати власні розширення та завдання.

- Інтеграція з Іншими Інструментами: Gradle інтегрується з різними інструментами розробки, такими як IntelliJ IDEA, Eclipse, та має підтримку для використання в середовищах CI/CD.

- Підтримка Багатьох Мов: Відмінною рисою Gradle є можливість використовуватися для проектів, написаних на різних мовах програмування, не лише на Java.

Gradle намагається поєднати силу Apache Ant та Apache Maven, пропонуючи відмінні можливості конфігурації, зручність опису проектів та продуктивність.

Хоча Gradle має багато переваг [25], але він буде занадто громіздким для даного програмного забезпечення [26]. Тому буде використовуватись Maven, який є більш простішим та легшим. Він більше підходить для невеликих проектів які не потребують складних налаштувань та маніпуляцій зі сторонніми репозиторіями. Принципи роботи показані у у додатку Б на рисунку Б.3.

2.7 Архітектурні шаблони

При розробці програмного забезпечення етап проектування є обов’язковим, це допомагає уникнути проблем у майбутньому та зробити систему масштабованою і відкритою до майбутніх змін. Існує дуже багато архітектурних прийомів які перевірені часом та дозволяють легко визначити як має виглядати загальна структуру та компоненти програми.

Для системи, що розробляється, підходять два архітектурних шаблони Модель–вигляд–контролер (MVC, Модель–представлення–контролер, англ. Model–view–controller) та Передача репрезентативного стану (REST, англ. Representational State Transfer) Схему даного шаблону можна побачити на рисунку 2.4.

MVC застосовується для розділення даних так, щоб зміни вигляду (графічного інтерфейсу користувача) мали мінімальний вплив або взагалі не мали впливу на роботу з даними, а нові зміни в моделі даних можна було здійснювати без змін графічного інтерфейсу. Також він дає можливість повторного використання окремих модулів та компонентів програми.

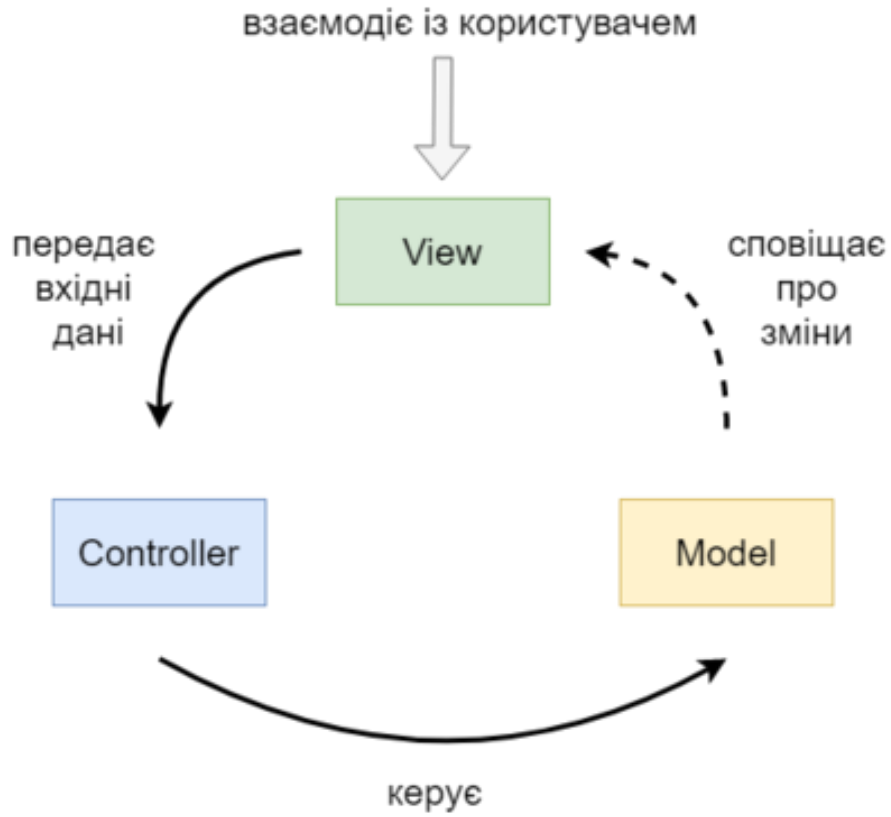

Рисунок 2.3 – Схема роботи MVC

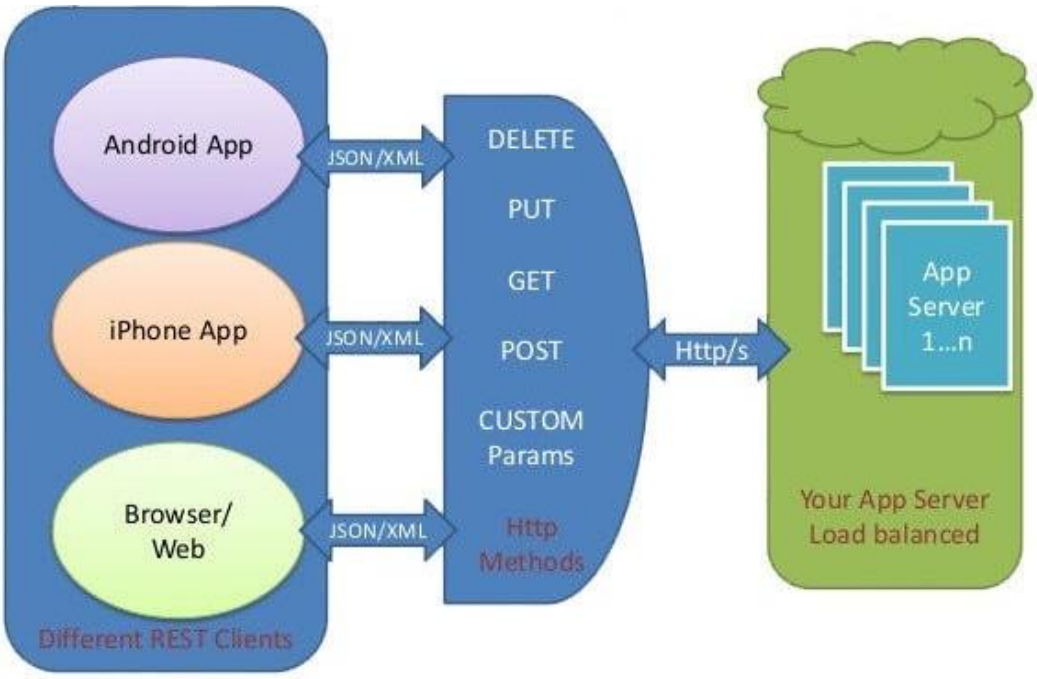

Рисунок 2.4 – Схема роботи

Відповідно мета MVC – це гнучкий дизайн програмного забезпечення, який полегшує розширення та подальші зміни в архітектури програмного забезпечення. Більше того використання даного шаблону у системах великих розмірів сприяє модульності та впорядкованості їхньої структури. Це зменшує складність і робить їх більш зрозумілішими.. [27, 28].

Цей шаблон передбачає розділення системи на три взаємопов'язані між собою частини:

- модель даних (Model);
- вигляд, або ж графічний інтерфейс користувача (View);
- модуль керування. (Controller).

Хоча MVC є дуже поширеним шаблоном проектування та архітектурою для розробки програмного забезпечення, він є дещо застарілим і має свої недоліки:

- Складність при масштабуванні: при збільшенні розміру системи або зростанні обсягу коду, важко управляти всією логікою в рамках одного проекту. Підвищується складність обслуговування.

- Перенавантаженість контролерів: контролер може стати перенавантаженим, особливо у великих проектах, де він відповідає за обробку багатьох запитів та управління потоком додатка.

- Невизначеність між компонентами: важко чітко визначити, яка логіка повинна бути в моделі, а яка - в контролері або представленні. Це може призвести до плутанини та невизначеності у деяких питаннях під час розробки.

- Контроль над відображенням: у деяких випадках контролер може занадто активно взаємодіяти з представленням, що може призвести до втрати модульності та ускладнення тестування.

- Змішування рівнів: У реалізації деяких додатків може статися змішування логіки різних рівнів (наприклад, бізнес-логіки із представленням), що може призвести до менш “чистого” та менш структурованого коду.

- Залежність від реалізації конкретного інтерфейсу: MVC часто пов'язаний із конкретним інтерфейсом користувача (наприклад, веб-інтерфейсом), що може ускладнити використання його в інших контекстах.

REST (Representational State Transfer) - це архітектурний стиль, який визначає набір обмежень та принципів для побудови масштабованих та ефективних систем в розподіленому середовищі. Цей стиль був описаний у дисертації Роя Філдінга (Roy Fielding) в 2000 році та став широко використовуваним при розробці веб-сервісів.

Основні принципи та обмеження REST:

- Клієнт-серверна архітектура: система розділена на дві частини - клієнт та сервер. Це дозволяє їм еволюціонувати незалежно один від одного.

- Відсутність стану (Statelessness): кожен запит від клієнта до сервера повинен містити всю необхідну інформацію для зрозуміння та обробки запиту. Сервер не повинен зберігати стан клієнта між запитами.

- Кешування: клієнти можуть кешувати відповіді сервера, щоб покращити продуктивність. Сервери можуть вказувати, чи можуть клієнти кешувати відповіді.

- Єдність ідентифікаторів (Uniform Interface): взаємодія між клієнтом і сервером повинна бути однорідною. Це включає чотири принципи:

- Ідентифікація ресурсів: Кожен ресурс повинен мати унікальний ідентифікатор (URI).

- Універсальний та зручний спосіб обміну інформацією через маніпуляцію ресурсами через представлення: клієнти можуть отримувати та взаємодіяти з ресурсами через представлення (наприклад, JSON або XML).

- Самоописані повідомлення: кожне повідомлення для його подальшої обробки повинно містити достатньо інформації.

- Гіпермедіа як рушій стану: клієнти які мають доступ для взаємодії, повині мати всі методи та сервіси за гіперпосиланнями.

- Шари (Layered System): система може бути розділена на рівні, причому кожен рівень взаємодіє тільки з прилеглими рівнями. Це дозволяє покращувати масштабованість та безпеку.

- Код на запит (Code on Demand): принцип, який дозволяє передавати та виконувати код на стороні клієнта. Наприклад, JavaScript, який виконується в браузері.

Переваги REST:

- Простота реалізації та розуміння.

- Легко інтегрується з іншими системами через HTTP.

- Єдність ідентифікаторів та стандартні HTTP-методи полегшують взаємодію.

- REST спрощує архітектуру, роблячи її більш масштабованою та простішою для розуміння. Його основна перевага полягає в тому, що він використовує стандартні HTTP-методи (GET, POST, PUT, DELETE), що полегшує розробку та розуміння. REST також забезпечує більш гнучку та легко розширювану архітектуру.

Недоліки REST:

- Недостатня підтримка для операцій, які вимагають стану.

- Може призвести до великої кількості запитів для складних дій.

REST залишається одним з найпопулярніших архітектурних шаблонів для створення веб-сервісів через свою простоту та гнучкість. REST може вважатися кращим варіантом ніж MVC через його простоту та інтеграцію з веб-стандартами. Він сприяє зменшенню залежності від сервера та дозволяє клієнтам отримувати ресурси у вигляді представлень (наприклад, JSON) через стандартні HTTP-запити. Це сприяє створенню більш гнучких, легко розширюваних та масштабованих систем.

2.8 PaLM

У якості сторонього API планується використовувати PaLM, для реалізації асистента, який зможе допомогати користувачам з різними аспектами і аналітикою по коментарям.

PaLM, що означає Pathways Language Model, є великою мовною моделлю, розробленою в Google AI. Вона була представлена у 2022 році і вважається однією з найбільших мовних моделей у світі. PaLM пройшла навчання на обсязі текстових та кодових даних, який налічує понад 600 мільярдів параметрів. Це надає їй здатність генерувати текст, виконувати переклад мов, творити різноманітний творчий вміст та відповідати на запитання інформативним способом. У порівнянні з іншими мовними моделями, PaLM має кілька переваг. Вона здатна генерувати більш реалістичний та складний текст, краще розуміє контекст запитань і використовує свої знання для відповіді на запитання більш інформативним способом.

PaLM все ще знаходиться в розробці, але вже проявила значний потенціал для використання в різноманітних додатках. Її можна використовувати для створення нових форм творчого вмісту, таких як вірші, код, сценарії, музичні твори, електронні листи та інше. Також PaLM може служити для перекладу мов, написання різноманітних творчих текстів та відповіді на запитання інформативним способом [29].

PaLM є потужною технологією, яка може вносити значний внесок у спосіб взаємодії з комп'ютерами, хоча вона все ще перебуває в стадії розробки, вона вже використовується для великих проектів корпорації Google, таких як Bard AI, приклад роботи якого зображено на рисунку 2.5.

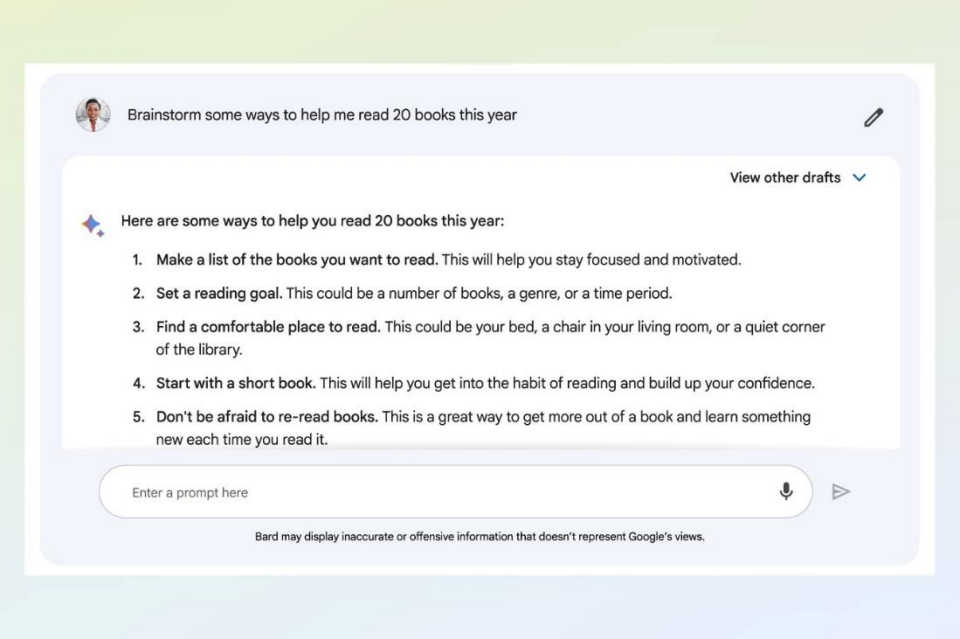

Рисунок 2.5 – Приклад роботи Bard AI який використовує PaLM

2.9 Apache Lucene

Apache Lucene – це потужний та високоефективний інструмент для повнотекстового пошуку та індексації даних. Основна мета Lucene – надати програмістам засоби для створення додатків, які потребують можливостей повнотекстового пошуку. Цей проект розробляється як відкрите програмне забезпечення Apache Software Foundation і написаний на мові Java.

