Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій

Бакалаврська дипломна робота на тему:

«ІНТЕЛЕКТУАЛЬНА ВЕБ-СИСТЕМА АНАЛІЗУ ВІДГУКІВ НА ПРОГРАМНІ ЗАСТОСУНКИ»

Виконав: студент 4 курсу, групи 2ІСТ-20б спеціальності 126 – «Інформаційні системи та технології»
_____ Владислав ПОБІДАШ
Керівник: к.т.н., доц. каф. САІТ
_____ Євгеній КРИЖАНОВСЬКИЙ
«31» _____05_____ 2024 р.
Рецензент: к.т.н., доц. каф. КН
_____ Володимир ОЗЕРАНСЬКИЙ
«10» _____06_____ 2024 р.

Допущено до захисту

Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
«01» _____06_____ 2024 р.

Вінниця ВНТУ – 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій
Рівень вищої освіти – перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність 126 – «Інформаційні системи та технології»
Освітньо-професійна програма – Прикладні інформаційні технології

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
к.т.н., проф. Віталій МОКІН
"05" 05 2024 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ
Побідашу Владиславу Віталійовичу

1. Тема роботи: «Інтелектуальна веб-система аналізу відгуків на програмні застосунки»
керівник роботи: Крижановський Є. М., к.т.н., доц. каф. САІТ
затверджені наказом ВНТУ від «11» 05 2024 року №80
2. Термін подання студентом роботи 31.05
3. Вихідні дані до роботи:
1) Відгуки користувачів на застосунки в Google Play та App Store.
4. Зміст текстової частини:
1) Аналіз проблематики відгуків на програмні застосунки.
2) Вибір оптимальних інформаційних технологій.
3) Розроблення інтелектуальної веб-системи аналізу відгуків на програмні застосунки.
5. Перелік ілюстративного матеріалу:
1) Контекстні діаграми;
2) Архітектура інтелектуальної веб-системи;
3) Діаграма USE CASE інтелектуальної веб-системи.

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 11.03

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області	11.03	31.03	вик
2	Порівняльний аналіз проблематики	01.04	15.04	вик
3	Вибір оптимальних інформаційних технологій	16.04	30.04	вик
4	Розроблення інтелектуальної веб-системи аналізу відгуків на програмні застосунки	01.05	15.05	вик
5	Оформлення матеріалів до захисту БДР	16.05	31.05	вик

Студент _____ Владислав ПОБІДАШ

Керівник роботи _____ Євгеній КРИЖАНОВСЬКИЙ

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 80 сторінок формату А4, на яких є 41 рисунок, в списку використаних джерел міститься 21 найменування.

Мета даної роботи – підвищення ефективності аналізу відгуків на програмні застосунки шляхом розробки інтелектуальної веб-системи.

В роботі розглянуто методи аналізу та класифікації текстових даних, використовуючи великі мовні моделі (LLM) та навчання з перенесенням (transfer learning). Розроблено інтелектуальну веб-систему для збору та аналізу відгуків на програмні застосунки. Система представляє собою веб-додаток, який розроблено на базі платформи Node.js, застосовуючи Next.js та Nest.js, які є фреймворками мови програмування TypeScript.

Інтелектуальна веб-система здійснює ефективний аналіз текстових відгуків на програмні застосунки за допомогою GPT-4, використовуючи його можливості обробки природної мови для розуміння семантики і контексту відгуків. Вона виконує тематичний аналіз для класифікації відгуків за темами, сентимент-аналіз для визначення емоційного настрою, а також генерує короткі підсумки відгуків. Система також може генерувати відповіді на відгуки і виявляє помилки та пропозиції з поліпшення, які зазначаються у відгуках, що покращує взаємодію з користувачами і підвищує ефективність програми.

Ключові слова: інтелектуальна веб-система, веб-додаток, аналіз даних, обробка природних мов, класифікація даних, великі мовні моделі, навчання з перенесенням, інженерія підказок.

ABSTRACT

The bachelor's thesis consists of 80 pages of A4 format, with 41 figures, and 21 references.

The aim of this paper is to improve the efficiency of analyzing feedback on software applications by developing an intelligent web-based system.

The paper considers methods for analyzing and classifying text data using large language models (LLM) and transfer learning. An intelligent web-based system for collecting and analyzing feedback on software applications has been developed. The system is a web application developed on the Node.js platform using Next.js and Nest.js, which are TypeScript programming language frameworks.

The intelligent web-based system efficiently analyzes textual reviews of software applications using GPT-4, utilizing its natural language processing capabilities to understand the semantics and context of reviews. It performs topic analysis to categorize reviews by topic, sentiment analysis to determine emotional sentiment, and generates summaries of reviews. The system can also generate responses to reviews and detects errors and suggestions for improvement that are mentioned in reviews, which improves user experience and increases the effectiveness of the program.

Keywords: intelligent web-system, web application, data analysis, natural language processing, data classification, large language models, transfer learning, prompt engineering.

ЗМІСТ

ВСТУП

Актуальність теми. Споживачі сучасного світу в більшості випадків орієнтуються на відгуки і огляди при виборі програмного забезпечення. Усвідомлення цієї тенденції та здатність ефективно вивчати й використовувати таку інформацію може суттєво вплинути на успіх у сфері розробки і просування програмних продуктів.

Інформаційна система, яка здатна автоматизувати процес збору, оцінювання та візуального представлення відгуків, могла б стати корисним інструментом для визначення потреб користувачів. Вона також допомогла б розкрити слабкі місця програмного забезпечення та вдосконалити продукцію на основі відгуків від користувачів.

Мета і задачі дослідження. Метою є підвищення ефективності аналізу відгуків на програмні застосунки шляхом використання великих мовних моделей.

Щоб досягнути поставленої мети потрібно:

- проаналізувати предметну область;
- дослідити й обрати методи збору та аналізу відгуків користувачів;
- розробити відповідну інтелектуальну веб-систему.

Об’єктом дослідження є процес створення інтелектуальної веб-системи для збору, аналізу та візуалізації текстових відгуків користувачів на ПЗ.

Предметом дослідження є методи і програмні засоби створення інтелектуальної веб-системи для збору, аналізу та візуалізації текстових відгуків користувачів на програмні застосунки.

Публікації. За результатами виконання даної роботи опубліковано тези доповідей на тему: «Інформаційна система аналізу відгуків на програмні застосунки» на Міжнародній науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (Вінниця, 2023-2024 рр.) [1].

1 АНАЛІЗ ПРОБЛЕМАТИКИ ВІДГУКІВ НА ПРОГРАМНІ ЗАСТОСУНКИ

1.1 Аналіз проблематики відгуків на програмні застосунки

ПЗ (програмне забезпечення) відіграє величезну роль у сучасному житті людей, оскільки воно широко використовується в різноманітних сферах, від електронної комерції і соціальних медіа до освіти та медицини. Це дозволяє людям виконувати задачі швидше та ефективніше, обмінюватися інформацією, часто підвищуючи якість їх життя та продуктивність роботи.

З кінця 70-х років вважається, що залучення користувачів до розробки системи забезпечує її успіх [2-4]. Користувачі зазвичай володіють значними знаннями про сферу застосування, завдання, які вони виконують, робочі практики, контекст використання системи, а також про їхню поведінку та вподобання. Ця форма знань часто є неявною за своєю природою, і тому її важко сформулювати за допомогою типових методів збору інформації. Залучення користувачів до життєвого циклу розробки систем (SDLC) полегшує розуміння їхнього робочого середовища і може покращити якість, точність і повноту їхніх вимог [2, 5].

Магазини додатків, такі як Google Play та Apple AppStore, мають понад 3 мільйони додатків, що охоплюють майже всі види програмного забезпечення та послуг. Мільярди користувачів регулярно завантажують, використовують і рецензують ці додатки. Нещодавні дослідження показали, що огляди, написані користувачами, є багатим джерелом інформації для постачальників додатків і розробників, оскільки вони містять інформацію про помилки, ідеї для нових функцій або документацію до вже випущених функцій [6].

Процес збору та аналізу відгуків користувачів на програмне забезпечення може бути автоматизований за допомогою специфічних інструментів чи API (Application Programming Interface), які надають доступ до цих даних великих обсягів.

Результативні дані часто є неструктурованими та містять шум і нерелевантну інформацію. Вони можуть включати пусті поля, дублікати, невідповідність форматам та інше. Попередня обробка включає очищення даних, нормалізацію тексту (наприклад, переведення тексту в нижній регістр, видалення знаків пунктуації), видалення стоп-слів (наприклад, "та", "або") та стемінг (приведення слів до їх кореневої форми) [6].

Після того, як дані були підготовлені, вони можуть бути аналізовані. Основні методи аналізу відгуків включають аналіз настрою (визначення позитивних, негативних чи нейтральних емоцій в тексті), тематичне моделювання (виявлення ключових тем в тексті), а також виявлення взаємозв'язків та закономірностей [7].

Нарешті, отримані результати аналізу відгуків можуть бути використані для покращення програмного забезпечення, визначення проблем, які виникають у користувачів, і прийняття рішень про подальші кроки розвитку продукту.

1.2 Аналіз ринку програмних застосунків

Ринок програмного забезпечення охоплює розробку, розповсюдження та продаж програмних продуктів і послуг, включаючи операційні системи, програмне забезпечення для підвищення продуктивності, бізнес-додатки, інструменти для дизайну та мультимедіа, а також спеціалізоване програмне забезпечення для конкретних галузей.

Основними тенденціями останніх років стали перехід до хмарних обчислень і моделей програмного забезпечення як послуги (SaaS), що забезпечують масштабованість, гнучкість та економічну ефективність. Зростаюча увага до розробки програмного забезпечення для мобільних пристроїв пов'язана з широким використанням смартфонів і планшетів, що спричинило збільшення попиту на мобільні додатки.

Важливий вплив на ринок програмного забезпечення мають технології машинного навчання та штучного інтелекту, які автоматизують процеси, роблять прогнози та покращують взаємодію з користувачем. Зростаюча кількість

кібератак та витоків даних підвищує потребу в надійному програмному забезпеченні для забезпечення кібербезпеки. Постійні зміни в індустрії програмного забезпечення впливають на її функціонування, комунікацію та існування [8].

У 2022 році обсяг світового ринку програмного забезпечення оцінювався в 589,6 мільярда доларів США, а до 2032 року очікується, що він досягне близько 1 789,14 мільярда доларів США і буде зростати на 11,74% у середньорічному обчисленні протягом прогнозованого періоду з 2023 по 2032 рік (рис. 1.1) [8].

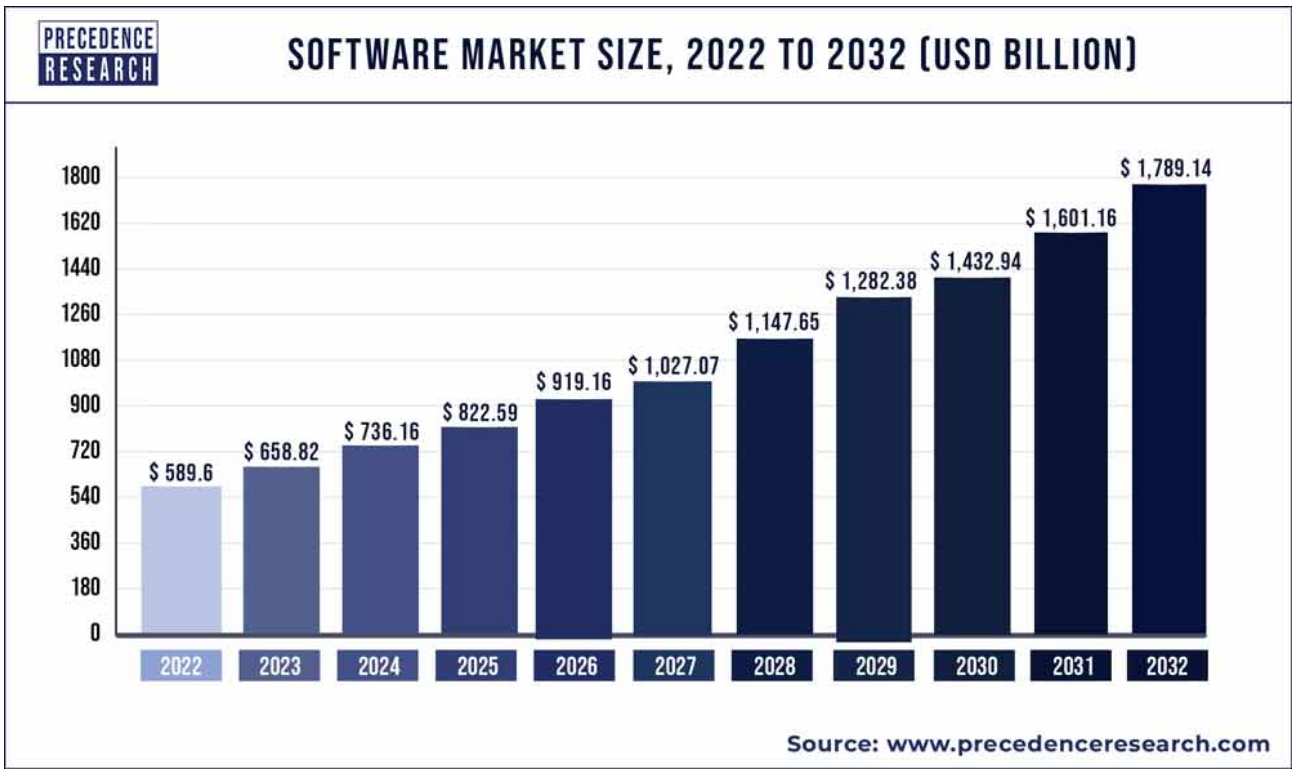


Рисунок 1.1 – Прогнозований обсягу ринку ПЗ [8]

Обсяг ринку програмного забезпечення для бізнесу оцінюється в 0,65 трильйона доларів США у 2024 році і, як очікується, досягне 1,10 трильйона доларів США до 2029 року, зростаючи на 11,23% протягом прогнозованого періоду (2024-2029 рр.) [9].