Основні характеристики Apache Lucene:

- Індексація: Lucene забезпечує ефективні механізми для створення оберненого індексу, який дозволяє швидко знаходити документи, що містять певні терміни.

- Пошук: можливості потужного повнотекстового пошуку, який дозволяє виконувати різноманітні запити та фільтри для отримання точних результатів.

- Масштабованість: здатність масштабування забезпечує ефективну роботу індексів і пошуку навіть для великих обсягів даних.

- Можливості розширення: Lucene дозволяє розширювати та модифікувати його функціонал за допомогою різноманітних плагінів та розширень.

- Підтримка мов: має підтримку різних мов та алгоритмів аналізу тексту, що робить його гнучким для різноманітних вимог.

- Широке використання: застосовується в різних сферах, таких як веб–пошук, інтелектуальні системи, пошукові системи, бібліотеки, та інші області.

Apache Lucene використовується як базова технологія для інших проектів, включаючи Apache Solr та Elasticsearch, які надають додаткові можливості та інтерфейси для використання функціоналу Lucene в різних варіантах вирішення завдань повнотекстового пошуку [30]. Приклад загального процесу роботи Apache Lucene зображено на рисунку 2.6. Архітектура Apache Lucene зображена у додатку Б на рисунку Б.4.

2.10 Apache OpenNLP

Apache OpenNLP (Natural Language Processing) – це відкрите програмне забезпечення, розроблене Apache Software Foundation, яке надає інструменти та бібліотеки для обробки природної мови. OpenNLP дозволяє виконувати різноманітні завдання, такі як розпізнавання іменованих сутностей, розпізнавання частин мови, аналіз синтаксису, розпізнавання відносин та інші [31].

Основні можливості OpenNLP включають:

– Розпізнавання іменованих сутностей (NER): визначення та класифікація іменованих сутностей у тексті, таких як особи, місця, дати, організації та інші.

– Розпізнавання частин мови (POS): визначення частин мови для кожного слова у реченні, таких як іменники, прикметники, дієслова тощо.

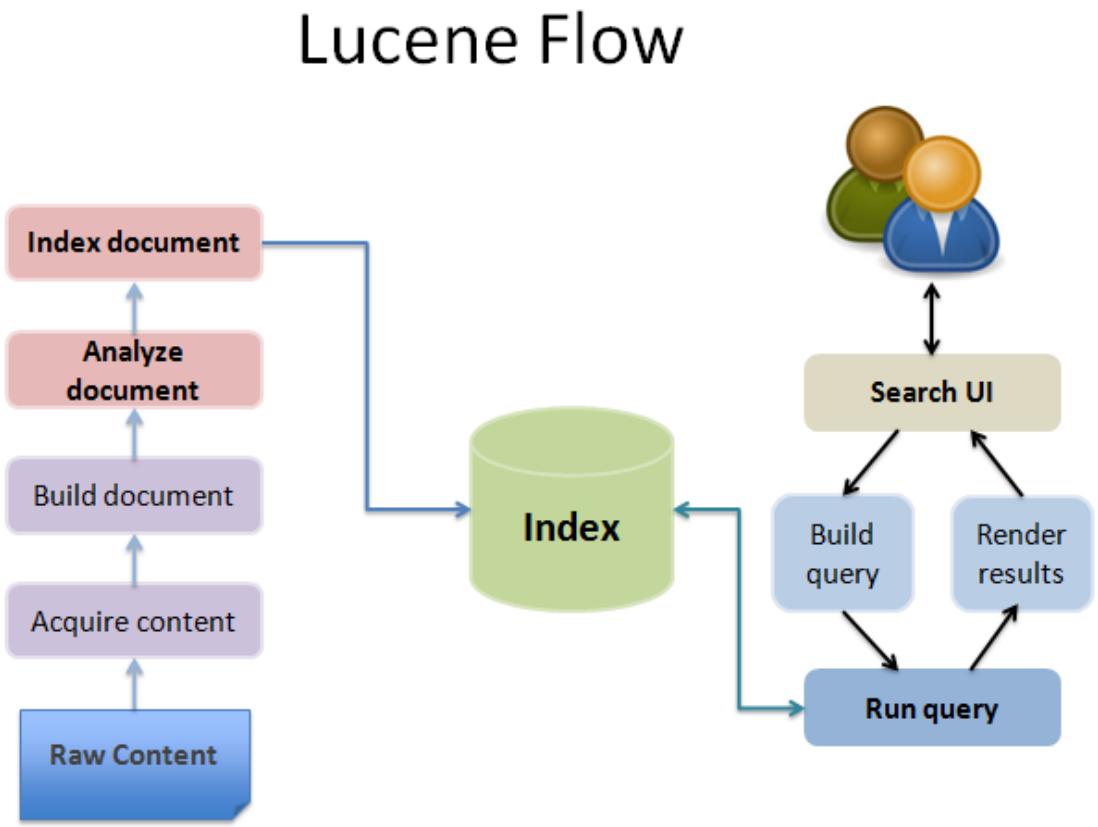

Рисунок 2.6 – Приклад загального процесу роботи Apache Lucene

– Аналіз синтаксису: визначення структури речення та залежностей між словами для розуміння синтаксису.

– Визначення відносин: виявлення відносин між сутностями у тексті, наприклад, визначення, що одна сутність є батьківською для іншої.

– Розпізнавання дат: визначення та екстракція інформації про дати та час з тексту.

– Токенізація: розділення тексту на токени (основні одиниці, такі як слова чи речення).

– Вивчення імпліцитних відносин: Аналіз тексту для виявлення імпліцитних відносин між сутностями.

OpenNLP надає легкий та ефективний спосіб розробки систем обробки природної мови для різноманітних завдань та додатків. Він може бути використаний в поєднанні з іншими інструментами та бібліотеками для створення комплексних рішень в області обробки природної мови.

Приклад застосування Apache OpenNLP зображений на рисунку 2.7.

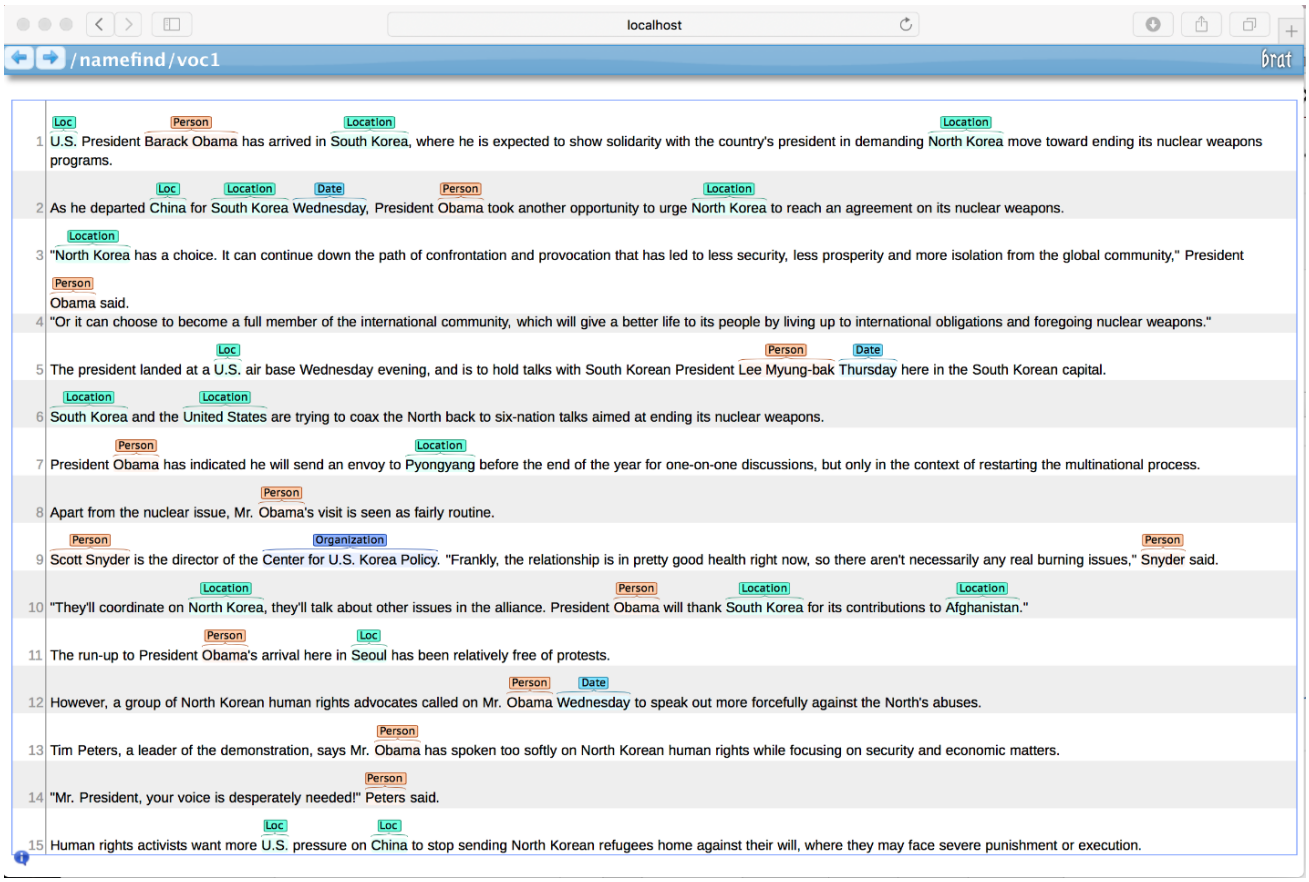

Рисунок 2.7 – Приклад застосування Apache OpenNLP

2.11 Висновки

У даному розділі було розглянуто і обґрунтовано вибір підходів та технологій для подальшої розробки системи. Були обрані перевірені та надійні підходи з популярними і затребуваними технологіями, які швидко та постійно розвиваються.

Також у даному розділі було обрано основні інструменти та мову програмування Java. Було обґрунтовано вибір цієї мови та інструментів для розробки системи.

3 ПРОЄКТУВАННЯ АРХІТЕКТУРИ СИСТЕМИ

Проєктування архітектури системи є вирішальним етапом у процесі створення програмного забезпечення, що передбачає важливі рішення щодо структури, взаємодії між компонентами та визначення внутрішніх взаємозв'язків. Цей етап відіграє ключову роль у визначенні ефективності, розширюваності та супроводження системи в подальшому.

Правильно спроєктована архітектура системи може значно спростити процес розробки, підвищити його стабільність та масштабованість. Вона дозволяє визначити модульність системи, сприяє врегулюванню взаємодій між її складовими частинами та встановлює загальну стратегію розвитку програмного продукту.

3.1 Визначення функціональних вимог системи

Функціональні вимоги визначають, які функції повинна виконувати система, які операції вона повинна виконувати та які функціональні можливості мають бути реалізовані в програмному продукті. FRD може включати в себе такі елементи:

Опис функцій:

1. Введення Даних: Система повинна дозволяти користувачам вводити коментарі для аналізу. Це може бути введення текстового вмісту або завантаження файлів з коментарями у форматі JSON.

2. Зберігання коментарів: система повинна мати можливість збереження коментарів для подальшого користування.

3. Визначення Ключових Слів: Виявлення та виділення ключових слів або термінів у коментарях, що може бути корисним для подальшого аналізу.

4. Категоризація Тем: Система може категоризувати коментарі за темами чи тематикою, щоб забезпечити більше деталей щодо контексту аналізу.

5. Оцінка популярності та корисності відгуку: визначення рівня публічного відгуку для кожного коментаря, можливо, за допомогою показників популярності чи впливу.

6. Звіти та Візуалізація Даних: Передбачення можливості генерації звітів та візуалізації результатів аналізу для зручності користувачів.

7. Пошук та Фільтрація: Можливість користувачів використовувати пошук та фільтрацію для вибору конкретних груп коментарів або певного типу.

8. Інтеграція з Зовнішніми Джерелами: Здатність інтегруватися з іншими джерелами даних або зовнішніми API для отримання додаткової інформації для аналізу.

9. Безпека та Конфіденційність: Забезпечення заходів безпеки для зберігання та обробки коментарів, зокрема, якщо вони містять конфіденційну інформацію.

- Вимоги до введення та виведення даних: система може зчитувати дані, з текстового поля або з наданого файла у визначеному форматі .csv, в подальшому планується обробка коментарів у довільному форматі у файлах розширення .txt

- Обмеження системи: на даному етапі система працює лише з англійською мовою, та зчитує коментарі лише у визначеному форматі.

- Вимоги до продуктивності: Визначення стандартів продуктивності, таких як час відгуку системи, швидкість обробки даних тощо. У випадку даної системи вимоги до продуктивності визначаються найбільш поширеними характеристиками персональних комп’ютерів, які було описано в технічних вимогах.

Створення програми для аналізу та оцінки коментарів може включати різні функціональні вимоги в залежності від того, яку інформацію необхідно отримати з коментарів. Однак основні функціональні вимоги можуть виглядати приблизно так:

Ці функціональні вимоги можуть слугувати початковим каркасом для вашої системи аналізу та оцінки коментарів. Також варто зазначити, чи розглядається оцінка коментарів у контексті якогось конкретного проекту чи

дослідження. Це допоможе забезпечити більш точне і адаптоване до контексту перефразування.

В процесі проектування також була розроблена схема роботи програми, яку представлено у додатку Б на рисунку Б.5, яка описує загальну задачу та дії програми під час роботи.

3.2 Проєктування бази даних системи

База даних є не лише основною, але й ключовою складовою будь-якого програмного забезпечення, де необхідно ефективно зберігати, організовувати та управляти великим обсягом даних для подальшого повторного використання. Ця система створює надійний механізм для забезпечення доступу до інформації, а також забезпечує її цілісність та безпеку.

Ось деякі аспекти, які підкреслюють важливість баз даних у розробці програмного забезпечення:

- Зберігання та організація даних: база даних дозволяє ефективно зберігати та організовувати різноманітні типи даних, починаючи від текстової інформації до зображень та аудіофайлів. Вона створює структуроване середовище для зберігання та управління цими даними.

- Ефективний доступ до інформації: бази даних забезпечують швидкий та ефективний доступ до інформації, завдяки використанню індексів, запитів та оптимізації. Це дозволяє програмному забезпеченню працювати ефективно та швидко, навіть з великим обсягом даних.

- Цілісність та спільна використання: база даних гарантує цілісність даних, забезпечуючи, що вони зберігаються та змінюються таким чином, щоб уникнути суперечностей та помилок. Крім того, вона надає можливість спільного використання даних різними компонентами системи.