Популярність смартфонів, планшетів та доступу до інтернету значно збільшила попит на мобільні та онлайн-додатки. Мобільні та веб-додатки стали незамінними для корпоративної та споживчої комунікації, розваг, електронної комерції, підвищення продуктивності тощо. Ця тенденція сприяє зростанню

ринку програмного забезпечення завдяки розробці та впровадженню інноваційних і оптимізованих мобільних та веб-додатків. Через збільшення кількості та складності кібератак і витоків даних, організації приділяють більше уваги безпеці даних та конфіденційності. Зростає попит на потужні рішення для захисту даних та кібербезпеки. Індустрія програмного забезпечення задовольняє цю потребу, пропонуючи передові рішення для забезпечення відповідності, технології шифрування та безпекові додатки.

Аналіз ринку бізнес-програмного забезпечення вказує на кілька ключових чинників, що сприяють його зростанню: підвищення цифровізації бізнесу, прогрес у хмарних технологіях, інтеграція мультиканальних точок взаємодії на єдину платформу, ефективне управління ресурсами компанії та потреба в аналізі великих обсягів даних для отримання інсайтів, що підвищують доходи компаній. Бізнес-програмне забезпечення автоматизує корпоративні процеси, допомагаючи компаніям у плануванні, маркетингу, продажу та управлінні. Воно знижує витрати на інвентаризацію та сировину, підвищуючи прибутковість. Водночас високі витрати на авторизацію та підтримку можуть стримувати ріст ринку. Пандемія COVID-19 прискорила цифровізацію та хмарну міграцію, додатково сприяючи розвитку ринку [9].

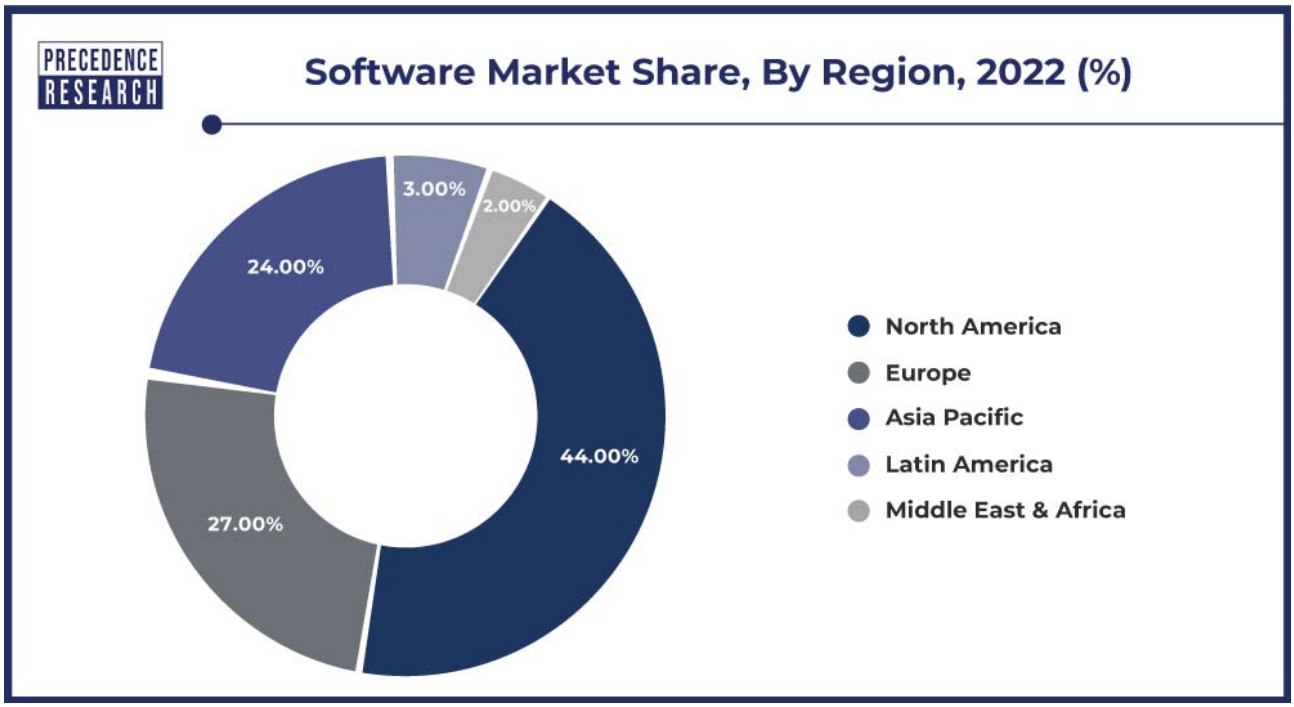


Рисунок 1.2 – Регіональні обсяги ринку ПЗ [8]

У 2022 році Сполучені Штати були лідером на світовому ринку програмного забезпечення (рис. 1.2). Ринок у цьому регіоні добре розвинений, а використання програмного забезпечення є значним у різних галузях. Це пов'язано з наявністю великих компаній-розробників програмного забезпечення, здоровою екосистемою стартапів та значними інвестиціями в інновації у сфері програмного забезпечення. Основними галузями, що стимулюють попит на програмні рішення, є технології, банківська справа, охорона здоров'я та електронна комерція.

Ринок програмного забезпечення в Азійсько-Тихоокеанському регіоні швидко зростає. Основними пріоритетами є використання технологій, цифровізація та зростаюча проникність інтернету. Високий попит на програмні рішення спостерігається в таких галузях, як електронна комерція, мобільні додатки, банківська справа та логістика. Крім того, розвинені економіки Південно-Східної Азії, такі як В'єтнам та Індонезія, мають значні можливості для розвитку індустрії програмного забезпечення.

Ринок програмного забезпечення в Латинській Америці також розвивається, і рівень використання та прийняття програмних рішень зростає. Регіон пропонує можливості для компаній, що надають програмні рішення для галузей роздрібної торгівлі, банківської справи, охорони здоров'я та сільського господарства, а також має зростаючу стартап-спільноту.

На основі аналізу ринку програмного забезпечення можна зробити висновок, що врахування відгуків користувачів є критично важливим для успішного розвитку та впровадження програмних рішень. Крім того, наявність інструментів для аналізу відгуків користувачів допомагає систематизувати та інтерпретувати великий обсяг зворотного зв'язку, виявляючи ключові тенденції та проблеми. Це сприяє прийняттю обґрунтованих рішень щодо розвитку продукту, маркетингових стратегій та підтримки клієнтів.

В умовах швидко змінюваного ринку програмного забезпечення, де конкуренція є дуже високою, здатність оперативно реагувати на потреби та побажання користувачів стає важливим фактором для збереження конкурентної переваги та досягнення успіху.

1.3 Огляд існуючих інформаційних систем аналізу відгуків

Аналіз аналогічних систем є ключовим етапом у розробці інформаційної системи, оскільки він дозволяє виявити оптимальні практики та функціональні можливості, уникнути повторного винаходження велосипеда, ідентифікувати слабкі сторони та помилки інших систем для їх уникнення, а також врахувати користувацькі потреби та успішно впроваджені рішення на ринку.

Appbot – це інструмент для аналізу відгуків додатків, що спеціалізується на відстеженні і аналізі відгуків користувачів в різних магазинах додатків, таких як Google Play, App Store, Amazon Appstore та інших. На рисунку 1.3 наведено ключові можливості Appbot.

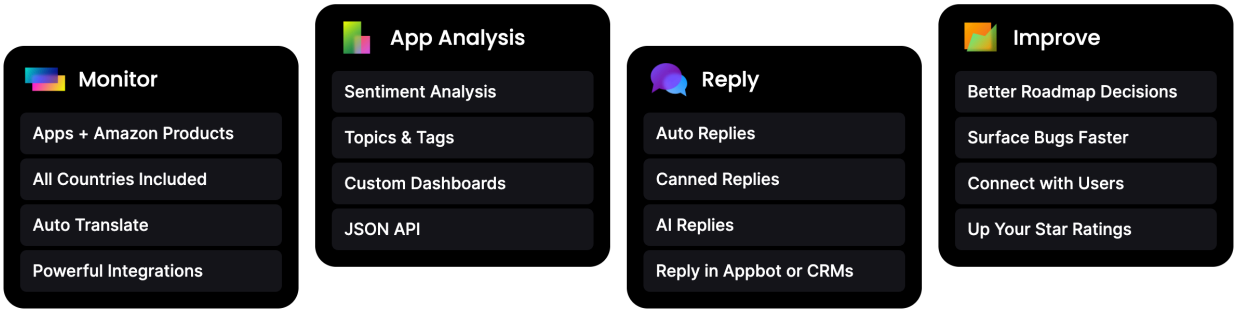


Рисунок 1.3 – Функціональні можливості Appbot

Appbot автоматично відстежує відгуки додатка в різних магазинах і країнах (рис. 1.4). Також можна налаштувати сповіщення по електронній пошті або Slack для обраних подій, таких як отримання негативного відгука.

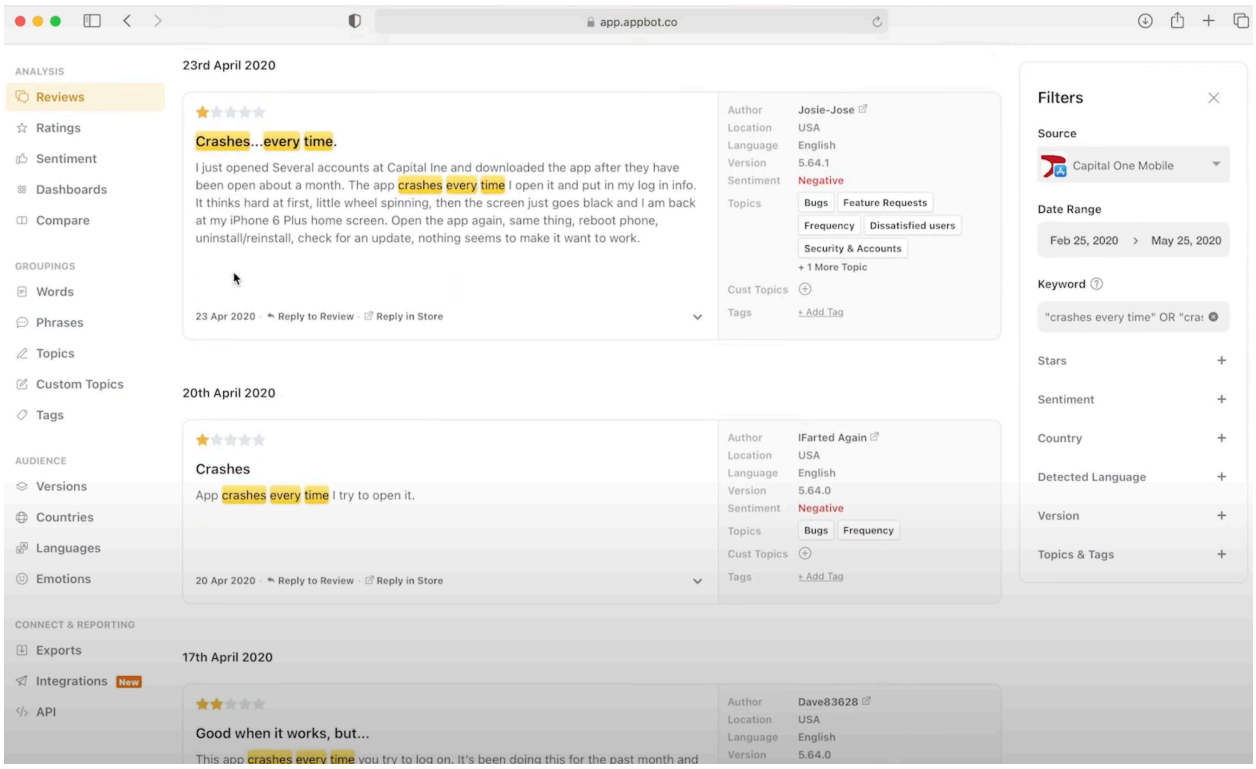


Рисунок 1.4 – Огляд зібраних відгуків в Appbot

Appbot використовує машинне навчання та обробку природної мови для аналізу відгуків. Це дозволяє зрозуміти загальні теми, настрої та тенденції в відгуках користувачів (рис. 1.5).