- Безпека та керування доступом: за допомогою різноманітних механізмів, таких як ролі, права доступу та шифрування, бази даних забезпечують безпеку

даних. Це особливо важливо для забезпечення конфіденційності та запобігання несанкціонованому доступу.

- Підтримка транзакцій: бази даних забезпечують підтримку транзакцій, що дозволяє виконувати групу операцій як єдину атомарну одиницю. Це забезпечує консистентність даних та можливість відновлення системи в разі виникнення помилок.

- Масштабованість та розширюваність: бази даних можуть бути масштабовані від невеликих систем до великих даних. Вони також дозволяють розширювати функціональність шляхом додавання нових таблиць та змінювання схеми бази даних.

Узагальнюючи, база даних є невід'ємною частиною розробки програмного забезпечення, що забезпечує ефективну та безпечну роботу з даними, що є важливою передумовою успішного функціонування сучасних інформаційних систем.

Зазвичай прийнято вважати що таблиця в базах даних це набір елементів даних, які організовані з використанням моделі вертикальних стовпчиків і горизонтальних рядків. Таблиця має визначену кількість стовпчиків, в той час як кількість рядків може бути різною в різні моменти. Кожен рядок ідентифікується унікальним ключем, або id (ідентифікатором).

У проєктованій базі даних будуть використовуватись наступні таблиці:

У додатку Б на рисунку Б.6 проілюстровано як виглядає спроєктована база даних, яка буде використовуватись системою, а саме таблиці та поля, що будуть використовуватись.

Детальний опис кожної таблиці подано нижче:

- Таблиця comment зберігає в собі основні елементи коментаря, а саме заголовок і текст, якщо такі є, вся інша інформація зберігається в comment_details.

- Таблиця comment_details зберігає в собі різну додаткову інформацію яку іноді можна знайти в коментарях, а саме оцінка, рейтинг коментарю, автора, сайт

на якому коментар був опублікований, номери, емейли, посилання, згадки назв країн, згадки локацій.

- Таблиця project_comment пов’язує таблиці comment та project.
- Таблиця project зберігає в собі назву і опис проєкта, з яким користувач працює. На кожен проєкт зберігаються коментарі які до цього проєкту відносяться. Це реалізовано для того щоб користувач міг зручно зберігати коментарі і ділити їх на різні задачі у вигляді проєктів.

3.3 Проєктування графічного інтерфейсу користувача

Графічний інтерфейс користувача (GUI) – це спосіб взаємодії користувача з програмним чи апаратним забезпеченням за допомогою графічних елементів, таких як іконки, кнопки, меню та вікна. GUI забезпечує інтуїтивний та зручний спосіб взаємодії з комп'ютером, роблячи використання програм та систем більш доступним для широкого кола користувачів.

Ось деякі важливі аспекти GUI та їх значення:

- Інтуїтивність та Легкість Використання: GUI дозволяє користувачам взаємодіяти з системою без необхідності вивчення складних команд або мов програмування. Елементи інтерфейсу, такі як кнопки та меню, можуть бути легко зрозумілі та використовувані без спеціальної підготовки.
- Візуалізація Інформації: GUI дозволяє візуалізувати дані та інформацію, роблячи їх більш зрозумілими для користувача. Графічні елементи, такі як графіки, діаграми та таблиці, можуть поліпшити сприйняття інформації.
- Мультимедіа: GUI дозволяє інтегрувати різноманітні мультимедійні елементи, такі як зображення, відео та звук, що покращує взаємодію користувача з системою та забезпечує більш насичений досвід.
- Навігація: GUI надає зручні засоби навігації для користувача, такі як меню, вкладки та гіперпосилання. Це допомагає швидко переміщатися між різними частинами програми чи системи.

- Підтримка Взаємодії: GUI дозволяє користувачам взаємодіяти з програмою або системою через введення за допомогою миші, клавіатури або інших пристроїв. Це робить взаємодію більш гнучкою та зручною.

- Естетика та Користувацький Досвід: Якщо GUI виглядає естетично та забезпечує позитивний користувацький досвід, це може позитивно впливати на сприйняття програми чи системи користувачами.

Узагальнено, GUI грає важливу роль у розробці програмного забезпечення, оскільки він сприяє зручності взаємодії та поліпшує користувацький досвід, що важливо для успіху програм та систем на ринку.

З усіх видів інтерфейсів, для архітектурного шаблону REST підходить саме веб-інтерфейс. Він є найбільш універсальним та кросплатформеним. Для його розробки будуть використовуватись HTML, CSS та JS. Інтерфейс повинен бути простим та зрозумілим для користувача та зручним доступом до всіх можливостей застосунку.

Приклад схеми GUI зображено на рисунку 3.2.

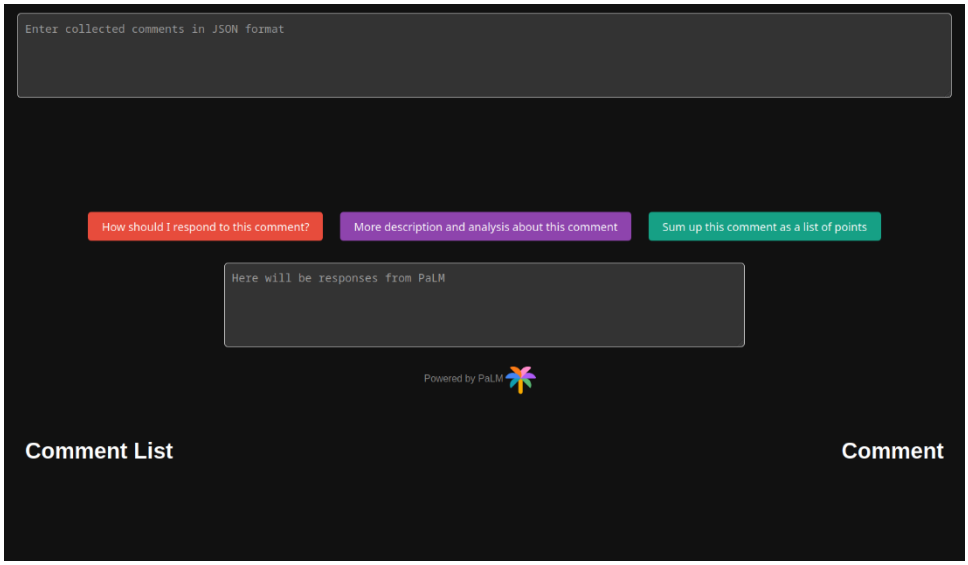

Рисунок 3.1 – Схема GUI

3.4 Висновки

У даному розділі було визначено функціональні вимоги до системи, які є початковим та основним етапом в розробці. А саме як система буде себе поводити в певних умовах, як здійснювати передачу даних та як система буде опрацьовувати ці дані. Також були описані функціональні особливості системи.

Було спроектовано базу даних, та описано які таблиці необхідно додати і які дані зберігати.

Також було описано проєктування графічного інтерфейсу, вимоги до нього, можливості та як зробити його зручним для користувача.

4 РОЗРОБКА ТА ТЕСТУВАННЯ СИСТЕМИ

Розробка та тестування системи належать до етапів, які мають критичне значення у життєвому циклі програмного забезпечення. Успішне виконання цих етапів визначає якість, функціональність та ефективність розробленої системи. Надійна реалізація цього процесу стає важливим кроком для досягнення зазначених цілей та вимог проекту.

Цей розділ спрямований на розгляд ключових аспектів, пов'язаних із розробкою та тестуванням системи, а також на висвітлення їх значущості у відповідному контексті створення програмного забезпечення. Від цих етапів залежить якість та придатність фінальної системи, оскільки вони дозволяють перевірити відповідність програмних рішень вимогам, а також виявити і виправити можливі недоліки чи помилки.

4.1 Інтеграція інструменту автоматичної збірки проекту

Для автоматизації та контролювання версій бібліотек що використовуються системою, будьмо використовувати інструмент автоматичної збірки проектів Maven. Також, це допомагає налаштувати автоматизацію порядку виконання запуску проекту, чи тестів, а також деяких команд для його збірки, або запуску окремих частин.

IntelliJ IDEA має вбудовану підтримку даного інструменту, тому процес налаштування є простим та зрозумілим. Також є можливість вибору додаткових функцій та можливостей

На рисунках 4.1 та 4.2 можна побачити діалогові вікна з вибором інструменту Maven, також ми вказуємо інші необхідні параметри для створення проекту.

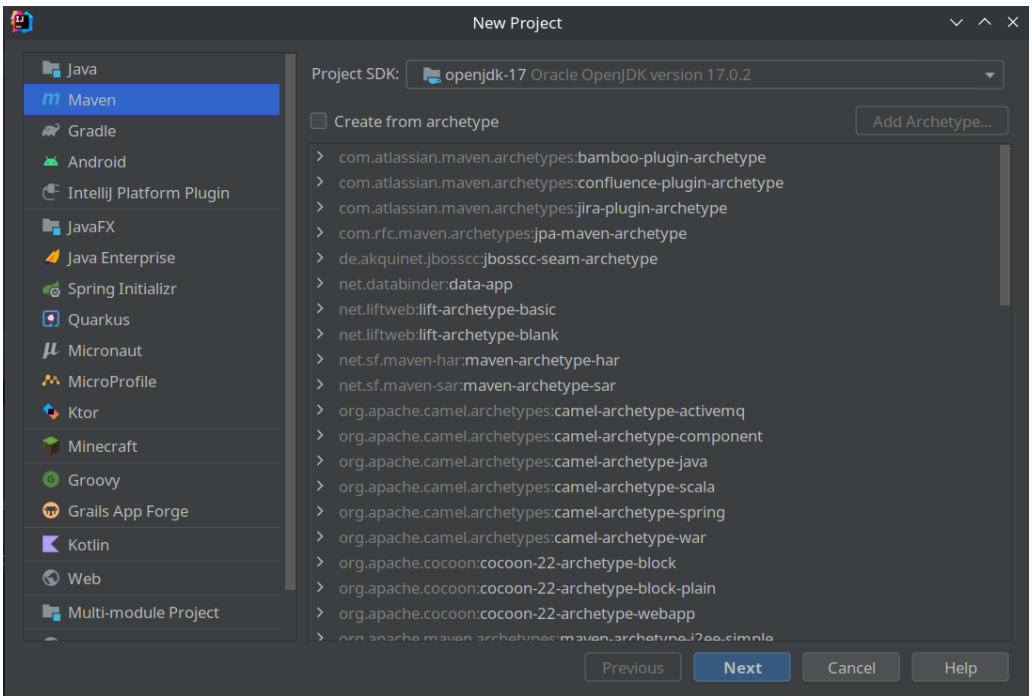

Рисунок 4.1 – Вибір Maven при створенні нового проекту

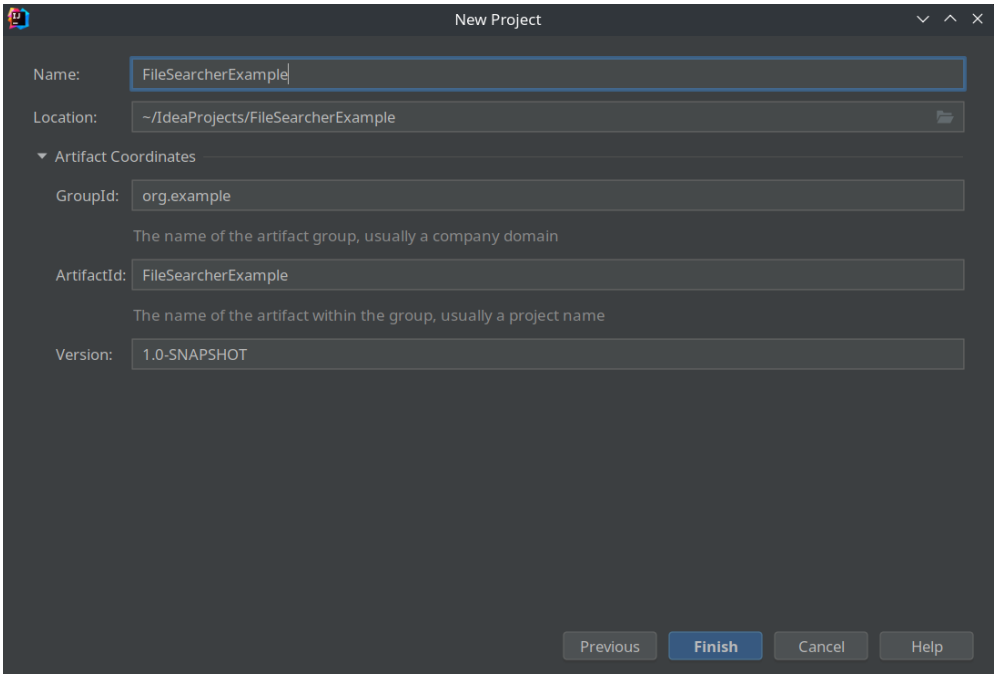

Рисунок 4.2 – Задання параметрів для Maven та підтвердження вибору

В результаті налаштування ми отримуємо автоматично згенерований скрипт в який можна додати потрібні бібліотеки.

4.2 Розробка програмного забезпечення

Розробка програмного забезпечення є складним процесом, і для ефективного його виконання існують різні підходи та правила. Ось деякі з них:

Основні правила:

- Розуміння Вимог: Чітке визначення та розуміння вимог до продукту перед початком розробки.
- Етапи та Документація: Визначення чітких етапів розробки та ведення документації на кожному етапі.
- Використання Систем Керування Версіями: Використання систем керування версіями, таких як Git, для відстеження та контролю версій програмного коду.
- Тестування та Якість Коду: Регулярне тестування та забезпечення високої якості програмного коду.
- Забезпечення Безпеки: Врахування аспектів безпеки на кожному етапі розробки для захисту від потенційних загроз.
- Командна Робота та Спілкування: Сприяння ефективній командній роботі та відкритому спілкуванню всередині команди.
- Спрощення та Перевірка Коду: Уникання зайвого складності та перевірка коду для виявлення можливих проблем та вдосконалень.
- Ітерації та Відгук: Ітеративний підхід до розробки з регулярним отриманням та врахуванням відгуку користувачів.

Ці підходи та правила можуть бути адаптовані відповідно до конкретних потреб та характеристик проекту. Важливо визначити оптимальний підхід для конкретної розробки, враховуючи її обсяг, складність та вимоги.

Основні підходи:

- Гнучка розробка (Agile Development): Розвиток продукту через ітерації, акцент на співпраці команди, інкрементальному вдосконаленні та відгуку користувачів.

- Каскадна розробка (Waterfall Development): Послідовний процес розробки, де кожен етап завершується, перш ніж розпочинається наступний.

- Методологія DevOps: Інтеграція розробки та операцій для автоматизації процесів випуску та розгортання програм.

- Модель V–образу (V–Model): Послідовна модель, яка підтримує паралельний процес тестування та розробки.

- Мікросервісна архітектура: Розробка програми у вигляді невеликих, незалежних сервісів для полегшення масштабування та розвитку.

Під час розробки даної системи планується використовувати модель V–образу, так як це найбільш зручний варіант для конкретної задачі.

Наведемо нижче основні переваги цього підходу:

- Чітке визначення етапів: V–Model визначає чіткі етапи розробки, відповідні та дзеркально відображені для кожного етапу тестування. Це сприяє легкості розуміння та виконання процесу.

- Покращена передбачуваність: завдяки відповідності етапів розробки та тестування, можна передбачати, що вимоги будуть виконані на кожному етапі, сприяючи покращеній передбачуваності результатів.

- Спрощення управління проектом: модель V–образу дозволяє легко визначити завдання та взаємозв'язки між різними етапами, що спрощує управління проектом.

- Зниження ризиків: послідовність етапів та відповідність вимогам на кожному етапі сприяють ранньому виявленню та виправленню помилок, зменшуючи ризик невірного розуміння вимог [32].

Принципи, методи та їх візуальний вигляд наведено на рисунку 4.3

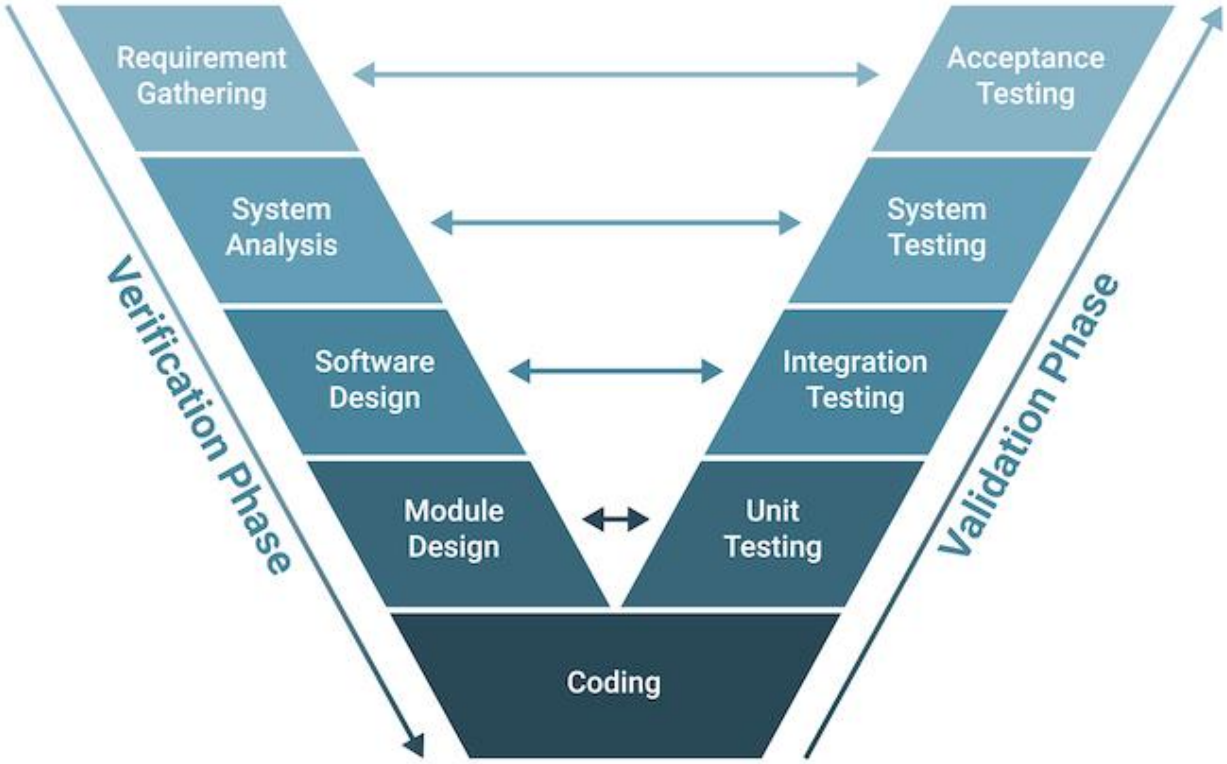

Рисунок 4.3 – Структура V-образу

На рисунку 4.4 зображено розроблений метод для спілкування з PaLM API.

```
@SneakyThrows
public String curlToGoogleModel(String prompt) {
    URL url = new URL( spec: "https://generativelanguage.googleapis.com/v1beta3/models/text-bison-001:generateText?

    HttpURLConnection httpConnection = (HttpURLConnection) url.openConnection();
    httpConnection.setRequestMethod("POST");
    httpConnection.setRequestProperty("Content-Type", "application/json");
    httpConnection.setDoOutput(true);

    OutputStreamWriter writer = new OutputStreamWriter(httpConnection.getOutputStream());
    writer.write( str: "{ \"prompt\": { \"text\": \"" + prompt + "\" } }");
    writer.flush();
    writer.close();
    httpConnection.getOutputStream().close();

    InputStream responseStream = httpConnection.getResponseCode() / 100 == 2
            ? httpConnection.getInputStream()
            : httpConnection.getErrorStream();
    Scanner scanner = new Scanner(responseStream).useDelimiter( pattern: "\\A");
    return scanner.hasNext() ? scanner.next() : "";
}

public CommentResponse getComment(Long id) { return CommentResponse.builder().build(); }
}
```

Рисунок 4.4 – Код методу для спілкування з PaLM API

4.3 Тестування програмного забезпечення

Тестування представляє собою невід'ємну та ключову складову процесу розробки програмного забезпечення, оскільки його основна мета полягає в виявленні помилок, забезпеченні відповідності функціоналу вимогам та гарантуванні високої якості продукту. Тестування є ефективним інструментом для перевірки функціональності, надійності та коректності роботи програмного забезпечення перед його випуском на ринок чи в експлуатацію. Виявлення та усунення помилок на етапі тестування дозволяє знизити ймовірність виникнення проблем після впровадження програмного продукту.

Основні правила:

- Визначення цілей тестування: чітке визначення цілей тестування та його обсягу на кожному етапі розробки.

- Покриття тестування: покриття якнайбільшої кількості коду тестами для впевненості в його стабільності та правильності.

- Повторне використання тестових наборів: використання попередньо створених тестових наборів для забезпечення стабільності та уніфікації тестування.

- Тестові дані: використання реалістичних тестових даних для відображення реальних умов експлуатації.

- Регулярне виконання тестів: регулярне та автоматизоване виконання тестів для виявлення проблем якнайшвидше.

- Відслідковування та аналіз помилок: систематичний аналіз та відстеження помилок для покращення якості продукту та процесу розробки.

Працюючи з цими підходами та правилами, команди розробників можуть ефективно забезпечити якість програмного забезпечення та зменшити ймовірність виникнення помилок під час розробки продукту.

Основні підходи:

- Модульне тестування (Unit Testing): тестування окремих модулів або компонентів програми для перевірки їх правильності та функціональності.

- Інтеграційне тестування (Integration Testing): перевірка взаємодії між різними модулями або сервісами для впевненості, що вони працюють разом правильно.

- Системне тестування (System Testing): тестування всієї системи як єдиної одиниці для перевірки відповідності вимогам та функціональності системи в цілому.

- Приймальне тестування (Acceptance Testing): визначення того, чи відповідає продукт вимогам та чи задовольняє потребам користувачів.

- Автоматизоване тестування: використання автоматизованих інструментів для виконання тестів, що допомагає збільшити ефективність та швидкість тестування.

- Тестування безпеки: перевірка системи на вразливості та забезпечення заходів безпеки.

У даній системі передбачається інтеграція різноманітних підходів до тестування, описаних вище, з пріоритетним використанням модульного тестування та системного тестування в поєднанні з автоматизованим тестуванням. Модульне тестування спрямоване на перевірку окремих компонентів програми з метою визначення їхньої функціональності та коректності роботи незалежно від інших частин системи. Системне тестування, у свою чергу, ставить за мету перевірку роботи всієї системи як єдиного цілого, переконуючись у відповідності системи вимогам та взаємодії всіх компонентів між собою. Крім того, впроваджується автоматизоване тестування, яке дозволяє виконувати тестові скрипти та процеси автоматично, спрощуючи та прискорюючи процес перевірки програмного забезпечення. Такий комплексний підхід сприяє покращенню якості тестування та забезпечує високу стабільність та надійність розроблюваної системи [33].

Піраміда етапів тестування показана на рисунку 4.3

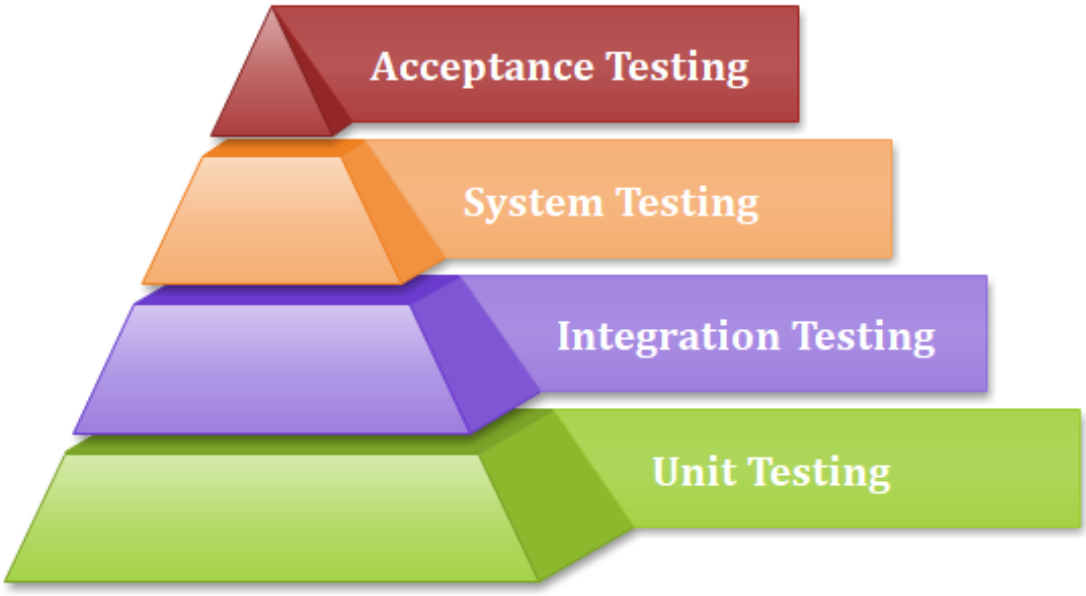

Рисунок 4.6 – Піраміда етапів тестування

Для проведення модульного тестування обрано JUnit, який є фреймворком для автоматизованого тестування в контексті розробки програмного забезпечення на платформі Java. Цей інструмент зосереджений на впорядкуванні та виконанні тестів для перевірки коректності функціональності програмного коду. У науковому стилі відзначається рядом важливих аспектів:

- Анотації: JUnit використовує механізм анотацій мови програмування Java для позначення тестових методів. Анотація @Test служить ключовим індикатором, вказуючи, що конкретний метод є тестом.

- Асерти: для формалізованої перевірки очікуваних результатів тестів використовуються асерти, які представлені вбудованими методами JUnit. вони дозволяють порівнювати значення та перевіряти умови для ефективного тестування.

- Фіксація стану: існує можливість вказати методи, які повинні виконуватися перед (@Before) та після (@After) кожного тестового методу, надаючи можливість встановлення чи очищення необхідних ресурсів.

- Сюїти тестів: JUnit дозволяє об'єднувати тести в сюїти (test suites), що забезпечує організаційну структуру для зручного виконання групи тестів.

- Параметризовані тести: можливість параметризації тестів дозволяє використовувати різні вхідні дані для одного тесту, підвищуючи гнучкість та покращуючи покриття.

JUnit визначає стандарти та інструменти, спрямовані на покращення процесу розробки, забезпечуючи високу надійність та стабільність програмного коду через автоматизоване тестування. Приклад написання тестів та приклад успішного тестування наведено нижче на рисунках 4.5 та 4.6.

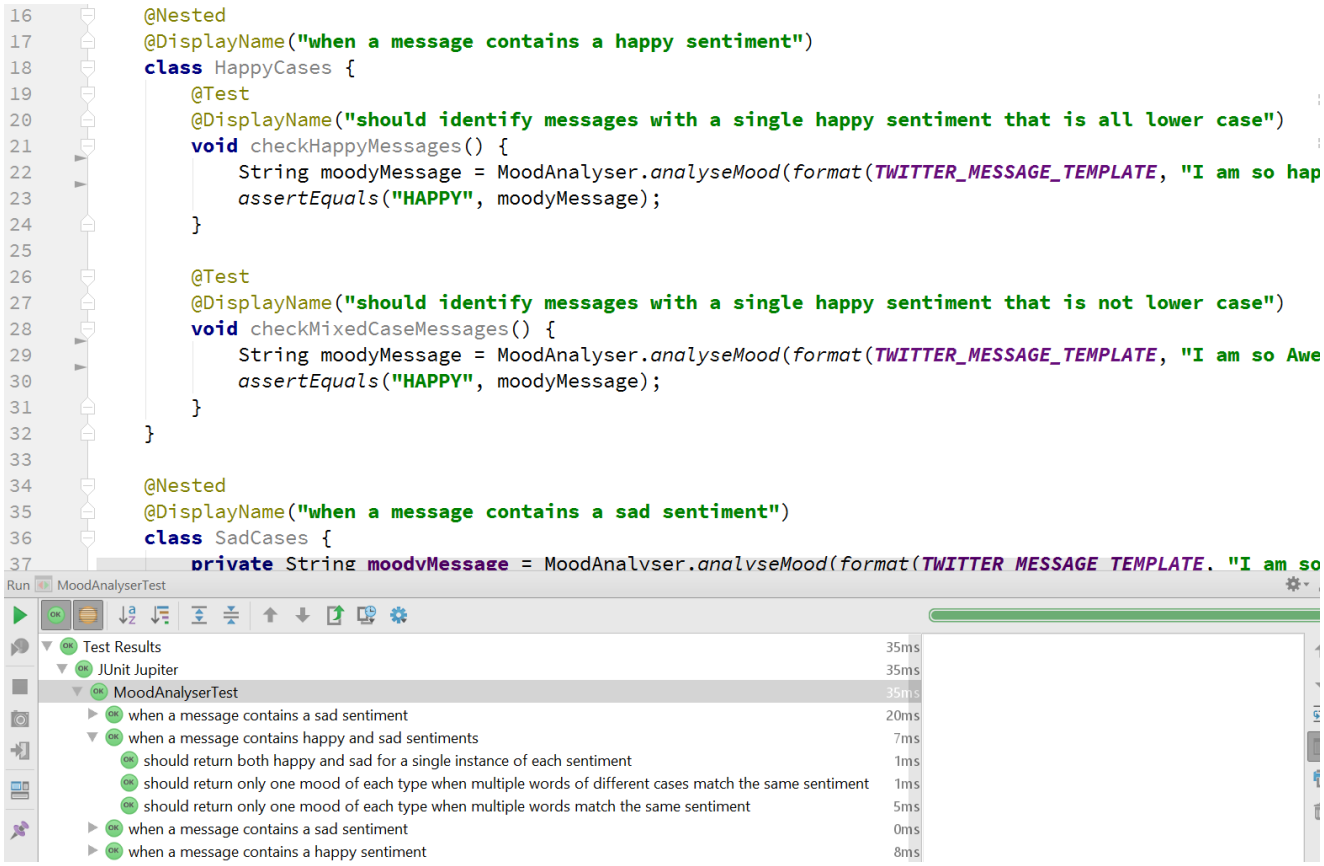

Рисунок 4.7 – Написання тестів

Рисунок 4.8 – Успішне виконання тестів

4.4 Перспективи вдосконалення системи у майбутньому

З метою подальшого вдосконалення системи рекомендується врахувати можливість додати до неї наступні функціональні можливості, які могли б бути корисними для користувачів:

- Аналіз емоційного тону: додатковий функціонал, що дозволить системі проводити аналіз емоційного забарвлення коментарів, визначаючи, наприклад, чи відноситься кожен коментар до позитивного, негативного або нейтрального контексту.