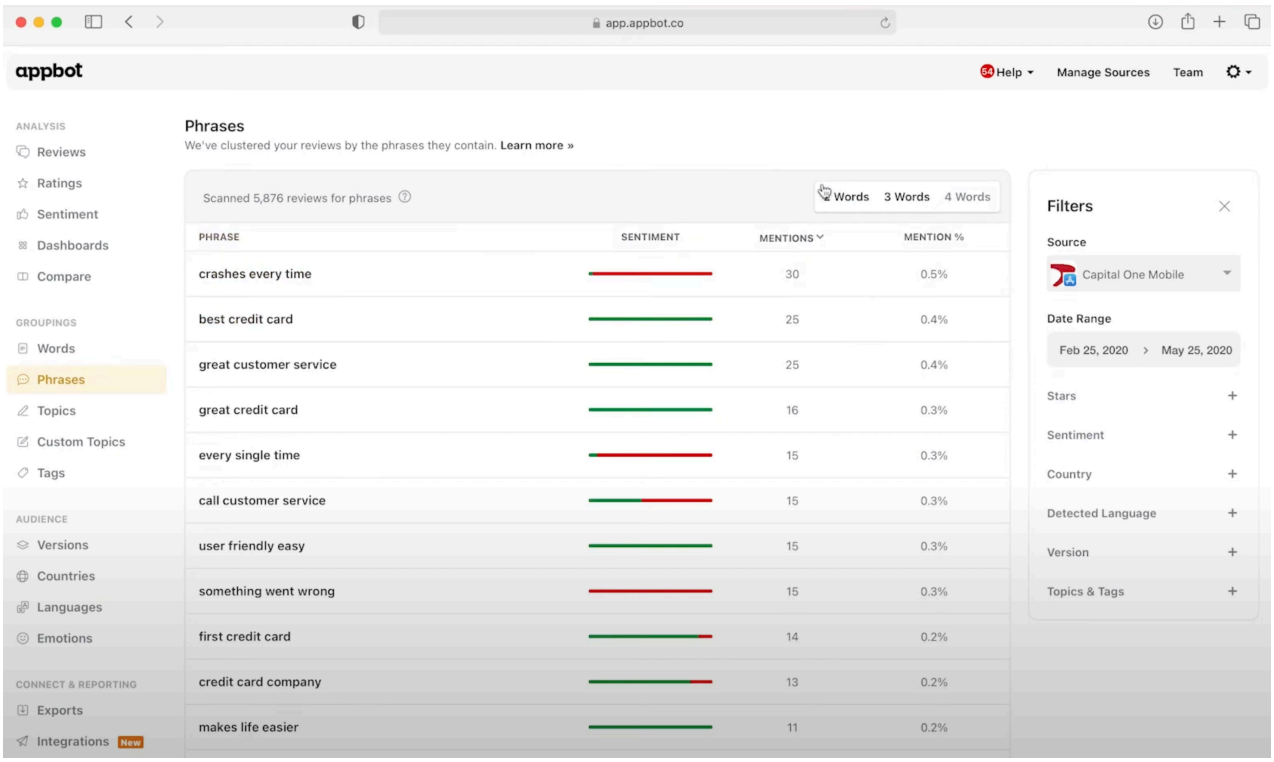


Рисунок 1.5 – Результат використання машинного навчання в Appbot

Appbot надає графіки та діаграми, щоб відобразити аналіз відгуків на основі рейтингу, країни, версії додатка та інших параметрів (рис. 1.6).

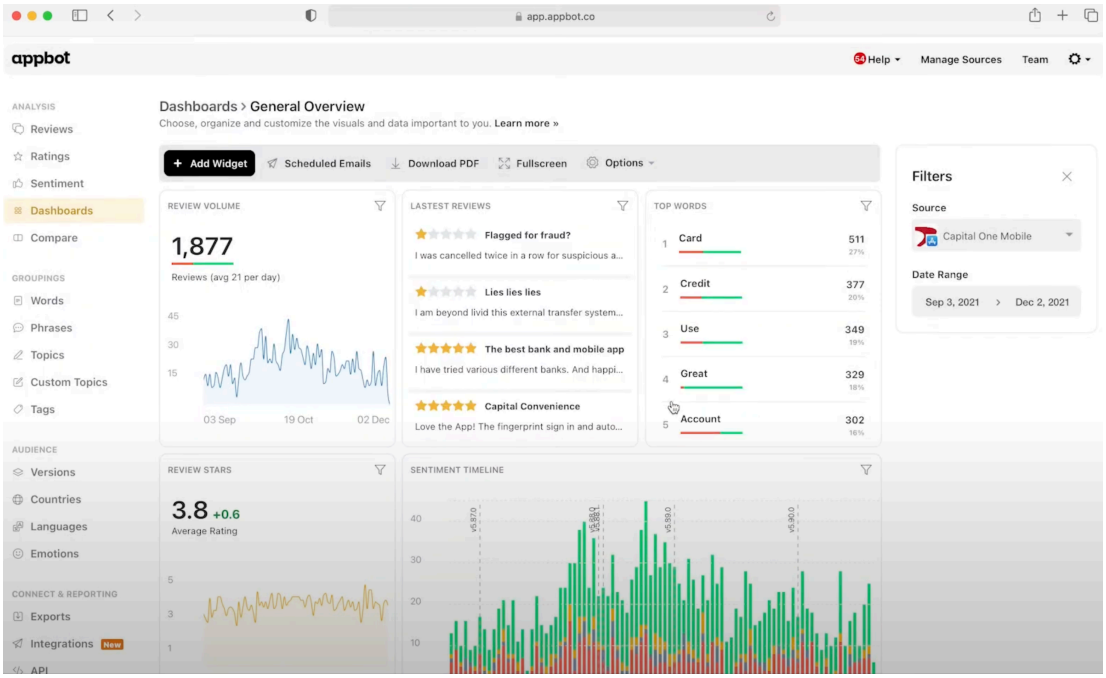


Рисунок 1.6 – Графіки для відображення аналізу відгуків в Appbot

Appbot може інтегруватися з різними інструментами, такими як Slack, Microsoft Teams, Tableau, Google Data Studio, що дозволяє легко відстежувати відгуки і швидко реагувати на них (рис. 1.7).

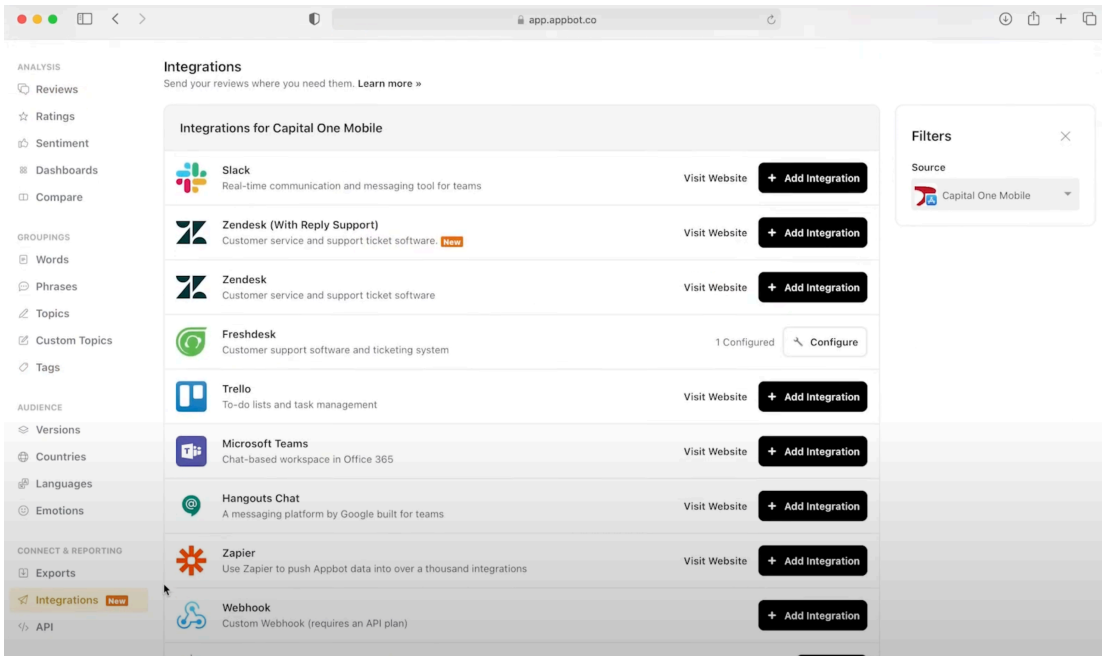


Рисунок 1.7 – Інтеграції в Appbot

Загалом, Appbot – це потужний інструмент для команд розробників програмного забезпечення, менеджерів продуктів та спеціалістів з обслуговування клієнтів, що допомагає видобувати цінні відомості з відгуків користувачів і використовувати ці дані для поліпшення додатка.

UserVoice позиціонує себе як «найкращий у світі інструмент зворотного зв'язку з продуктами» [10]. Йому довіряють провідні продуктові команди як-от Zendesk, PandaDoc, Adobe, Electronic Arts. Доступний функціонал залежить від обраного тарифного плану (рис. 1.8).

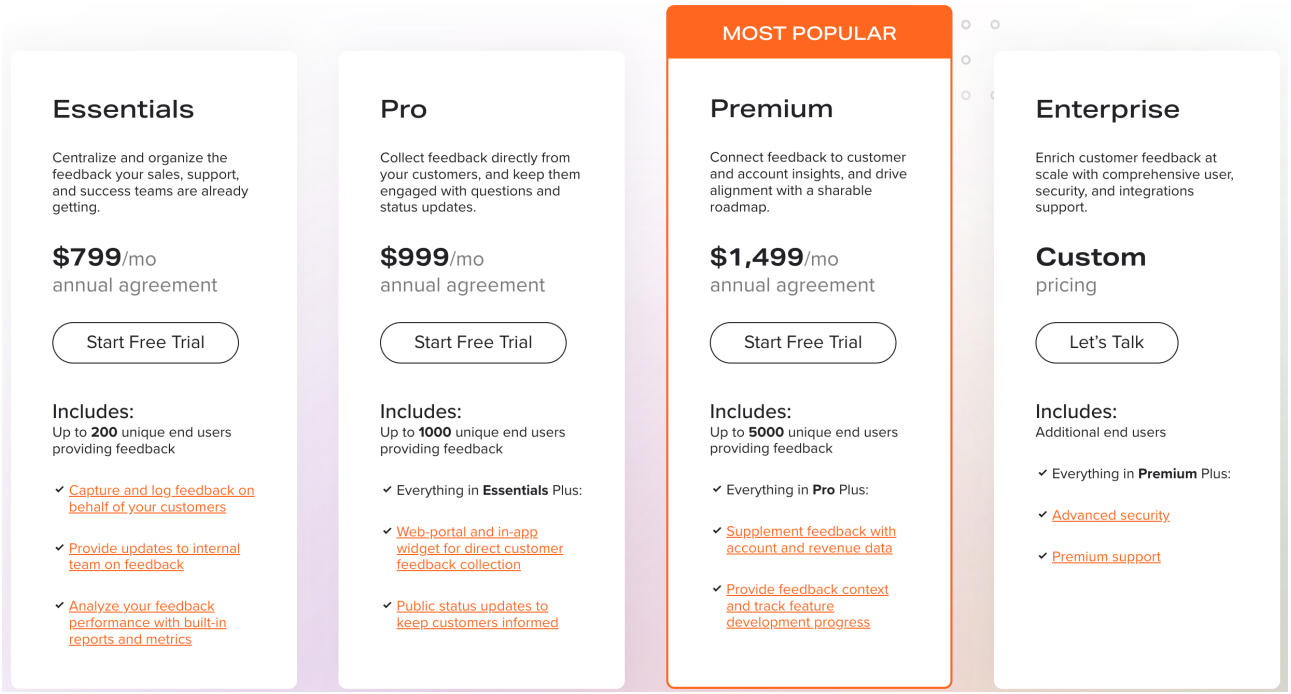


Рисунок 1.8 – Тарифні плани UserVoice

UserVoice представляє собою WEB-застосунок (рис 1.9), який надає миттєвий доступу до автентичних відгуків користувачів, допомагає у використання їх ідей та побажань для розвитку продукту відповідно до їх потреб, що в свою чергу забезпечує відповідності рішень тому, чого дійсно хочуть користувачі, підвищуючи їхню задоволеність.

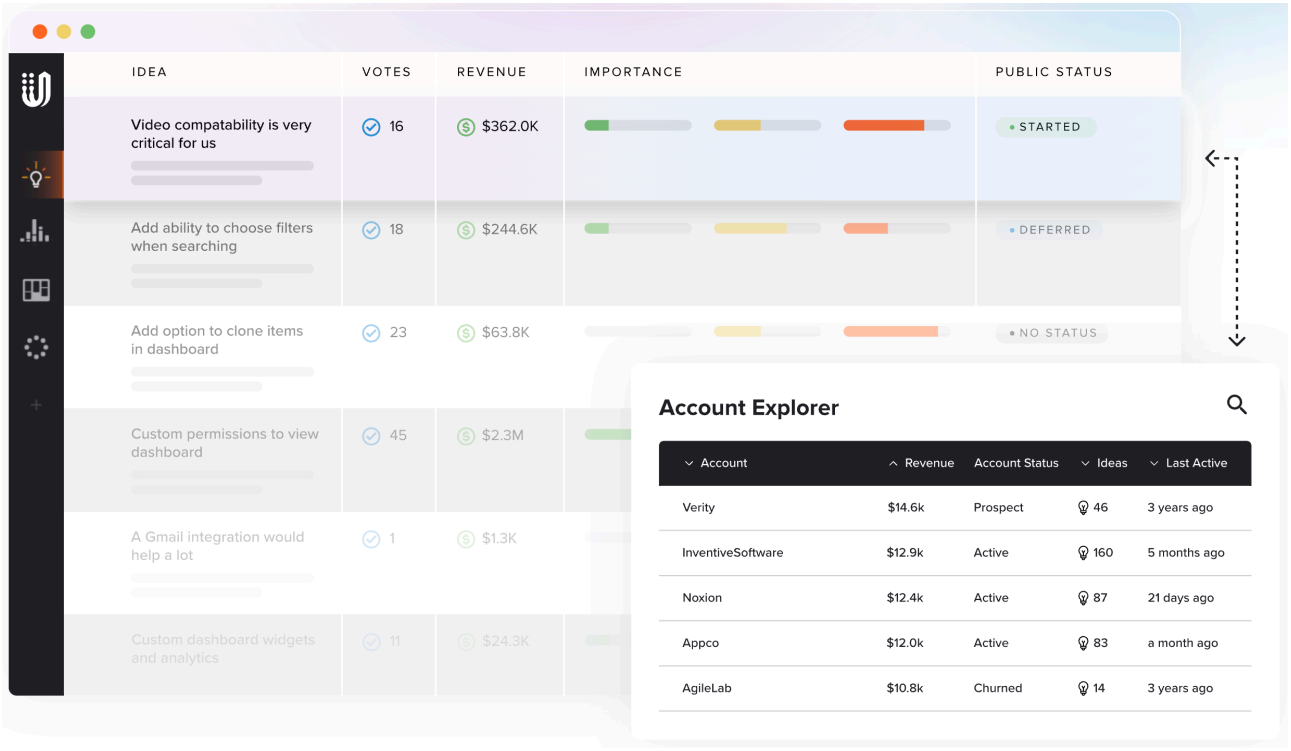


Рисунок 1.9 – Огляд інтерфейсу та функціональних можливостей UserVoice

Таким чином, UserVoice надає цінну інформацію, засновану на даних, від спільноти користувачів, що покращує рішення щодо продуктів і сприяє орієнтованому на користувача підходу до розробки.

ReviewTrackers на головній сторінці свого сайту зазначає, що «успіх вашого бренду залежить від голосу вашого клієнта» [11]. Завдяки своїм функціональним можливостям допомагає створити клієнтоорієнтовану організацію, використовуючи відгуки, та підвищуйте лояльність, утримання та зростання бренду [11].

Це тип додатка SaaS (Software as a Service), який розміщено в хмарі і доступний для користувачів через Інтернет у вигляді мобільного застосунку. ReviewTrackers пропонує зустріч зі спеціалістом з продукту, щоб створити пакет, адаптований до унікальних бізнес-потреб клієнта, отже немає фіксованої ціни. На рисунку 1.10 зображено інтерфейс для огляду відгуків.

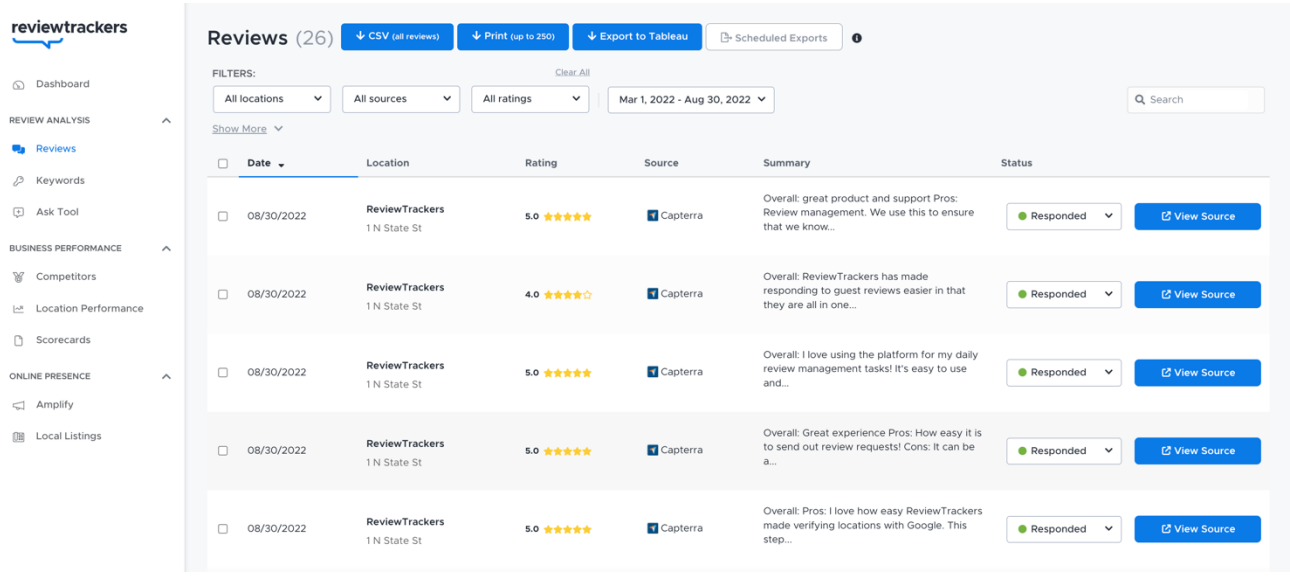


Рисунок 1.10 – Огляд інтерфейсу ReviewTrackers

Основні функціональні можливості ReviewTrackers наступні:

– збір відгуків відбувається в режимі реального часу з різних джерел, включаючи більш ніж 100 веб-сайтів з відгуками;
– всі відгуки можна переглядати і управляти з єдиної платформи, що дозволяє користувачам легко переглядати та реагувати на відгуки;
– використання аналітики та машинне навчання, щоб аналізувати відгуки та надавати корисні узагальнення та візуалізації;
– можливість відповідати на відгуки прямо з платформи;
– сповіщення про нові відгуки через електронну пошту або SMS.

Таким чином, користуючись ReviewTrackers, компанії можуть заощадити час на моніторинг відгуків та краще розуміти своїх клієнтів.

Dataiku спеціалізується на розробці програмного забезпечення для обробки та аналізу великих даних. Їх основний продукт, Dataiku DSS (Data Science Studio), дозволяє швидко розробляти прогнозуючі моделі та інтелектуальні програми. Customer Review Analyzer від Dataiku є інструментом для аналізу відгуків користувачів (рис. 1.11).

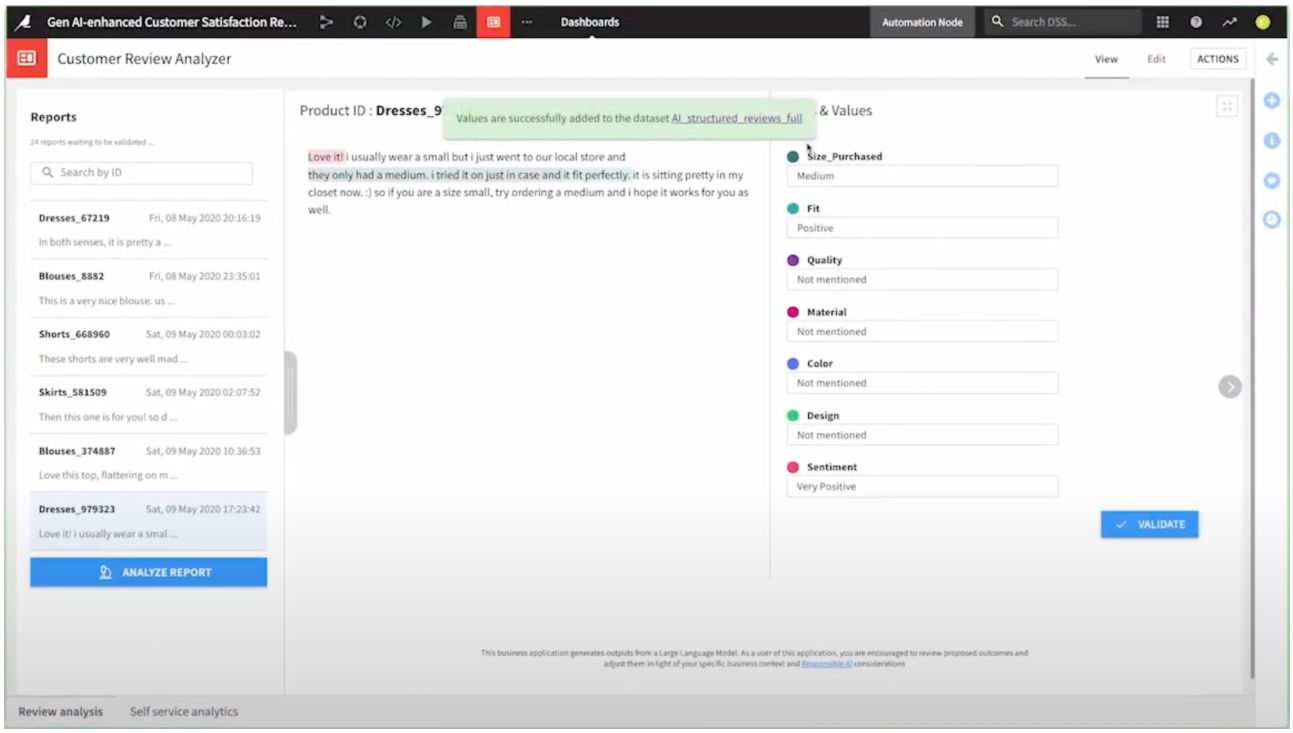


Рисунок 1.11 – Огляд інтерфейсу Customer Review Analyzer від Dataiku

Цей сервіс використовує техніки обробки природної мови (NLP) та машинного навчання (ML) для аналізу тексту відгуків, визначення настроїв і виявлення важливих тем. Основні функції включають:

– машинне навчання для визначення позитивних, негативних або нейтральних настроїв в відгуках користувачів;
– техніки NLP для виявлення ключових тем в відгуках;
– набір графіків і діаграм для візуалізації аналізу відгуків, що допомагає легше розуміти результати;
– інтеграцію з різними джерелами даних для збору відгуків, включаючи соціальні медіа, веб-сайти з відгуками та інше;
– генерація графіків на основі інструкцій користувача (рис. 1.12).

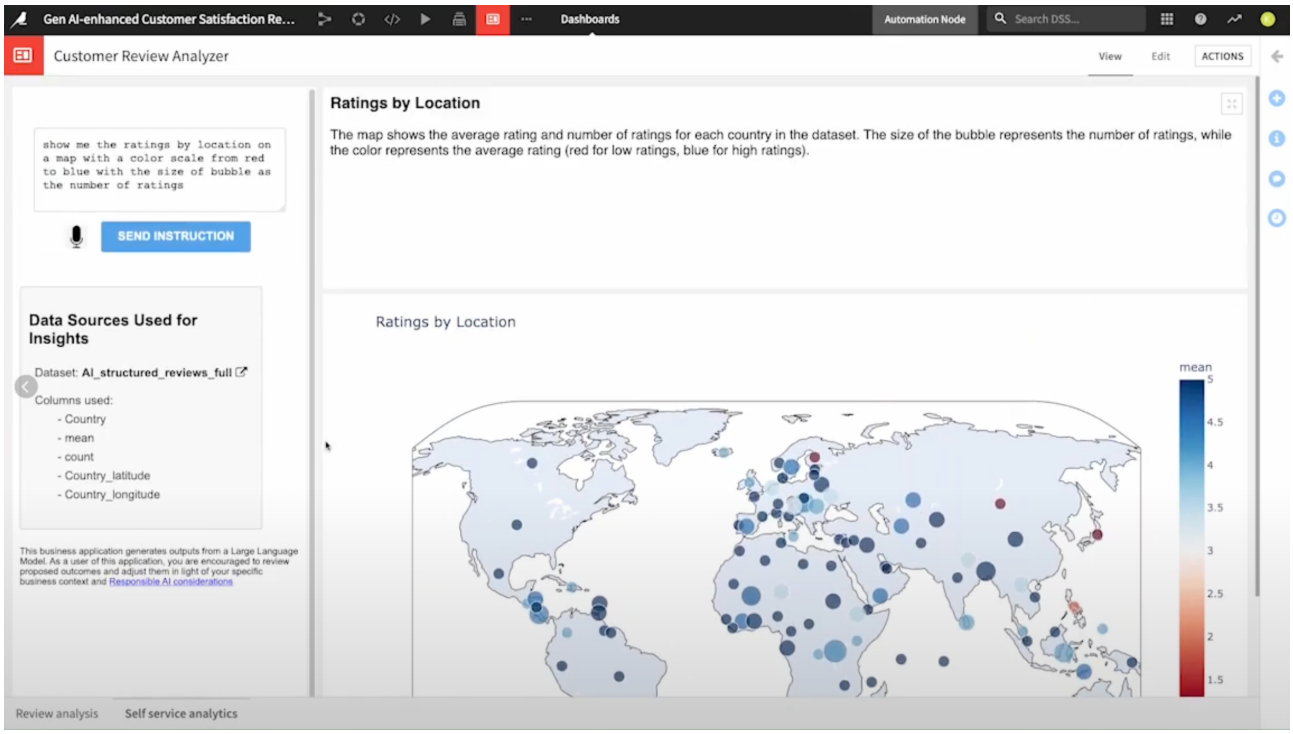


Рисунок 1.12 – Приклад згенерованого графіку на основі заданої інструкції

Отже, усі ці інструменти можуть бути корисним для компаній, які хочуть краще розуміти відгуки своїх користувачів, виявляти проблемні області та можливості для покращення.

Переваги існуючих аналогів:

– Вони автоматично обробляють великі набори даних, значно зменшуючи потребу в ручному введенні даних.
– Вони швидко аналізують великі обсяги відгуків, часто в режимі реального часу.
– Вони ведуть аналіз без людської упередженості або емоційного впливу.