- Обробка коментарів у різних форматах: система матиме можливість ефективно обробляти коментарі в різноманітних форматах, автоматично визначаючи способи розпізнавання та зберігання інформації в них.

- Фільтрація грубого або неприпустимого вмісту: врахування можливості системи розпізнавати та фільтрувати використання грубої лексики, неприпустимих слів чи нецензурного вмісту в коментарях, забезпечуючи відповідне повідомлення користувачам щодо такого змісту.

4.5 Порівняння розробленої системи з аналогами

Порівняння роботи полягає в тому, що запропоноване програмне забезпечення для аналізу та пошуку коментарів, яка за рахунок використання мови програмування Java та таких фреймворків як Spring, Hibernate, а також мовній моделі PaLM, на відміну від існуючих засобів та інструментів, дозволяє розширити функціональні можливості програмного забезпечення, а саме надати кросплатформеність, суттєво спростити, розширити та зробити більш точним аналіз коментарів, а також надати поради користувачам як взаємодіяти з коментарями. Для цього було проведено порівняння розробленого програмного забезпечення з аналогічними засобами та інструментами. Результати порівняння наведено в таблиці 4.2.

Таблиця 4.2 – Результати порівняння розробленої системи з аналогами

Фунціонал	Можливості браузера	Можливості OpenKM	Можливості Repustate	Розроблена система
Гнучкість налаштування	X	✓	✓	✓
Простота налаштування	✓	X	X	✓
Кросс-платформеність	✓	X	✓	✓
Підтримка читання великих обсягів даних	X	✓	✓	✓
Мовна модель в якості асистенту	X	X	X	✓
Робота з великим навантаженням	X	✓	✓	✓
Пошук серед конкретних коментарів	✓	✓	X	✓
Можливість аналізу кожного коментаря окремо	X	X	X	✓

Як можна побачити, мета роботи була досягнута через отримання підтвердження шляхом порівняння функціоналу розробленої системи з аналогами. Переважна кількість аналогів має неповний функціонал для вирішення задачі валідації подій.

4.6 Висновки

У даному розділі було описано процес розробки та тестування даної системи. А саме підходи до розробки і тестування, які є зручними та ефективними. Велика увага приділялась архітектурній складовій та гнучкості системи, а також перевірці правильності роботи за допомогою тестування.

Також у даному розділі було описано можливі покращення системи в майбутньому що є важливим етапом для подальшого вдосконалення.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Технологічний аудит розробленої системи пошуку та аналізу відгуків і коментарів

В наш час робота з інформацією є невід'ємною частиною життя і діяльності кожної людини. На роботі, вдома чи під час навчання люди переглядають, надсилають, редагують, зберігають, коментують файли, тексти тощо, тобто мають справу з різноманітною інформацією. Особливість цього процесу полягає також у тому, що необхідно забезпечити не тільки точну передачу самої інформації, яка цікавить споживача, але й забезпечити зручну обробку та аналіз відгуків на цю інформацію, що надходить з різних джерел. Складності додає і те, що коментарі та відгуки можуть мати різну структуру та вигляд, можуть використовувати різні мови тощо, і це все потрібно швидко обробити і надати користувачу зручну інформацію та аналітику по його запит.

Тому метою виконаної нами магістерської кваліфікаційної роботи було розроблення системи (програмного забезпечення), яка б суттєво покращувала аналізування відгуків і коментарів, що з'являються на ту чи іншу інформацію.

Для досягнення поставленої мети нами було: проведено порівняльний аналіз схожих платформ; порівняно відомі методи аналізу інформації (текстової, зображення тощо); розроблено загальну структуру системи для роботи з відгуками та коментарями і інтерфейс користувача для цієї системи; розроблено програмне забезпечення з використанням мови програмування Java; проведено тестування системи.

В результаті було створено новий програмний засіб та розроблено алгоритми роботи з інформацією, які забезпечать зручний пошук, аналіз та інші дії, що дозволять кінцевим користувачам швидко та ефективно знаходити необхідну інформацію, а також відгуки на неї та коментарі.

Для встановлення комерційного потенціалу розробленої нами системи пошуку та аналізу відгуків і коментарів було запрошено 3-х відомих і знаних в

Україні експертів: д.т.н., професора Квєтного Р.Н., к.т.н., доцента Гармаша В.В. та к.т.н., доценткиню Барабан М.В. Встановлення комерційного потенціалу розробленої нами системи пошуку та аналізу відгуків і коментарів і було здійснено за критеріями, наведеними в таблиці 5.1.

Таблиця 5.1 – Рекомендовані критерії оцінювання технічного рівня та комерційного потенціалу будь-якої розробки і їх бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
	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 роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
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

Продовження таблиці 5.1

Ринкові перспективи					
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
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Критерії оцінювання та бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Практична 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 і з 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

Продовження таблиці 5.1

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
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

Запрошені експерти оцінили розроблену нами систему пошуку та аналізу відгуків і коментарів так, як показано в табл. 5.2.

Таблиця 5.2 – Результати технологічного аудиту розробленої нами системи пошуку та аналізу відгуків і коментарів (за шкалою оцінювання «0»-«1»-«2»-«3»-«4»)

Критерії	Прізвище, ініціали експертів		
	Квєтний Р.Н.	Гармаш В.В.	Барабан М.В.
	Бали, що їх виставили експерти:		
1	4	3	3
2	3	3	4
3	4	3	3
4	3	4	3
5	4	3	4
6	3	3	3
7	4	3	4
8	4	4	4
9	4	4	3
10	3	3	3
11	4	4	3
12	4	3	4
Сума балів	$СБ_1 = 44$	$СБ_2 = 40$	$СБ_3 = 41$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{3} = \frac{44+40+41}{3} = \frac{125}{3} = 41,67$		

Встановлення комерційного потенціалу розробленої нами системи пошуку та аналізу відгуків і коментарів було зроблено на основі рекомендацій, наведених в таблиці 5.3 [34].

Таблиця 5.3 – Рівні комерційного потенціалу будь-якої наукової розробки

Середньоарифметична сума балів $\overline{СБ}$, 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

Оскільки середньоарифметична сума балів, що їх виставили експерти, складає 41,67 балів, то це свідчить, що розроблена нами система пошуку та аналізу відгуків і коментарів має рівень комерційного потенціалу, який вважається «високим». Це пояснюється тим, що розроблена нами система об’єднує в собі особливості та інструменти різних платформ для роботи з відгуками та коментарями і їх аналізуванням, а також дає можливість працювати з файлами, поєднуючи різні інструменти для різних типів та взаємодіяти з іншими користувачами.

5.2 Розрахунок витрат на розроблення системи пошуку та аналізу відгуків і коментарів

При розробленні системи пошуку та аналізу відгуків і коментарів були зроблені певні витрати. Зокрема:

А). Основна заробітна плата $З_о$ розробників, яка визначається за формулою:

$$З_о = \frac{М}{Т_р} \cdot t \quad \text{грн}, \qquad (5.1)$$

де М – місячний посадовий оклад розробника, грн; приймемо, що

М = (6700…24200) грн/місяць;

$T_р$ – число робочих днів в місяці; приймемо $T_р$ = 22 дні;

t – число днів роботи розробників.

Зроблені розрахунки зведемо до таблиці 5.4.

Таблиця 5.4 – Основна заробітна плата розробників

Найменування посади виконавця	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на оплату праці, грн
1. Науковий керівник магістерської роботи	22000	1000,00	20 годин	3333,33 ≈ 3334
2. Магістрант-студент-виконавець	2000 (беремо 6700)	304,54	72	21926,88 ≈ 21927
3. Консультант з економічної частини	17996	818,00	1,5 години	204,50 ≈ 202 (при 6-годинному робочому дні)
4.Фахівець бібліотеки	14000	636,36	1	≈ 637
Загалом				$З_о$ = 26 100 грн

Б). Додаткова заробітна плата $З_д$ розробників розраховується як (10…12)% від величини їх основної заробітної плати, тобто:

$$З_д = \alpha \cdot З_о = (0{,}1...0{,}12) \cdot З_о \qquad (5.2)$$

Приймемо, що α = 0,114. Тоді для нашого випадку отримаємо:

$$З_д = 0{,}114 \times 26100 = 2975{,}40 \approx 2976 \text{ грн.}$$

В). Нарахування на заробітну плату $НЗП_{зп}$ розробників (дослідників) розраховуються за формулою:

$$НЗП_{зп} = (З_о + З_д) \cdot \frac{\beta}{100}, \quad (5.3)$$

де β – ставка обов'язкового єдиного внеску на державне соціальне страхування, %. β = 22%. Тоді:

$$НЗН_{зп} = (26100 + 2976) \times 0,22 = 6396,72 \approx 6397 \text{ грн.}$$

Г). Амортизація основних засобів А, які використовувались під час виконання цієї роботи:

$$А = \frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12} \quad \text{грн}, \quad (5.4)$$

де Ц – загальна балансова вартість основних засобів, грн;

$Н_а$ – річна норма амортизаційних відрахувань; $Н_а$ = (2,5...25)%;

Т – термін використання основних засобів, місяці.

Зроблені розрахунки зведено в таблицю 5.5.

Таблиця 5.5 – Розрахунок амортизаційних відрахувань

Найменування обладнання, приміщень тощо	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань, грн
1. Комп'ютерна техніка, обладнання тощо	31500	25	3,3 (при 75% використанні	1624,22 ≈ 1625
2. Приміщення університету, кафедри	11300	3,5	3,3 при 65% використанні	70,69≈ 71
Всього				А = 1696 грн

Д). Витрати на матеріали М розраховуються за формулою:

$$М = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i - \sum_{1}^{n} В_i \cdot Ц_в \quad грн., \tag{5.5}$$

де $Н_i$ – витрати матеріалу *і*-го найменування, кг; $Ц_i$ – вартість матеріалу *і*-го найменування; $К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15); $В_i$ – маса відходів матеріалу *і*-го найменування; $Ц_в$ – ціна відходів матеріалу *і*-го найменування; n – кількість видів матеріалів.

Е). Витрати на комплектуючі К розраховуються за формулою:

$$К = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i \quad грн, \tag{5.6}$$

де $Н_i$ – кількість комплектуючих *і*-го виду, шт.; $Ц_i$ – ціна комплектуючих *і*-го виду; $К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15); n – кількість видів комплектуючих.

Під час виконання роботи загальні витрати на матеріали та комплектуючі склали приблизно 1120 грн.

Ж). Витрати на силову електроенергію $В_е$ розраховуються за формулою:

$$В_е = \frac{В \cdot П \cdot Ф \cdot К_п}{, \ К_д} \tag{5.7}$$

де В – вартість 1 кВт-год. електроенергії, в 2023 р. В ≈ 4,5 грн/кВт;

П – установлена потужність обладнання, кВт; П = 0,79 кВт;

Ф – фактична кількість годин роботи обладнання, годин.

Приймемо, що Ф = 293 годин;

$К_п$ – коефіцієнт використання потужності; $К_п$ < 1 = 0,74.

$К_д$ – коефіцієнт корисної дії, $К_д$ = 0,63.

Тоді витрати на силову електроенергію будуть дорівнювати:

$$В_е = \frac{В \cdot П \cdot Ф \cdot К_п}{К_д} = \frac{4,5 \cdot 0,79 \cdot 293 \cdot 0,74}{0,63} = 1223,48 \approx 1224 \text{ грн.}$$

И). Інші витрати $В_{інш}$ можна прийняти як (50…300)% від основної заробітної плати розробників, тобто:

$$В_{інш} = (0,5....3) \times З_о. \quad (5.8)$$

Для нашого випадку отримаємо:

$$В_{інш} = 0,6 \times 26100 = 15660 \text{ грн.}$$

К). Сума всіх попередніх статей витрат складає витрати на виконання роботи безпосередньо розробником-магістрантом – В.

$$В = 26100 + 2976 + 6397 + 1696 + 1120 + 1224 + 15660 = 55173 \text{ грн.}$$

Л). Загальні витрати на розроблення системи пошуку та аналізу відгуків і коментарів $В_{заг}$ становлять:

$$В_{заг} = \frac{В}{\beta} \quad (5.9)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання цієї роботи. Можна прийняти, що, $\beta \approx 0,91$ [31], оскільки робота практично завершена.

Тоді: $В_{заг} = \frac{55173}{0,91} = 60629,57$ грн. або приблизно 61 тисяча грн.

Тобто прогнозовані загальні витрати на розроблення системи (програмного забезпечення) пошуку та аналізу відгуків і коментарів становлять приблизно 61 тисячу грн.

5.3 Розрахунок економічного ефекту від можливої комерціалізації розробки

Економічний ефект від можливої комерціалізації розробленої нами системи (програмного забезпечення) пошуку та аналізу відгуків і коментарів виявляється у її значно кращих функціональних можливостях, а також у тому, що створені нами програмний засіб та алгоритми роботи з інформацією для зручного пошуку, аналізу та інших дій дозволяють кінцевим користувачам ефективно знаходити необхідну інформацію.

Попит на нашу розробку складають низка компаній, які займаються рекламою, бізнесом, торгівлею. Розробка також актуальна в сфері науки і просто для персонального застосування.

Аналіз ринку також показав, що потенційна кількість користувачів аналогічних систем становить сьогодні приблизно 40 осіб і буде постійно зростати. Тобто, якщо наша розробка буде впроваджена з 1 січня 2024 року, то її результати будуть виявлятися протягом 2024-го, 2025-го та 2026-го років.

Прогноз зростання попиту на нашу розробку складає по роках:

а) 2024 р. – + 10 шт. до базового року;

б) 2025 р. – + 20 шт. до базового року;

в) 2026 р. – + 30 шт. до базового року.