Недоліки існуючих аналогів:

– Існуючі системи можуть не завжди точно визначати тон, контекст або нюанси природної мови в відгуках.
– Деякі відгуки можуть бути багатозначні або містити складні концепції, які важко інтерпретувати для системи.
– Системи потребують постійного навчання і оновлення для покращення точності аналізу.
– Ціни на використання ІС не завжди зрозумілі для клієнта.

1.4 Висновки

Згідно з проведеним вивченням даної тематики, було виявлено, що ринок програмних застосунків продемонстрував тенденцію зростання. До того ж, було підкреслено значущість використання відгуків користувачів як ключового фактора для ефективного розвитку та вдосконалення продукту.

Розробка інтелектуальної системи для аналізу відгуків на програмне забезпечення тягне за собою технічні складнощі, які пов'язані зі зміною технологій, можливість неправильної інтерпретації відгуків через недосконалості в алгоритмах NLP, та проблеми з приватністю та захистом даних при обробці відгуків користувачів.

Крім того, існує ризик високих витрат на проєкт, якщо система не зможе ефективно адаптуватись до змінних вимог або не виконає задані функції. Також є ризик недосягнення очікуваних результатів, якщо система не зможе надати корисної інформації, що може бути використана для покращення програмного забезпечення.

2 ВИБІР ОПТИМАЛЬНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

2.1 Формування системних вимог

Оптимальні інформаційні технології для розробки залежать від вимог до системи. На основі проведеного аналізу можна визначити мінімально життєздатний продукт (MVP). Цей підхід представляє собою одну з технік розробки, де новий продукт створюється з достатнім набором функціональних особливостей для задоволення потреб перших користувачів [12].

Отже, до основних вимог інтелектуальної веб-системи аналізу відгуків на програмні застосунки відноситься:

- наявність зручного та зрозумілого інтерфейсу;
- автоматизований збір відгуків на відповідний застосунку;
- попередня обробка даних (очищення, нормалізація, переклад, видалення стоп-слів і стемінг);
- аналіз настрою відгуків, виявлення ключових тем (інсайтів);
- візуалізація результатів аналізу у вигляді діаграм та графіків;
- фільтрація результатів аналізу за датою та джерелом.

Згідно з MVP методологією, остаточний набір функцій проектується і розробляється тільки після врахування відгуків перших користувачів продукту. Потенційно, це б могли бути вимоги як:

- відправка сповіщення про нові відгуки або важливі зміни в настрої;
- надання відповіді на відгук безпосередньо з платформи;
- інтеграція з існуючими системами компанії замовника;
- управління параметрами аналізу;
- порівняння відгуків на різні програмні застосунки.

Сформовані вимоги відповідають ключовим потребам клієнта та надають цінну інформацію, що може вплинути на прийняття подальших рішень для ефективного розвитку програмного застосунку.

SADT діаграма допомагає визначити основні модулі системи та їх взаємозв’язки, що спрощує розробку, впровадження й подальшу підтримку [13].

Контекстна діаграма А-0 зображена на рисунку 2.1 та описує вхідні та вихідні дані, контролюючі фактори та виконуючі механізми.

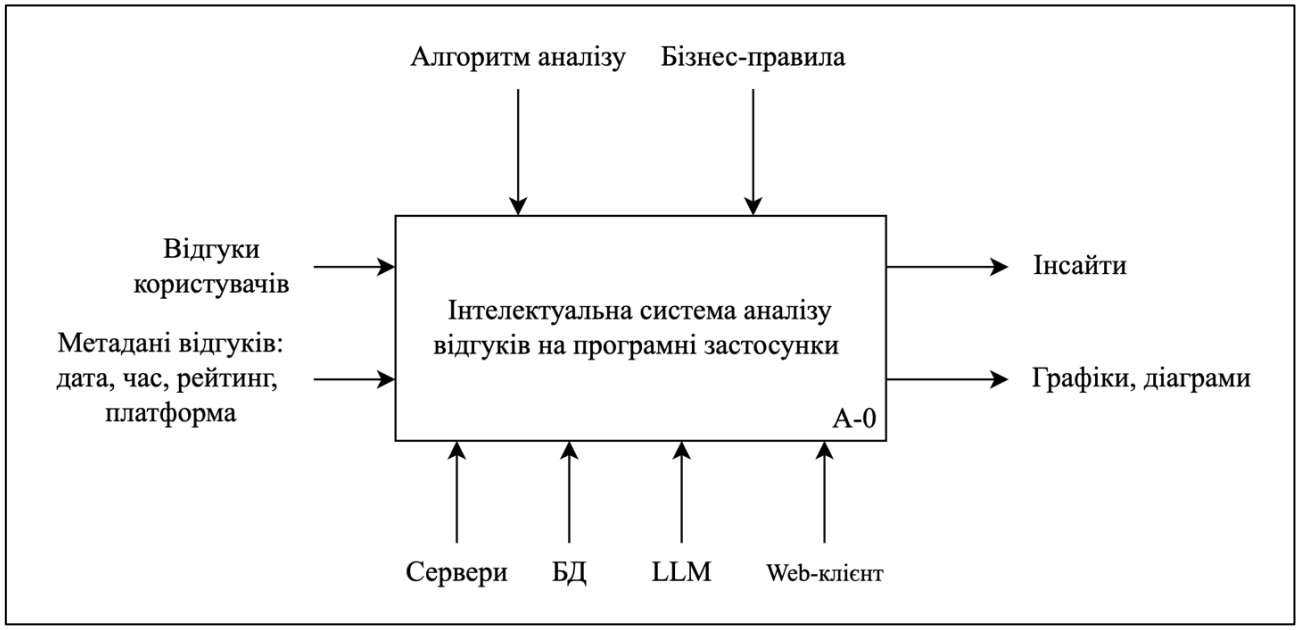


Рисунок 2.1 – Контекстна діаграма А-0

Наступним етапом є декомпозиція головної функції на більш детальні діаграми (рис. 2.2) на основі яких чітко видно усі наявні модулі системи.

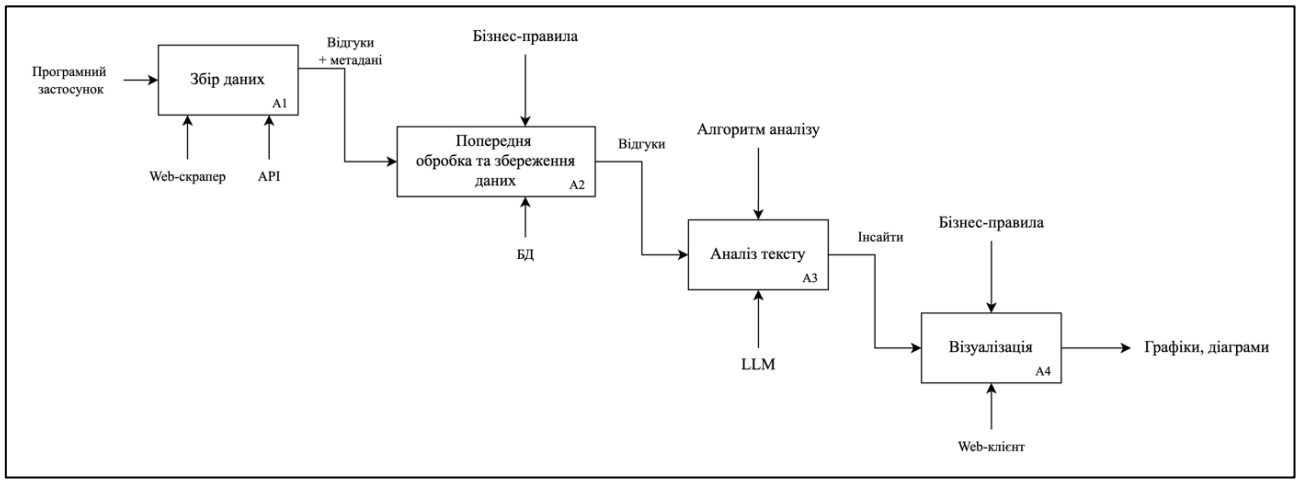


Рисунок 2.2 – Діаграма модулів системи

Використання веб-архітектури для розробки інтелектуальної системи аналізу відгуків на програмні застосунки має багато переваг, серед яких доступність і універсальність. Веб-додатки можна використовувати на будь-

якому пристрої з браузером та інтернет-з'єднанням, що робить їх доступними для користувачів з будь-якого місця і в будь-який час. Вони також працюють на різних операційних системах, таких як Windows, macOS, Linux, iOS та Android, що забезпечує їхню кросплатформеність і зручність для широкої аудиторії.

Масштабованість і гнучкість веб-архітектури дозволяють легко розширювати систему у міру зростання кількості користувачів та обсягу даних. Використання мікросервісної архітектури сприяє поділу системи на окремі сервіси, що полегшує її обслуговування та оновлення. Веб-додатки також легко інтегруються з іншими сервісами та API, що дозволяє використовувати сторонні інструменти для збору відгуків, аналізу даних або машинного навчання. Технології, такі як WebSockets, забезпечують оновлення даних у реальному часі, що є важливим для своєчасного аналізу відгуків.

Сучасні веб-додатки можуть забезпечувати високий рівень безпеки даних завдяки використанню технологій шифрування (SSL/TLS) та автентифікації (OAuth). Централізоване адміністрування веб-архітектури дозволяє легко оновлювати систему та впроваджувати нові функції без необхідності оновлювати кожен клієнтський додаток окремо. Це підвищує зручність як для розробників, так і для кінцевих користувачів.

Веб-додатки також забезпечують високий рівень користувацького досвіду завдяки адаптації до мобільних пристроїв і можливості створення інтуїтивно зрозумілого інтерфейсу за допомогою сучасних front-end фреймворків. Це робить їх зручними для користувачів, які вважають за краще використовувати смартфони та планшети, а також для тих, хто очікує високого рівня інтерактивності та зручності у користуванні.

Отже, веб-архітектура є ідеальним вибором для створення інтелектуальних систем аналізу відгуків, забезпечуючи доступність, масштабованість, безпеку, гнучкість та зручність для користувачів. Це дозволяє створювати потужні та ефективні системи аналізу даних, які відповідають сучасним вимогам та очікуванням.

2.2 Огляд інформаційних технологій для розробки веб-додатків

Веб-додаток використовує веб-сервер для взаємодії з користувачами через веб-браузер, працюючи в реальному часі, вони не потребують завантаження та інсталяції на пристрій користувача, так як вони доступні через Інтернет [14].

Full-stack архітектура представляє собою систему, яка включає всю інфраструктуру, необхідну для роботи веб-додатку [15]. Ось декілька популярних інструментів та технологій, які розробники використовують для створення веб-додатків:

- JavaScript / TypeScript + Node.js: Мови програмування JavaScript та TypeScript широко використовуються для розробки веб-додатків. Node.js - це середовище, яке дозволяє виконувати JavaScript на сервері.
- React.js / Angular / Vue.js: Це JavaScript бібліотеки/фреймворки, які використовуються для створення інтерактивних інтерфейсів користувача на стороні клієнта.
- Django / Flask: Це Python фреймворки, які використовуються для розробки веб-додатків.
- Ruby on Rails: Це фреймворк, базований на Ruby, який використовується для розробки веб-додатків.
- Laravel / Symfony: Це PHP фреймворки, які використовуються для розробки веб-додатків.
- ASP.NET: Це фреймворк від Microsoft, який використовується для розробки веб-додатків на C#.
- Express.js: Це легкий фреймворк для Node.js, який використовується для розробки веб-додатків.
- MongoDB / PostgreSQL / MySQL: Це системи керування базами даних, що використовуються для зберігання та отримання даних.
- Docker: Це платформа, що використовується для контейнеризації додатків, що дозволяє їм працювати у будь-якому оточенні.
- Git: Це система контролю версій, яка дозволяє кільком розробникам працювати над одним проєктом, відстежуючи зміни коду.

Інструменти та технології, які вибираються для конкретного веб-проекту, залежать від його вимог, а також від досвіду та переваг розробників. Full-stack розробники можуть працювати над всіма аспектами проекту, а саме: frontend (інтерфейс користувача) та backend (серверна частина). Це забезпечує ефективність та економію часу, однак, вони не завжди мають глибокі знання в окремих областях. Через головоломку з декількома технологіями, full-stack розробники можуть іноді не мати достатнього часу для оновлення своїх навичок у всіх областях.

IDE (Integrated Development Environment) – це інтегроване середовище розробки, яке надає набір інструментів для створення програмного забезпечення. До таких інструментів як правило входять текстовий редактор для написання коду, інструменти для побудови та відлагодження програм, а також інші корисні функції, такі як система контролю версій. Різні середовища розробки (IDE) призначені для використання з різними мовами програмування та стеками технологій. Ось декілька популярних IDE для веб-розробки:

- Visual Studio Code: безкоштовне IDE від Microsoft, яке підтримує безліч мов програмування, включаючи JavaScript, TypeScript, Python, C#, PHP і багато інших. VS Code має велику кількість розширень, що надають додаткові можливості, такі як підтримка фреймворків та бібліотек, інтеграція з Git, підсвічування синтаксису, автодоповнення коду тощо.
- Sublime Text: це потужний і гнучкий текстовий редактор, який часто використовується як IDE для різних веб-технологій.
- Atom: безкоштовне IDE від GitHub, яке підтримує багато мов програмування та має велику кількість розширень.
- WebStorm: комерційне IDE від JetBrains, яке спеціально розроблено для JavaScript розробки, включаючи Node.js та фронтенд JavaScript фреймворки і бібліотеки, такі як React, Angular, Vue.js тощо.
- PyCharm: комерційне IDE від JetBrains, яке розроблено спеціально для Python, включаючи Django та Flask, і підтримує веб-розробку на Python.