У 2022 році подібна за функціями розробка коштувала на ринку приблизно 60 тисяч грн. Тоді дану, значно кращу за своїми функціями розробку, можна буде реалізовувати на ринку дорожче, в середньому за 70 тисяч грн або на 10 тисяч грн дорожче.

За твердженням експертів, загальна сума прибутку для власника подібної розробки становитиме 480 тисяч грн за перший рік (при 100-ти користувачах). При більшій кількості користувачів величина прибутку буде відповідно вищою.

Проведемо більш точні розрахунки.

Можливе збільшення чистого прибутку $\Delta П_i$, що його може отримати потенційний інвестор від виведення даної розробки на ринок, становитиме:

$$\Delta П_i = \sum_1^n (\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\upsilon}{100}) \qquad (5.10)$$

де $\Delta Ц_o$ – покращення основного якісного показника від впровадження результатів нашої розробки у цьому році. Для даного випадку це є збільшення ціни реалізації нашої розробки $\Delta Ц_o = 70 – 60 = 10$ тисяч грн;

N – основний кількісний показник, який визначає обсяг діяльності у році до впровадження результатів розробки; N = 40 шт.;

ΔN – покращення основного кількісного показника від впровадження результатів розробки.

Таке покращення становитиме по роках, відповідно: у 2024 році – + 10 шт., у 2025 році + 20 шт., та у 2026 році + 30 шт.;

$Ц_o$ – основний якісний показник (тобто ціна), який визначає обсяг діяльності у році після впровадження результатів розробки, грн; $Ц_o$= 70 тисяч грн;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки; для нашого випадку n = 3;

λ – коефіцієнт, який враховує сплату податку на додану вартість; $\lambda = 0{,}8333$;

ρ – коефіцієнт, який враховує рентабельність продукту. Рекомендується приймати ρ = (0,2...0,5); візьмемо ρ = 0,5;

υ – ставка податку на прибуток. У 2023-26 роках υ = 18% (припущення).

Тоді можливе зростання чистого прибутку $\Delta П_1$ для потенційного інвестора протягом першого року від можливого впровадження і комерціалізації нашої розробки (2024 р.) становитиме:

$$\Delta П_1 = [10 \cdot 40 + 70 \cdot 10] \cdot 0{,}8333 \cdot 0{,}5 \cdot (1 - \frac{18}{100}) \approx 376 \quad \text{тис. грн,}$$

що практично збігається з висновками експертів, які прогнозувати загальний (не чистий) прибуток протягом першого року у 480 тисяч грн ($12000).

Можливе зростання чистого прибутку $\Delta П_2$ для потенційного інвестора від можливого впровадження і комерціалізації даної розробки протягом другого (2025) року складе:

$$\Delta П_2 = [10 \cdot 40 + 70 \cdot 20] \cdot 0,8333 \cdot 0,5 \cdot (1 - \frac{18}{100}) \approx 615 \quad \text{тис. грн,}$$

що практично збігається з висновками експертів, які прогнозувати загальний (не чистий) прибуток протягом другого року у 900 тисяч грн ($24000).

Можливе зростання чистого прибутку $\Delta П_3$ для потенційного інвестора від можливого впровадження та комерціалізації даної розробки протягом третього (2026) року складе:

$$\Delta П_3 = [10 \cdot 40 + 70 \cdot 30] \cdot 0,8333 \cdot 0,5 \cdot (1 - \frac{18}{100}) \approx 855 \quad \text{тис. грн.}$$

Приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації нашої розробки для інвестора становитиме:

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

Де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої роботи, грн;

т – період часу, протягом якого виявляються результати впровадженої роботи, роки. Для даного випадку т = 3 роки;

τ – ставка дисконтування. Приймемо τ = 0,10 (10%);

t – період часу від моменту початку розроблення системи (програмного забезпечення) пошуку та аналізу відгуків і коментарів до моменту отримання можливих чистих прибутків потенційним інвестором.

Тоді приведена вартість зростання всіх можливих чистих прибутків ПП, що їх може отримати потенційний інвестор від впровадження та комерціалізації нашої розробки, складе:

$$ПП = \frac{376}{(1+0,1)^2} + \frac{615}{(1+0,1)^3} + \frac{855}{(1+0,1)^4} \approx 311 + 462 + 584 = 1357 \text{ тисяч грн.}$$

Теперішня вартість інвестицій PV, що повинні бути вкладені у реалізацію нашої розробки: $PV = (1,0…5,0) \times В_{заг}$.

Для нашого випадку $PV = (1,0…5,0) \times 61 = 5,0 \times 55 = 305$ тисяч грн.

Абсолютний ефект від можливих вкладених в реалізацію даної розробки інвестицій $Е_{абс}$.

$$Е_{абс} = ПП - PV, \quad (5.12)$$

де ПП – приведена вартість збільшення всіх чистих прибутків для потенційного інвестора від можливої комерціалізації даної розробки, грн;

PV – теперішня вартість інвестицій PV = 305 тисяч грн.

Абсолютний ефект від можливого впровадження даної розробки складе:

$$Е_{абс} = 1357 - 305 = 1052 \text{ тисяч грн.}$$

Далі розрахуємо внутрішню дохідність $Е_в$ вкладених інвестицій:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1 \quad (5.13)$$

де $Е_{абс}$ – абсолютний ефект вкладених інвестицій; $Е_{абс}$ = 1052 тис. грн;

PV –теперішня вартість початкових інвестицій PV = 305 тис. грн;

$Т_ж$ – життєвий цикл розробки, роки.

$Т_ж$ = 4 років (2023-й, 2024-й, 2025-й, 2026-й роки)

Для нашого випадку отримаємо:

$$Е_в = \sqrt[4]{1 + \frac{1052}{305}} - 1 = \sqrt[4]{1 + 3,4492} - 1 = \sqrt[4]{4,4492} - 1 = 1,452 - 1 = 0,452 = 45,2\%.$$

Далі визначимо ту мінімальну дохідність, нижче за яку потенційному інвестору не вигідно буде займатися комерціалізацією нашої розробки.

Мінімальна дохідність або мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau_{мін} = d + f, \quad (5.14)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2022-2023 роках в Україні d = (0,10...0,12);

f – показник, що характеризує ризикованість вкладень;

f = (0,1...0,50). Приймемо f = 0,30.

Для нашого випадку отримаємо:

$$\tau_{мін} = 0,12 + 0,30 = 0,42 \text{ або } \tau_{мін} = 42\%.$$

Оскільки величина $E_в$ = 45,2% > $\tau_{мін}$ = 42%, то потенційний інвестор у принципі може бути зацікавлений у фінансуванні та комерціалізації розробленої системи (програмного забезпечення) пошуку та аналізу відгуків і коментарів.

Далі розраховуємо термін окупності коштів, вкладених у можливу комерціалізацію розробленої системи (програмного забезпечення) пошуку та аналізу відгуків і коментарів.

Термін окупності $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в} \quad (5.15)$$

Для нашого випадку термін окупності $T_{ок}$ коштів становитиме:

$$T_{ок} = \frac{1}{0,452} = 2,21 \text{ років} < 3 \text{ років},$$

що свідчить про потенційну доцільність комерціалізації розробленої системи (програмного забезпечення) пошуку та аналізу відгуків і коментарів.

Далі проведено моделювання залежності величини внутрішньої дохідності вкладених потенційних інвестицій від рівня інфляції в країні.

Якщо рівень інфляції в країні зросте до 20%, то:

$$ПП = \frac{376}{(1+0,2)^2} + \frac{615}{(1+0,2)^3} + \frac{855}{(1+0,2)^4} \approx 261 + 356 + 412 = 1029 \text{ тисяч грн.}$$

Тоді абсолютний ефект від можливого впровадження даної розробки за три роки складе:

$$E_{абс} = 1029 - 305 = 724 \text{ тисяч грн.}$$

Внутрішня дохідність $E_в$ вкладених інвестицій становитиме:

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

де $E_{абс}$ – абсолютний ефект вкладених інвестицій; $E_{абс}$ = 724 тисяч грн;

PV –теперішня вартість початкових інвестицій PV = 305 тисяч грн.

Для нашого випадку отримаємо:

$$E_в = \sqrt[4]{1 + \frac{724}{305}} - 1 = \sqrt[4]{1 + 2,3738} - 1 = \sqrt[4]{3,3738} - 1 = 1,355 - 1 = 0,355 = 35,5\%.$$

Оскільки величина $E_в$ = 35,5% $\approx \tau_{мін}$ = 42%, то потенційний інвестор у принципі (після проведення додаткових уточнень) може бути зацікавлений у фінансуванні та комерціалізації розробленої системи (програмного забезпечення) пошуку та аналізу відгуків і коментарів.

Якщо рівень інфляції в країні зросте до 30%, то:

$$ПП = \frac{376}{(1+0,3)^2} + \frac{615}{(1+0,3)^3} + \frac{855}{(1+0,3)^4} \approx 223 + 280 + 300 = 803 \text{ тисяч грн.}$$

Тоді абсолютний ефект від можливого впровадження даної розробки за три роки складе:

$$Е_{абс} = 803 – 305 = 498 \text{ тисяч грн.}$$

Внутрішня дохідність $Е_в$ вкладених інвестицій становитиме:

$$Е_в = \sqrt[T_ж]{1 + \frac{Е_{абс}}{PV}} - 1,$$

де $Е_{абс}$ – абсолютний ефект вкладених інвестицій; $Е_{абс}$ = 498 тисяч грн;

PV –теперішня вартість початкових інвестицій PV = 305 тисяч грн.

Для даного випадку отримаємо:

$$Е_в = \sqrt[4]{1 + \frac{498}{305}} - 1 = \sqrt[4]{1 + 1,6328} - 1 = \sqrt[4]{2,6328} - 1 = 1,274 - 1 = 0,274 = 27,4\%.$$

Оскільки величина $Е_в$ = 27,4% < $\tau_{мін}$ = 42%, то потенційний інвестор може бути і незацікавлений у фінансуванні та комерціалізації даної розробки.

Зроблені розрахунки у вигляді графіків наведено на рис. 5.1.

Аналіз діаграм на рис 5.1 показує, що при рівні інфляції в 10% величина внутрішньої дохідності інвестицій становить $Е_в$ = 45,2%, що вище порогового значення $\tau_{мін}$ = 42% і тому комерціалізація даної розробки може бути доцільною. При рівні інфляції в 20% величина внутрішньої дохідності інвестицій, вкладених в комерціалізацію нашої розробки, становить $Е_в$ = 35,5% < 42%, а при рівні інфляції в 30% величина внутрішньої дохідності інвестицій, вкладених в комерціалізацію нашої розробки, становить $Е_в$ = 27,4% < 42%, тобто нижче порогового рівня $\tau_{мін}$ = 42%, що може викликати у потенційного інвестора сумніви у доцільності комерціалізації нашої розробки.

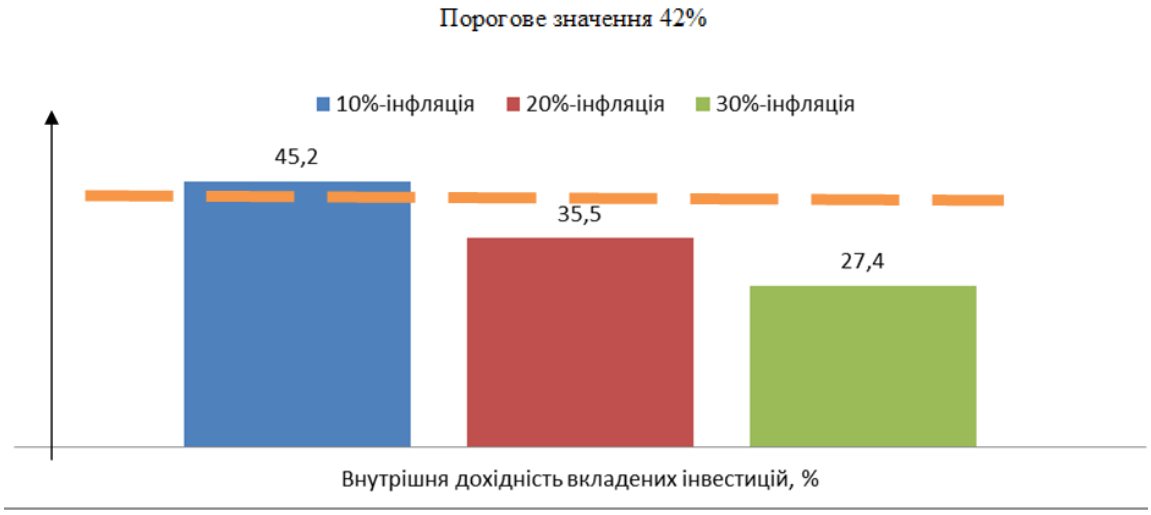

Рисунок 5.1 – Моделювання залежності величини внутрішньої дохідності потенційних інвестицій, вкладених у комерціалізацію нашої розробки, від рівня інфляції в країні (10%, 20% і 30%).

Тому у двох останніх випадках остаточне рішення з питання комерціалізації даної розробки потребує проведення додаткових розрахунків (можливо – зниження рівня ризикованості вкладень, збільшення попиту на розробку, збільшення ціни реалізації розробки тощо).

Результати виконаної економічної частини магістерської кваліфікаційної роботи зведено у таблицю 5.6.

Таблиця 5.6 – Результати виконаної економічної частини

Показники	Задані у ТЗ	Досягнуті у магістерській кваліфікаційній роботі	Висновок
1. Витрати на розробку	Не більше 65 тисяч грн	61 тисяча грн	Досягнуто
2. Абсолютний Ефект від впровадження розробки, тисяч грн	Не менше 1000 тисяч грн	1052 тисяч грн (при 10%-інфляції)	Виконано
3. Внутрішня дохідність інвестицій, %	не менше 42%	45,2% (при 10%-інфляції)	Досягнуто
4. Термін окупності інвестицій, роки	до 3-ти років	2,21 років	Виконано

Таким чином, основні техніко-економічні показники (характеристики) розробленої системи (програмного забезпечення) пошуку та аналізу відгуків і коментарів, визначені у технічному завданні, виконані.

ВИСНОВКИ

В результаті виконання даної роботи було розширено функціональні можливостей сервісів для індексації та обробки даних та розроблена система для пошуку та аналізу коментарів. Розроблена система викладена у відкритому доступі, є простою в установці, модульною та дозволяє доробку та можливість масштабування.

Система використовує наступні технології/компоненти: Java, Spring, Hibernate, Apache Lucene, Apache OpenNLP, PostgreSQL, Maven. Для роботи використовується середовище розробки Intellij IDEA.

У першому розділі роботи проведено аналіз сучасного стану проблеми та сформульовано основні задачі роботи. Проведений аналіз існуючих аналогів. Зроблено висновки про їх переваги та недоліки.

У другому розділі роботи було обґрунтовано вибір технологій, бібліотек та інструменти для розробки системи.