Кожне середовище має свої сильні та слабкі сторони, тому вибір IDE залежить від потреб та вподобань конкретного розробника.

Visual Studio Code (VS Code) є одним з найпопулярніших інтегрованих середовищ розробки (IDE). На рисунку 2.3 зображено його інтерфейс.

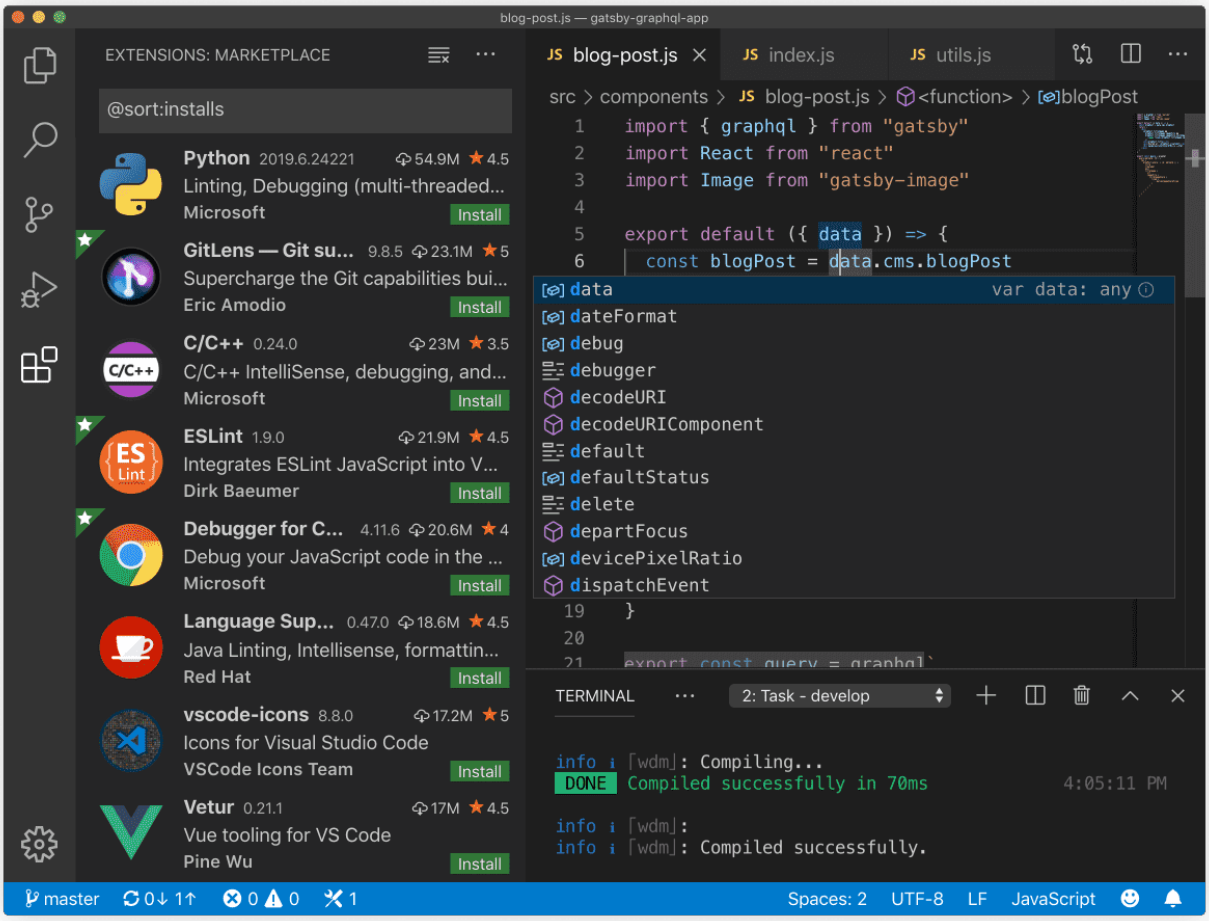


Рисунок 2.3 – Огляд IDE VS Code

Ось декілька причин, чому варто обрати саме його:

– Безкоштовність: VS Code – це вільне й відкрите програмне забезпечення його можна використовувати безкоштовно і навіть вносити зміни в вихідний код.

– Підтримка множини мов програмування: VS Code підтримує багато мов програмування "з коробки" і ще більше з допомогою розширень.

– Розширення: В VS Code доступні тисячі розширень, які додають нові можливості та покращують існуючі. Це дозволяє налаштувати середовище розробки під конкретні потреби.

– Інтеграція з Git: VS Code має вбудовану підтримку Git, що дозволяє

легко відстежувати зміну коду, створювати коміти та синхронізуватися з віддаленими репозиторіями прямо з IDE.

– Debugging: VS Code має вбудовану підтримку відлагодження, що дозволяє перевіряти код на наявність помилок без встановлення додаткових інструментів.
– Термінал: Вбудований термінал в VS Code дозволяє виконувати команди безпосередньо з IDE, що робить процес розробки швидшим і ефективнішим.
– Підтримка IntelliSense: IntelliSense в VS Code підтримує автодоповнення коду, визначення функцій та змінних, перегляд пов'язаних файлів і багато інших корисних функцій.
– Портативність: VS Code можна встановити на будь-якій операційній системі і мати однаковий досвід розробки.
– Спільнота: Велика, активна спільнота VS Code означає, що користувач завжди зможете знайти допомогу.

2.3 Огляд мови програмування TypeScript

TypeScript – це надмножина JavaScript (рис. 2.4), що додає статичну типізацію і інші функції, які допомагають зробити код більш надійним та легким для розуміння і підтримки [16].

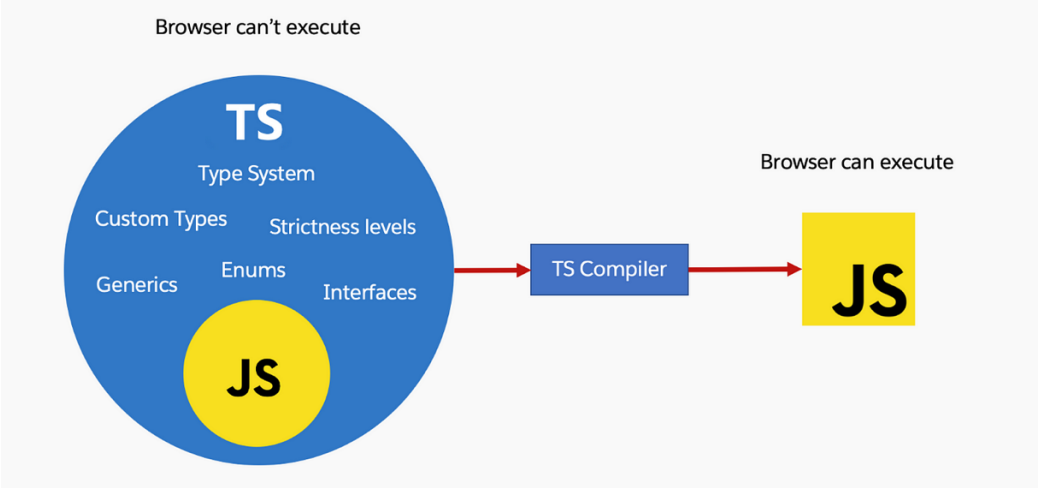


Рисунок 2.4 – TypeScript надмножина JavaScript

Розробка TypeScript почалася у Microsoft в кінці 2010 року. Основним рушієм створення цієї мови стала необхідність вирішення проблем, з якими стикалися великі проекти, написані на JavaScript. У той час JavaScript використовувався для розробки великих додатків, і його динамічна типізація створювала труднощі в управлінні складними кодовими базами, особливо коли йшлося про масштабування та підтримку таких проектів.

TypeScript був офіційно анонсований у жовтні 2012 року Андерсом Хейлсбергом, відомим розробником, який раніше створив Turbo Pascal і керував розробкою C#. Перший випуск TypeScript 0.8 вийшов у жовтні 2012 року. Мова відразу привернула увагу розробників завдяки своїй здатності підвищити продуктивність та якість коду за рахунок статичної типізації.

Одна з головних переваг TypeScript – це статична типізація, яка дозволяє визначати типи змінних, функцій і об'єктів. Статична типізація допомагає уникнути багатьох помилок на етапі компіляції, знижуючи ризик помилок під час виконання. Завдяки типізації код стає більш передбачуваним і зрозумілим, що полегшує його підтримку та розвиток, особливо в великих командах.

Інтеграція TypeScript з інструментами для розробки, такими як Visual Studio Code, також сприяла його популярності. Visual Studio Code, який також розроблений Microsoft, надає чудову підтримку TypeScript, включаючи автодоповнення, рефакторинг і налагодження, що значно підвищує продуктивність розробників.

TypeScript став невід'ємною частиною сучасної веб-розробки, надаючи потужні інструменти для створення надійного та масштабованого коду. Завдяки своїм перевагам, таким як статична типізація, сумісність з JavaScript та активна підтримка спільноти, TypeScript продовжує зростати в популярності і використовується у все більшій кількості проектів.

2.4 Огляд фреймворку для розробки веб-додатків Next.js

Сьогодні Next.js є одним з найпопулярніших фреймворків для розробки веб-додатків на React. Його використовують такі великі компанії, як Netflix,

TikTok, Hulu та багато інших. Next.js продовжує активно розвиватися, регулярно додаючи нові функції та покращення продуктивності.

Next.js був створений компанією Vercel (раніше Zeit) і вперше випущений у жовтні 2016 року. Основною метою фреймворку було спростити розробку серверних рендерингів React-додатків. Це вирішувало численні проблеми, з якими стикалися розробники при використанні React для створення складних веб-додатків [17].

До основних можливостей фреймворку належить:

– серверний рендеринг (SSR) дозволяє покращити SEO та зменшити час завантаження сторінок, оскільки HTML-сторінки генеруються на сервері перед надсиланням клієнту;
– статична генерація (SSG) дозволяє попередньо рендерити сторінки під час збірки, що забезпечує високу продуктивність і масштабованість, оскільки попередньо згенеровані сторінки можуть обслуговуватися CDN.

Таким чином Next.js дозволяє створювати гібридні додатки, які поєднують серверний рендеринг та статичну генерацію, а розробники можуть обирати найкращий підхід для кожної сторінки залежно від потреб.

Окрім цього, фреймворк підтримує TypeScript “з коробки”, має вбудований маршрутизатор, автоматичну оптимізацію зображень і підтримку API-маршрутів. Також Next.js може бути використаний для створення прогресивних веб-додатків (PWA), що поєднують переваги традиційних веб-сайтів та мобільних додатків.

Завдяки своїм можливостям і активному розвитку, Next.js став ключовим інструментом для багатьох розробників і компаній, які прагнуть створювати високопродуктивні, масштабовані та SEO-оптимізовані веб-додатки.

Next.js варто розглянути замість простого React, якщо потрібен рендеринг на стороні сервера, статична генерація сайту, спрощена маршрутизація, покращена продуктивність або вбудовані функції для веб-застосунку. Однак, якщо додаток вимагає високого рівня кастомізації, використання лише React може бути більш доцільним.

Якщо додаток сильно залежить від оновлень в реальному часі, наприклад, біржа, чат-додаток або багатокористувацька гра, то Next.js може бути не найкращим вибором. Хоча Next.js може обробляти оновлення в реальному часі за допомогою клієнтського JavaScript, він не оптимізований для управління з'єднаннями в реальному часі або обробки великих обсягів одночасних запитів.

Якщо проєкт вимагає складної анімації, взаємодії або складних макетів, Next.js може виявитися обмеженим. Хоча React надає гнучку основу для створення кастомних компонентів інтерфейсу, Next.js накладає певні умовності та абстракції, які можуть обмежити вашу здатність реалізовувати висококастомізовані дизайни. У таких випадках використання React з легким пакувальником, таким як Webpack, може бути більш доречним.

Next.js в першу чергу орієнтований на створення front-end, тому якщо проект включає важку бекенд-логіку, архітектуру мікросервісів або складну обробку на стороні сервера, потрібно також використовувати фреймворки або бібліотеки, розроблені спеціально для back-end-розробки.

2.5 Огляд можливостей платформи Node.js та фреймворку Nest.js

Node (або більш формально Node.js) - це кросплатформенне середовище виконання з відкритим вихідним кодом, яке дозволяє розробникам створювати всілякі серверні інструменти та додатки на JavaScript. Середовище виконання призначене для використання поза контекстом браузера (тобто безпосередньо на комп'ютері або серверній ОС). Таким чином, середовище опускає специфічні для браузера JavaScript API і додає підтримку більш традиційних API ОС, включаючи HTTP і бібліотеки файлової системи.

Node.js був створений Раяном Далем і вперше представлений у 2009 році. Головною метою створення Node.js було розширення можливостей JavaScript за межі браузера і забезпечення серверного виконання JavaScript-коду. В основі Node.js лежить V8 - високопродуктивний JavaScript-рушій, розроблений Google для браузера Chrome. Завдяки використанню V8, Node.js забезпечує високу швидкість виконання коду (рис. 2.5).