У третьому розділі роботи показний процес проектування архітектури системи. Велика увага приділена архітектурі програмного забезпечення та можливості до розширення функціоналу. Особливу увагу було приділено для того щоб система залишалась масштабованою та відкритою для розширення функціоналу, але залишилась при цьому самодостатньою. Також увага приділялась зручності взаємодії користувача із програмою та простій і зрозумілій реалізації графічного інтерфейсу. В цьому розділі були описані архітектурні шаблони які використовуються в роботі.

У четвертому розділі описано загальні принципи і процес розробки та тестування програми. В цьому розділі були описані загальні принципи, методи, підходи та загальні правила. Були обрані підходи за якими система розроблялась і показаний сам процес розробки і тестування. Тестування допомогло перевірити систему на наявність помилок та готовність працювати у критичних умовах. Додатково було проведено тестування непередбачуваних ситуацій під час роботи з програмою, в результаті якого система показала високі показники

надійності та роботи під час критичних ситуаціях. Також в цьому розділі були описані можливі способи покращення системи в майбутньому за рахунок додаткового функціоналу.

У п’ятому розділі було проведено економічний розрахунок і було проаналізовано комерціалізація розробки потенційним інвестором може бути переспективною при інфляції в 10% і нижче.

У даній роботі була спроектована система пошуку та аналізу коментарів з використанням: Java, Spring, Hibernate, Apache Lucene, Apache OpenNLP, PostgreSQL, Maven, що на відміну від існуючих засобів та інструментів, комбінує у собі засоби для пошуку і аналізу коментарів, та дає можливість без зайвих зусиль проіндексувати потрібні коментарі, та зробити їх аналізю на будь–якій з платформ так як реалізована у вигляді веб–застосунку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Share of households with a computer at home worldwide from 2005 to 2019. Statista[Електронний ресурс]. – Режим доступу: https://www.statista.com/statistics/748551/worldwide-households-with-computer/

2. 53 Important Statistics About How Much Data Is Created Every Day. FinancesOnline [Електронний ресурс]. – Режим доступу: https://financesonline.com/how-much-data-is-created-every-day/

3. Format race 2018: DOCX over DOC and ODT? ONLYOFFICE[Електронний ресурс]. – Режим доступу: https://www.onlyoffice.com/blog/2018/09/format-race-2018-docx-over-doc-and-odt/

4. PDF statistics – the universe of electronic documents. PDF Association[Електронний ресурс]. – Режим доступу: https://www.pdfa.org/wp-content/uploads/2018/06/1330_Johnson.pdf

5. Document managment. OpenKM [Електронний ресурс]. – Режим доступу: https://www.openkm.com/en/document-management.html

6. Designing Your Document Managment Solution. SimpleIndex[Електронний ресурс]. – Режим доступу: https://www.simpleindex.com/features/document-management/

7. Full-text search of documents. LogicalDOC [Електронний ресурс]. – Режим доступу: https://www.logicaldoc.com/software-features/full-text-indexing

8. AI-powered Text Analytics API [Електронний ресурс]. – Режим доступу: https://www.repustate.com/text-analytics-api/

9. Історія компанії: як Rozetka «вмикалася» в лідери [Електронний ресурс]. – Режим доступу: https://new.finance.ua/ua/30-rokiv-nezalezhnosti/rozetka

10. The Amazon Marketplaces Worldwide in Focus [Електронний ресурс]. – Режим доступу: https://www.blankspace.eu/blog-posts-en/amazon-marketplaces-worldwide

11. Participatory Culture, Community, and Play: Learning from Reddit [Електронний ресурс]. – Режим доступу:

https://www.researchgate.net/publication/284186822_Participatory_Culture_Community_and_Play_Learning_from_Reddit

12. Індексування. Вікіпедія[Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D0%B4%D0%B5%D0%BA%D1%81%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F

13. Пошуковий індекс. Вікіпедія[Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D1%88%D1%83%D0%BA%D0%BE%D0%B2%D0%B8%D0%B9_%D1%96%D0%BD%D0%B4%D0%B5%D0%BA%D1%81

14. Інвертований індекс. Вікіпедія [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D0%B2%D0%B5%D1%80%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B9_%D1%96%D0%BD%D0%B4%D0%B5%D0%BA%D1%81

15. Tokenization in NLP: Types, Challenges, Examples, Tools [Електронний ресурс]. – Режим доступу: https://neptune.ai/blog/tokenization-in-nlp

16. [Електронний ресурс]. – Режим доступу: https://medium.com/@alec.mccabe93/lemmatization-and-stemming-5b6b3718b49

17. Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/publication/318739283_Tokenizing_POS_Tagging_Lemmatizing_and_Parsing_UD_20_with_UDPipe

18. TIOBE Index for December 2023 [Електронний ресурс]. – Режим доступу: https://www.tiobe.com/tiobe-index/

19. Аналіз переваг та недоліків розробки програмного забезпечення за допомогою мови Java. Науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації [Електронний ресурс]. – Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2022/paper/view/15429

20. Intellij IDEA. JetBrains [Електронний ресурс]. – Режим доступу: https://www.jetbrains.com/idea/

21. Об'єктно-орієнтоване програмування. Вікіпедія[Електронний ресурс]. https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F – Режим доступу:

22. Мартін Р. Чиста архітектура – Фабула, 2019 – 368 с.

23. Порівняльний аналіз систем управління реляційними базами даних MySQL та PostgreSQL. Науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації: веб-сайт URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2022/paper/view/15430 (дата звернення: 19.03.2023)

24. Lesson: JDBC Basics. Oracle Java Documentation[Електронний ресурс]. – Режим доступу: https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html

25. The system of indexing documents and searching for their indexes stored in database written in Java using Gradle build automation tool. Молодь в науці: дослідження, проблеми, перспективи (МН-2022) [Електронний ресурс]. – Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2022/paper/view/13938

26. Review of build tool Gradle for Java. Молодь в науці: дослідження, проблеми, перспективи (МН-2021)[Електронний ресурс]. – Режим доступу: https://conferences.vntu.edu.ua/index.php/allvntu/mn2021/paper/view/13095

27. Mark Richards, Neal Ford Fundamentals of Software Architecture: An Engineering Approach 1st Edition. O'Reilly Media, 2020. 419 p.

28. Модель-вид-контролер. Вікіпедія [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C-%D0%B2%D0%B8%D0%B4-%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%B5%D1%80

29. Аналіз методів та підходів у використанні технологій штучного інтелекту для обробки текстової інформації на прикладі мовної моделі PaLM. Молодь в Науці [Електронний ресурс]. – Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/19720

30. Introduction to Apache Lucene [Електронний ресурс]. – Режим доступу: https://www.baeldung.com/lucene

31. Intro to Apache OpenNLP [Електронний ресурс]. – Режим доступу: https://www.baeldung.com/apache-open-nlp

32. V Model In Software Development-Best Practice in SDLC Process [Електронний ресурс]. – Режим доступу: https://www.bdtask.com/blog/v-model-in-software-development

33. The Practical Test Pyramid [Електронний ресурс]. – Режим доступу: https://martinfowler.com/articles/practical-test-pyramid.html

34. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт. / Укладачі В.О. Козловський, О.Й. Лесько, В.В.Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

ДОДАТКИ

102

Додаток А (обов'язковий)

Технічне завдання

ЗАТВЕРДЖУЮ

Завідувач кафедри АІІТ

д.т.н., проф. Олег БІСІКАЛО

«12» жовтня 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на магістерську кваліфікаційну роботу

« Система пошуку та аналізу відгуків і коментарів »

08-31.МКР.002.02.000 ТЗ

Керівник роботи

к.т.н.,доц.каф.АІІТ

Ілона БОГАЧ

«12» жовтня 2023 р.

Виконавець:

ст. гр. ІСТ-22м

Вільям ВОЙЦЕХОВСЬКИЙ

«12» жовтня 2023 р.

Вінниця ВНТУ 2023

103

1. Назва та галузь застосування

Магістерська кваліфікаційна робота: « Система пошуку та аналізу відгуків коментарів ». Галузь застосування – інформаційні технології.

2. Підстава для розробки

Розробку системи здійснювати на підставі наказу по університету№ від 2023 та завдання до магістерської кваліфікаційної роботи, складеного та затвердженого кафедрою «Автоматизації та інтелектуальних інформаційних технологій»

3. Мета та призначення розробки

Метою роботи є реалізація системи для пошуку та аналізу коментарів.

4. Джерела розробки

1. Tokenization in NLP: Types, Challenges, Examples, Tools [Електронний ресурс]. – Режим доступу: https://neptune.ai/blog/tokenization-in-nlp

2. Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/publication/318739283_Tokenizing_POS_Tagging_Lemmatizing_and_Parsing_UD_20_with_UDPipe

3. Intro to Apache OpenNLP [Електронний ресурс]. – Режим доступу: https://www.baeldung.com/apache-open-nlp

4. Mark Richards, Neal Ford Fundamentals of Software Architecture: An Engineering Approach 1st Edition. O'Reilly Media, 2020. 419 p.

5. Показники призначення

Основні технічні вимоги та мінімальні системні вимоги до програми:

- ОС: Manjaro Linux;
- Доступ до інтернету;
- Оперативна пам'ять: 2 ГБайт і вище
- Процесор: Intel Core i3 і вище

Результати роботи програми:

- Читання та зберігання коментарів
- Виведення списку коментарів;
- Пошук по коментарях;
- Можливість аналізу коментарів.

6. Економічні показники

- витрати на розробку – не більше 65 тис. грн;
- абсолютний ефект від впровадження розробки – не менше 1000 тис. грн;
- внутрішня дохідність інвестицій – не менше 45,2%;
- термін окупності – не більше 2,21 років.

7. Стадії розробки

1. Розділ 1 « Дослідження існуючих технологій індексації та аналізу інформації та коментарів » має бути виконаний до 30.09.2023.
2. Розділ 2 « Обґрунтування вибору технологій та методів розробки » має бути виконаний до 05.10.2023.
3. Розділ 3 « Проєктування архітектури системи » має бути виконаний до 15.10.2023.
4. Розділ 3 « Розробка та тестування системи » має бути виконаний до 12.11.2023.
5. Економічна частина має бути виконана до 18.11.2023.

8. Порядок контролю та приймання

1. Рубіжний контроль провести до 07.12.2023.
2. Попередній захист магістерської кваліфікаційної роботи провести до 21.11.2023.
3. Захист магістерської кваліфікаційної роботи провести в період з 12.12.2023 до 14.12.2023.

Додаток Б (Обов'зяковий)

Ілюстративна частина

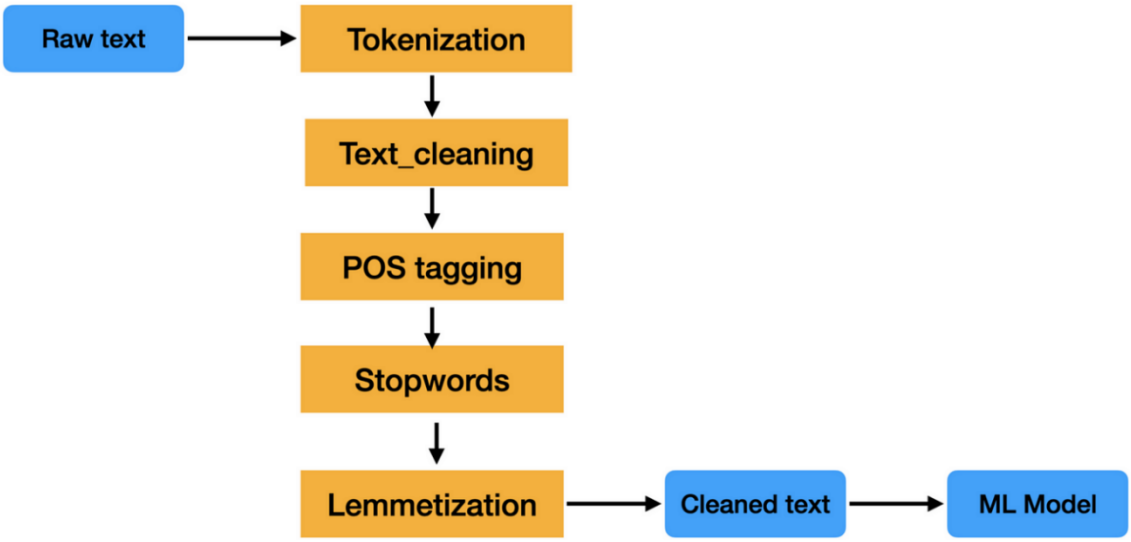

Рисунок Б.1 - Схема роботи Apache OpenNLP

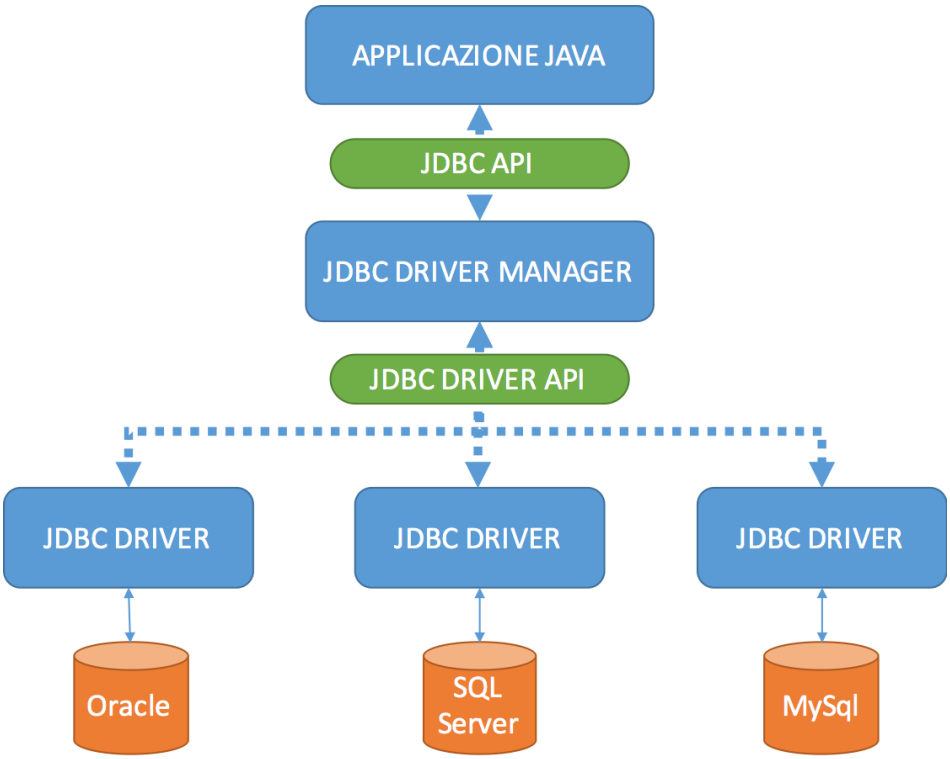

Рисунок Б.2 - Схема роботи JDBC

Продовження додатку Б:

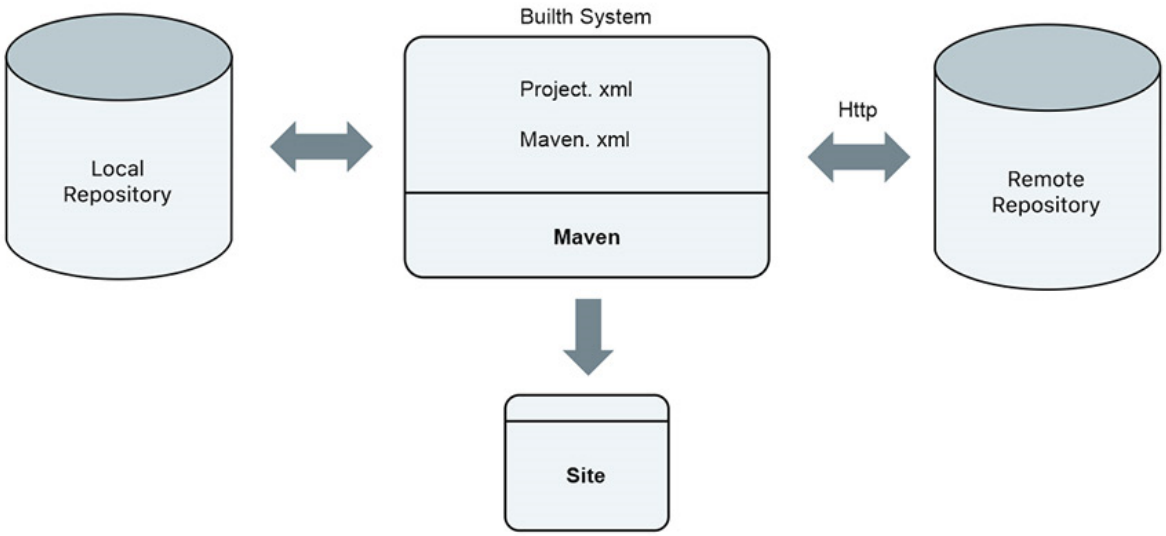

Рисунок Б.3 - Схема роботи Maven

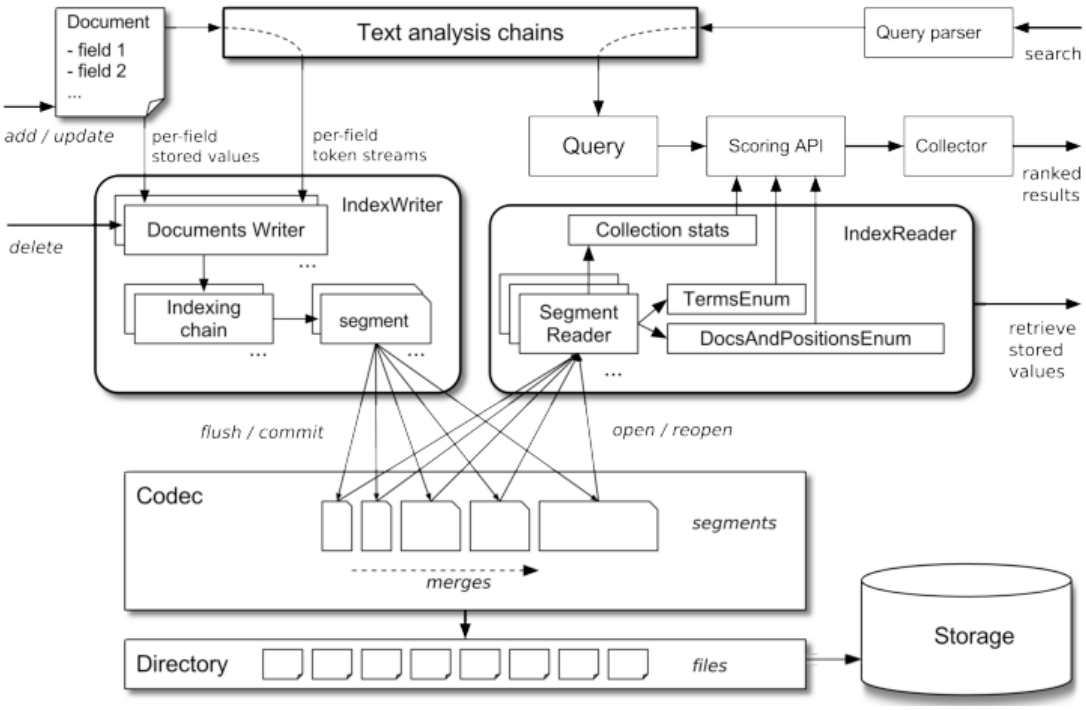

Рисунок Б.4 - Схема роботи Apache Lucene

Продовження додатку Б:

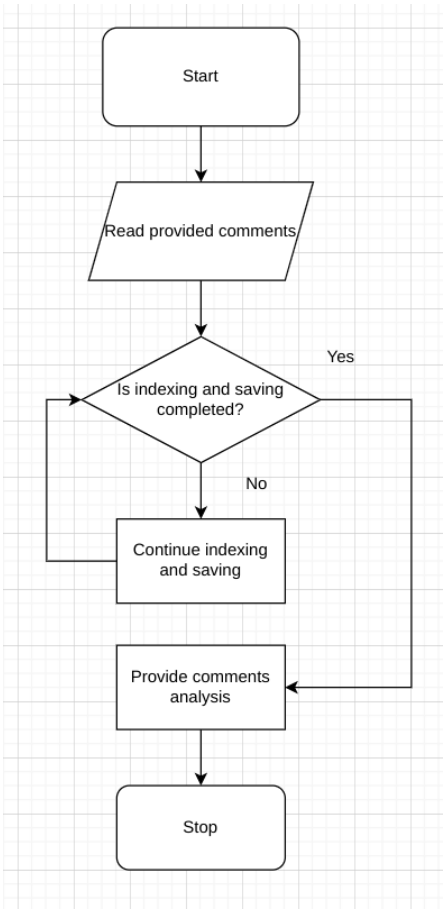

Рисунок Б.5 - Схема алгоритму роботи системи

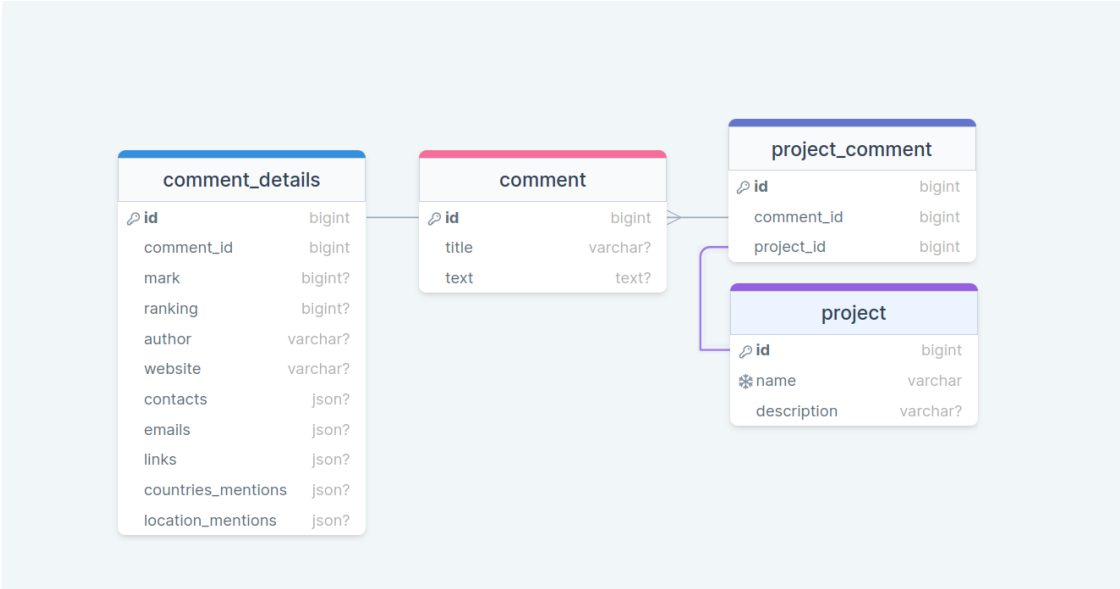

Рисунок Б.6 - Схема бази даних

Додаток В(обов'язковий)

Лістинг системи

```
package com.example.servingwebcontent;

import com.fasterxml.jackson.databind.ObjectMapper;
import lombok.SneakyThrows;
import opennlp.tools.lemmatizer.DictionaryLemmatizer;
import opennlp.tools.namefind.NameFinderME;
import opennlp.tools.namefind.TokenNameFinderModel;
import opennlp.tools.postag.POSModel;
import opennlp.tools.postag.POSTaggerME;
import opennlp.tools.tokenize.TokenizerME;
import opennlp.tools.tokenize.TokenizerModel;
import opennlp.tools.util.Span;
import org.springframework.stereotype.Service;

import java.io.InputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.*;
import java.util.function.Function;
import java.util.stream.Collectors;

@Service
public class CommentService {

    CommentRepository commentRepository;

    @SneakyThrows
    public CommentResponse readCommentsFromJson(String json) {
        ObjectMapper mapper = new ObjectMapper();
        CommentsContainer commentsContainer = mapper.readValue(json,
CommentsContainer.class);

        InputStream modelInputStream = getClass()
                .getResourceAsStream("/models/en-token.bin");
        TokenizerModel model = new TokenizerModel(modelInputStream);
        TokenizerME tokenizer = new TokenizerME(model);
```

```
        //
        InputStream inputStreamPOSTagger = getClass()
                .getResourceAsStream("/models/en-pos-maxent.bin");
        POSModel posModel = new POSModel(inputStreamPOSTagger);
        POSTaggerME posTagger = new POSTaggerME(posModel);
        //

        List<String> totalLemas = new ArrayList<>();
        Map<String, Long> totalWordFrequency = new HashMap<>();
        Map<String, Long> mostPopularWords = new HashMap<>();

        commentsContainer.getComments().forEach(comment ->{


comment.setLemmas(lemmatization(tokenizer.tokenize(comment.getTitle() + " " +
comment.getText())));
            System.out.println(comment.getLemmas());
            totalLemas.addAll(comment.getLemmas());

//comment.setWordFrequency(comment.getLemmas().stream().collect(Collectors.group
ingBy(Function.identity(), Collectors.counting())));
        });

        List<String> tags = List.of("NN", "NNS", "NNP", "JJ", "JJR", "JJS",
"FW", "CD", "RB", "RBR", "RBS", "UH", "VB", "VBD", "VBG", "VBN", "VBP", "VBZ" );

        totalWordFrequency =
totalLemas.stream().collect(Collectors.groupingBy(Function.identity(),
Collectors.counting()));
        mostPopularWords = totalWordFrequency.entrySet().stream()
                .filter(w -> !w.getKey().equals("O"))
                .filter(w -> !w.getKey().equals("be"))
                .filter(w -> !w.getKey().equals("have"))
                .filter(w -> w.getValue() > 1)
                .filter(w -> tags.contains(posTagger.tag(new
String[]{w.getKey()})[0]))
                .sorted(Map.Entry.<String, Long>comparingByValue().reversed())
                .collect(Collectors.toMap(Map.Entry::getKey,
Map.Entry::getValue, (oldValue, newValue) -> oldValue, LinkedHashMap::new));
        System.out.println(mostPopularWords);
```

```
        return CommentResponse.builder()
                .commentList(commentsContainer.getComments())
                .totalWordFrequency(mostPopularWords)
                .build();
    }

    @SneakyThrows
    public List<Span> personIdentifier(String[] tokens) {
        InputStream inputStreamNameFinder = CommentService.class
                .getResourceAsStream("/models/en-ner-person.bin");
        TokenNameFinderModel modelName = new TokenNameFinderModel(
                inputStreamNameFinder);
        NameFinderME nameFinderME = new NameFinderME(modelName);
        List<Span> spans = Arrays.asList(nameFinderME.find(tokens));
        return spans;
    }

    @SneakyThrows
    public List<String> lemmatization(String[] tokens) {
        InputStream inputStreamPOSTagger = CommentService.class
                .getResourceAsStream("/models/en-pos-maxent.bin");
        POSModel posModel = new POSModel(inputStreamPOSTagger);
        POSTaggerME posTagger = new POSTaggerME(posModel);
        String tags[] = posTagger.tag(tokens);
        InputStream dictLemmatizer = CommentService.class
                .getResourceAsStream("/models/en-lemmatizer.dict");
        DictionaryLemmatizer lemmatizer = new DictionaryLemmatizer(
                dictLemmatizer);
        String[] lemmas = lemmatizer.lemmatize(tokens, tags);
        System.out.println(Arrays.toString(lemmas));
        Map<String, Long> wordFreq =
Arrays.stream(lemmas).collect(Collectors.groupingBy(String::toString,
Collectors.counting()));
        List<String> smth = wordFreq.entrySet().stream()
                .filter(v -> v.getValue() > 1).map(Map.Entry::getKey).toList();

        System.out.println(curlToGoogleModel("h"));
        return Arrays.stream(lemmas).toList();
    }

    @SneakyThrows
```

```
    public String curlToGoogleModel(String prompt) {
        URL url = new
URL("https://generativelanguage.googleapis.com/v1beta3/models/text-bison-
001:generateText?key=" + API_KEY);

        HttpURLConnection httpConnection = (HttpURLConnection)
url.openConnection();
        httpConnection.setRequestMethod("POST");
        httpConnection.setRequestProperty("Content-Type", "application/json");
        httpConnection.setDoOutput(true);

        OutputStreamWriter writer = new
OutputStreamWriter(httpConnection.getOutputStream());
        writer.write("{ \"prompt\": { \"text\": \"" + prompt + "\" } }");
        writer.flush();
        writer.close();
        httpConnection.getOutputStream().close();

        InputStream responseStream = httpConnection.getResponseCode() / 100 == 2
                ? httpConnection.getInputStream()
                : httpConnection.getErrorStream();
        Scanner scanner = new Scanner(responseStream).useDelimiter("\\A");
        return scanner.hasNext() ? scanner.next() : "";
    }

    public CommentResponse getComment(Long id) {
        return CommentResponse.builder().build();
    }
}
```

112

Додаток Г (обов'язковий)

Протокол перевірки на плагіат

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Система пошуку та аналізу відгуків і коментарів

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ: кафедра Автоматизації та інтелектуальних інформаційних технологій, факультет інтелектуальних інформаційних технологій та автоматизації
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 99.3% Схожість 0.7%

Аналіз звіту подібності (відмітити потрібне):

☑ 1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

☐ 2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

☐ 3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Роман МАСЛІЙ
(підпис) (ім'я, ПРІЗВИЩЕ)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Вільям ВОЙЦЕХОВСЬКИЙ
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник роботи _____ Ілона БОГАЧ
(підпис) (ім'я, ПРІЗВИЩЕ)