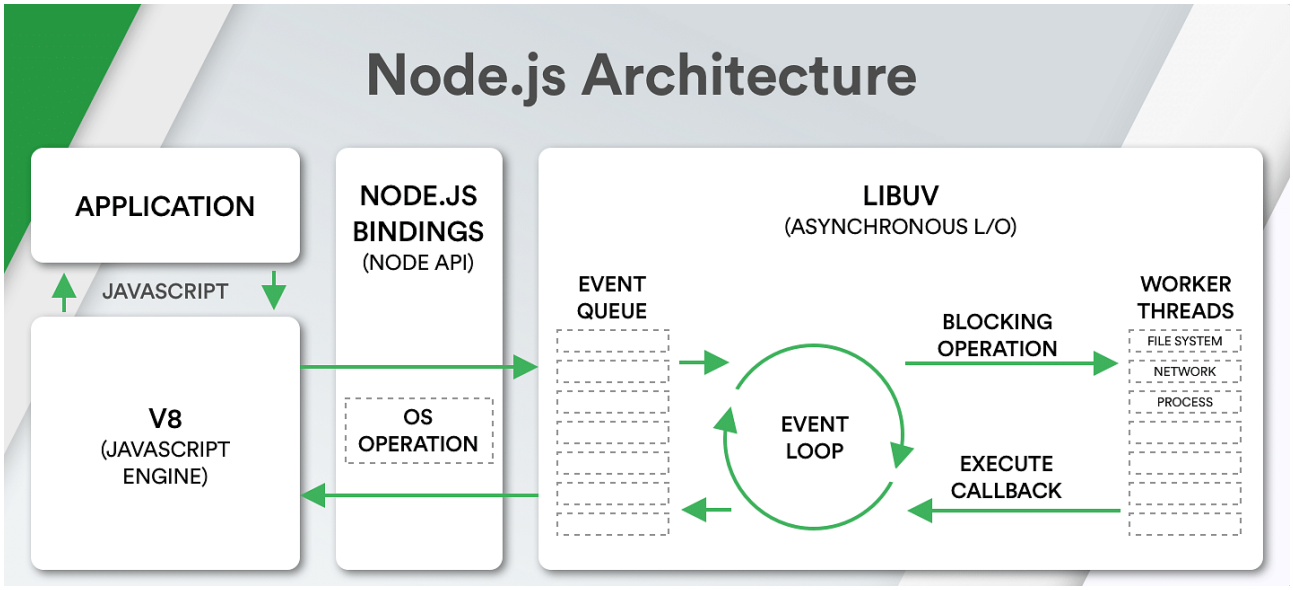


Рисунок 2.5 – Блок-схема архітектури Node.js

Node.js використовує подієву модель і неблокуючий ввід/вивід (I/O), що дозволяє обробляти велику кількість одночасних з'єднань з високою продуктивністю. Це особливо корисно для додатків, що потребують обробки багатьох запитів одночасно, таких як веб-сервери або системи реального часу.

Node.js працює на одному потоці, але використовує асинхронні виклики для виконання операцій введення/виведення. Це дозволяє уникнути блокування потоку і забезпечує високу продуктивність для I/O-інтенсивних додатків.

Завдяки своїй подієвій моделі та неблокуючій архітектурі, Node.js забезпечує високу масштабованість і продуктивність. Його легко масштабувати як вертикально (додавання ресурсів до окремих вузлів), так і горизонтально (додавання нових вузлів до системи).

З точки зору розробки веб-серверів Node має ряд переваг.

– Відмінна продуктивність! Node був розроблений для оптимізації пропускної здатності та масштабованості веб-додатків і є хорошим рішенням для багатьох поширених проблем веб-розробки (наприклад, веб-додатків у режимі реального часу).

- Код написаний на «старому доброму JavaScript», що означає, що менше часу витрачається на «зміну контексту» між мовами, коли розробник пишете код на стороні клієнта і на стороні сервера.
- JavaScript є відносно новою мовою програмування і виграє від покращення мовного дизайну у порівнянні з іншими традиційними веб-серверними мовами (наприклад, Python, PHP та ін.). Багато інших нових і популярних мов компілюються/конвертуються у JavaScript, тому можна використовувати TypeScript, CoffeeScript, ClojureScript, Scala, LiveScript тощо.
- NPM (node package manager) надає доступ до сотень тисяч бібліотек з відкритим вихідним кодом. Він також має найкращу у своєму класі роздільну здатність залежностей і може бути використаний для автоматизації більшої частини інструментарію збірки.
- Node.js є портативним. Він доступний на Microsoft Windows, macOS, Linux, Solaris, FreeBSD, OpenBSD, WebOS та NonStop OS. Крім того, вона добре підтримується багатьма хостинг-провайдерами, які часто надають спеціальну інфраструктуру та документацію для розміщення сайтів на Node.

Отже, Node.js є потужною платформою для розробки серверних додатків, яка забезпечує високу продуктивність і масштабованість завдяки своїй подієвій моделі і неблокуючому вводу/виводу. З великим числом доступних бібліотек та активною спільнотою, Node.js продовжує залишатися популярним вибором для розробників, що створюють сучасні веб-додатки, додатки реального часу та інші високонавантажені системи.

NestJS – прогресивний фреймворк Node.js для створення ефективних, надійних і масштабованих серверних додатків за допомогою мови програмування TypeScript. Він надає розробникам велику розвинену екосистему технологій, котра дозволяє робити інтеграції з багатьма різними інструментами, котрі зазвичай використовуються при розробці, таких як Redis, BullMQ, реляційні і нереляційні бази даних та інші [18].

Основні особливості та переваги Nest.js:

- використання TypeScript за замовчуванням;
- організація коду за допомогою модулів, що полегшує управління великими проектами, сприяє повторному використанню коду і робить систему легкою для тестування;
- використання декораторів та інверсію контролю для полегшення управління залежностями та налаштування сервісів;
- підтримує різних транспортних протоколів, таких як HTTP, WebSockets, GraphQL, і мікросервіси, що дозволяє створювати різноманітні додатки, включаючи додатки реального часу та розподілені системи;
- завдяки модульній структурі та інверсії контролю, Nest.js спрощує написання тестів, що підвищує якість і надійність коду.

Таким чином, використання Nest.js разом з Node.js приносить численні переваги, зокрема покращену організацію коду, прискорення розробки, забезпечення масштабованості та полегшення управління залежностями. Завдяки своїй модульній архітектурі, підтримці TypeScript та інтеграції з популярними бібліотеками, Nest.js допомагає розробникам створювати ефективні, надійні та масштабовані серверні додатки. Усе це робить Nest.js відмінним вибором для розробників, які прагнуть максимально використовувати можливості Node.js.

2.6 Інформаційні технології аналізу текстових даних

Для застосування класичних методів аналізу текстових даних потрібно мати гарне розуміння машинного навчання, обробки природного мовлення (NLP), статистики та лінгвістики. Також необхідними є навички програмування на Python – основною мовою для обробки даних та машинного навчання. Вона має безліч бібліотек для обробки даних, статистики, машинного навчання та NLP, включаючи Pandas, NumPy, SciPy, Scikit-Learn, NLTK, TextBlob, SpaCy,

Gensim та інші. Нижче наведено перелік основних методів, за допомогою яких виконується аналіз тексту.

- Сентимент-аналіз: можна використовувати бібліотеки NLP, такі як TextBlob або NLTK як основу для моделі машинного навчання, яка класифікує відгуки на позитивні, негативні та нейтральні.
- Тематичне моделювання дозволяє виявити основні теми у відгуках, для цього можуть використовуватися такі методики, як метод головних компонент (PCA) або латентний розподіл Діріхле (LDA), що доступний через такі бібліотеки як Gensim.
- Аналіз контенту: бібліотеки NLP, такі як NLTK або Spacy, можна використати для токенізації та стемінгу для отримання "сирих" слів і фраз для аналізу.
- Ізоляція ключових слів: це спосіб знаходження найбільш значущих слів в тексті. Зазвичай це слова, які найчастіше зустрічаються або які найбільш показові для аналізованої теми. Можна використовувати алгоритми як TF-IDF (частота терміну - обернена частота документів) для відбору ключових слів в тексті.
- Алгоритми машинного навчання також часто використовуються в аналізі текстів. Поверхневе навчання (supervised learning) та навчання без вчителя (unsupervised learning) включають класифікацію тексту, кластеризацію та ін. Бібліотеки як Scikit-learn можуть бути використані для навчання моделей на відміченому датасеті.
- Методики обробки природних мов (NLP) використовуються для більш глибокого аналізу. Зокрема, бібліотеки Spacy або NLTK можуть використовуватися для POS-тегування, синтаксичного аналізу відношень тощо.

Виконання подібних аналізів може бути важким і потребувати значних обчислювальних ресурсів, особливо при роботі з великими наборами даних. Для цього, може бути необхідним використання високопродуктивних комп'ютерів або обчислювальні хмарні ресурси, особливо для тренування складних моделей машинного навчання.

В ході розробки інтелектуальної системи було прийнято рішення піти шляхом використання попередньо натренованої моделі від OpenAI, а саме GPT-4. Такий підхід відомий як навчання з перенесенням (transfer learning) – це метод машинного навчання (ML), в якому знання, отримані при виконанні завдання, повторно використовуються для підвищення продуктивності при виконанні пов'язаного завдання. Це дозволяє економити значний час та обчислювальні ресурси, які були б необхідні для тренування масштабної моделі із нуля.

Однак попередньо натреновані моделі можуть не давати високу точність при роботі з конкретними даними або в специфічному контексті. В таких випадках точність залежить від якості формулювання запиту та наявності відповідних підказок.

Підказка – це вхідні дані для LLM. Вони, як правило, генеруються динамічно при використанні в додатку LLM і включають вхідні дані користувача (питання), набір декількох прикладів, які допомагають мовній моделі генерувати кращу відповідь, та інструкції для LLM щодо того, як обробити вхідні дані, які надходять від користувача [19].

LangChain – це фреймворк для розробки додатків, що використовують великі мовні моделі, і його мета – дозволити розробникам зручно використовувати інші джерела даних і взаємодіяти з іншими додатками. Для цього LangChain надає компоненти (модульні абстракції) та ланцюжки (конвеєри, що налаштовуються для конкретних сценаріїв використання). На рисунку 2.6 схематично зображено усі наявні модулі та пакети фреймворку.

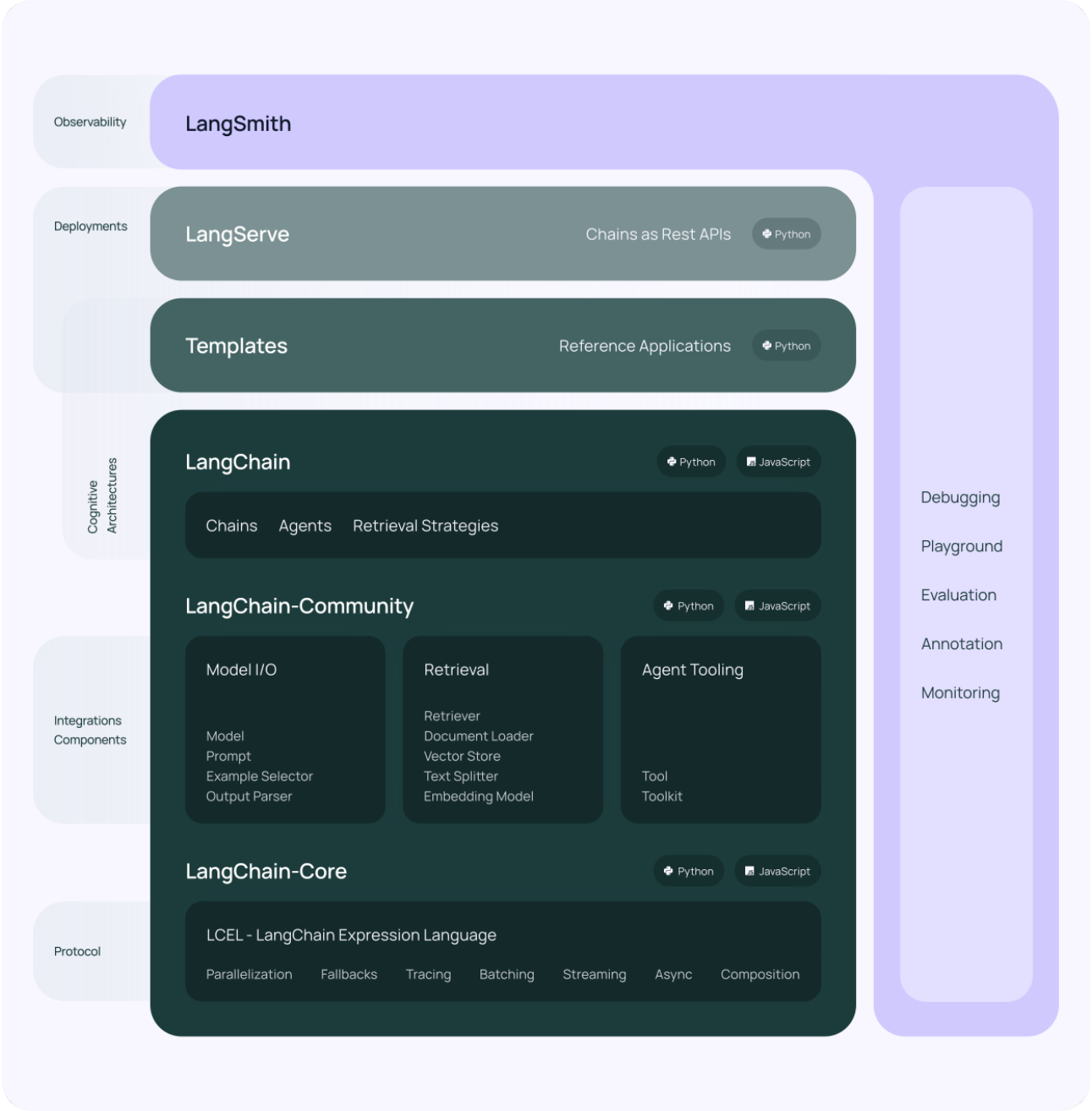


Рисунок – 2.6 – Модулі та пакети екосистеми LangChain

Великі мовні моделі (LLM) є основним типом моделей, що використовуються в LangChain. Вони приймають текстовий рядок (підказку) і виводять текстовий рядок. LangChain надає кілька класів для побудови підказок з використанням декількох спеціалізованих шаблонів підказок. Шаблон підказки відноситься до відтворюваного способу генерації підказки. Він містить текстовий рядок («шаблон»), який може приймати набір параметрів від кінцевого користувача і генерувати запит.

Існують також інші типи моделей, які використовуються в LangChain, а саме: чат-моделі та моделі вбудовування тексту. Моделі чату мають більш структурований API для обробки повідомлень чату, а моделі вбудовування тексту приймають текст і повертають його відповідне вбудовування у вигляді списку з плаваючими числами. Вбудовування потрібні, коли ми хочемо працювати з нашими власними документами.

Для створення об’єкту моделі, використовуючи LangChain, достатньо лише кілька стрічок коду (рис. 2.7).

```
13    private chat(options?: Partial<{ temperature: number; modelName: string }>) {
14      return new ChatOpenAI({
15        temperature: options.temperature ?? 0,
16        modelName: options.modelName ?? 'gpt-3.5-turbo',
17      });
18    }
```

Рисунок 2.7 – Метод для створення чату OpenAI на базі моделі GPT

LangChain, завдяки своїй здатності оптимізувати процес розробки LLM-додатків, продемонстрував значний потенціал в екосистемі штучного інтелекту. Він започаткував новий підхід, який дозволяє розробникам без особливих зусиль взаємодіяти з різними джерелами даних і додатками. Ця гнучкість та ефективність, що характеризуються модульними абстракціями LangChain та конвеєрами, що налаштовуються під конкретні випадки використання, роблять його безцінним інструментом для майбутнього LLM-додатків [19].

2.7 Висновки

В цьому розділі було сформовано системні вимоги та основні модулі інтелектуальної веб-системи аналізу відгуків на програмні застосунки, серед яких автоматичний збір даних, попередня обробка та збереження, аналіз та візуалізація результатів.

Проведено огляд інформаційних технологій для розробки веб-додатків. Для розробки обрано мову програмування TypeScript, яка використовується для написання front-end на Next.js та back-end на Nest.js.

Для аналізу текстових даних застосовано метод навчання з перенесенням (transfer learning), використовуючи фреймворк LangChain та модель GPT-4.

3 РОЗРОБЛЕННЯ ІНТЕЛЕКТУАЛЬНОЇ ВЕБ-СИСТЕМИ АНАЛІЗУ ВІДГУКІВ НА ПРОГРАМНІ ЗАСТОСУНКИ

3.1 Розроблення архітектури

На рисунку 3.1 зображено архітектуру інтелектуальної веб-системи на базі платформи Node.js.

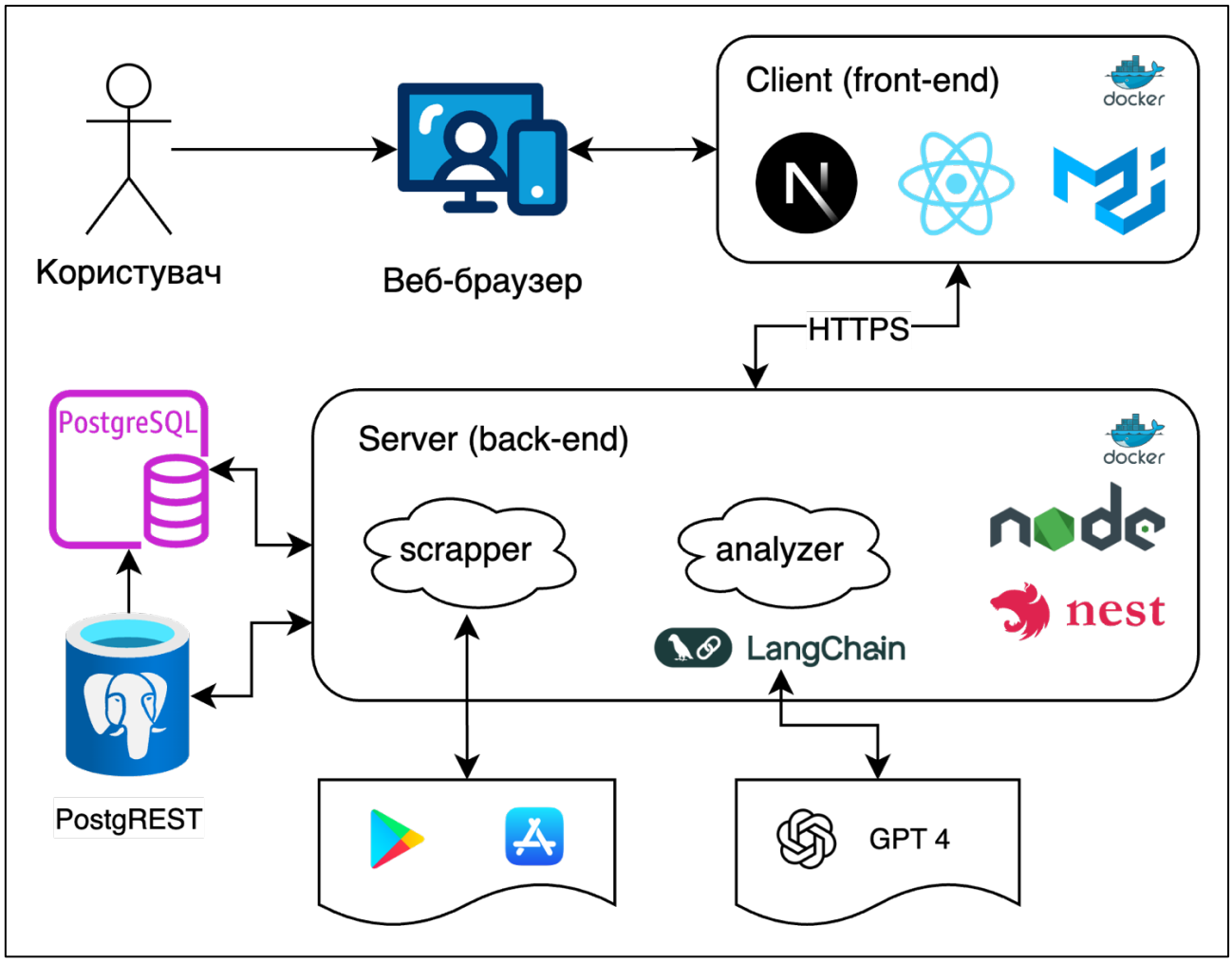


Рисунок 3.1 – Архітектура веб-додатку інтелектуальної системи

Відповідно до розробленої архітектури ми бачимо, що взаємодія відбувається за допомогою веб-браузера, який обробляє запити користувача та за допомогою безпечного протоколу HTTPS обмінюється даними з сервером.

Контейнеризація за допомогою Docker гарантує ізоляцію, переносимість і ефективність розробки додатку на локальному рівні. Застосування потужної і

надійної бази даних PostgreSQL дозволяє обробляти великі об'єми даних та складні запити. В системі вона використовується для зберігання відгуків та результатів їх аналізу [1].

Основним джерелом даних є маркетплейси Google Play та App Store, де розміщуються програмні застосунки на які будь-який користувач може залишити текстовий відгук та поставити оцінку.

Діаграма Use-Case інтелектуальної системи аналізу відгуків на програмні застосунки буде мати вигляд, як зображено на рисунку 3.2.

Рисунок 3.2 - Діаграма Use-Case інтелектуальної системи

3.2 Програмна реалізація

Для правильної ініціалізації Next.js додатку, варто виконати відповідну команду в термінал та слідувати інструкціям (рис. 3.3).

```
➜  ~ npx create-next-app@latest
✔ What is your project named? … review-analyzer
✔ Would you like to use TypeScript? … No / Yes
✔ Would you like to use ESLint? … No / Yes
✔ Would you like to use Tailwind CSS? … No / Yes
✔ Would you like to use `src/` directory? … No / Yes
✔ Would you like to use App Router? (recommended) … No / Yes
✔ Would you like to customize the default import alias (@/*)? … No / Yes
✔ What import alias would you like configured? … @/*
Creating a new Next.js app in /Users/Vladyslav_Pobidash/review-analyzer.

Using npm.

Initializing project with template: app-tw


Installing dependencies:
- react
- react-dom
- next

Installing devDependencies:
- typescript
- @types/node
- @types/react
- @types/react-dom
- postcss
- tailwindcss
- eslint
- eslint-config-next

npm WARN deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use
 it. Check out lru-cache if you want a good and tested way to coalesce async requests by a key
 value, which is much more comprehensive and powerful.
npm WARN deprecated rimraf@3.0.2: Rimraf versions prior to v4 are no longer supported
npm WARN deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported

added 359 packages, and audited 360 packages in 24s

133 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
Initialized a git repository.

Success! Created review-analyzer at /Users/Vladyslav_Pobidash/review-analyzer
```

Рисунок 3.3 – Ініціалізація Next.js додатку

Щоб зробити проєкт зрозумілим і структурованим, особливо в умовах регулярної зміни вимог бізнесу варто визначити звід правил і угод щодо організації коду. Feature-Sliced Design (FSD) – це архітектурна методологія для проектування frontend-додатків [20]. Проєкт на FSD складається з шарів (layers), кожен шар складається зі слайсів (slices), і кожен слайс складається із сегментів (segments), що і зображено на рисунку 3.4.

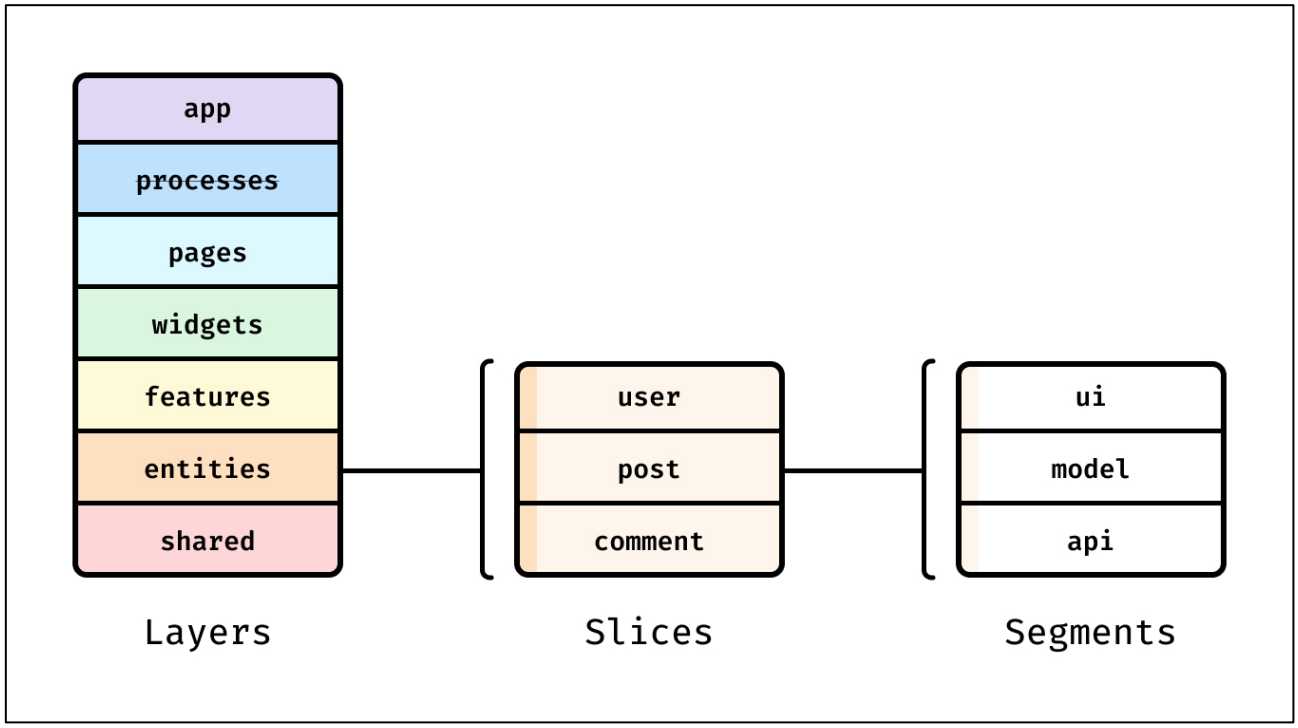


Рисунок 3.4 – Приклад використання методології FSD

Шари стандартизовані у всіх протктах і розташовані вертикально. Модулі на одному шарі можуть взаємодіяти лише з модулями, що розташовані на шарах строго нижче [20]. Наразі шарів сім (зверху вниз):

- app - налаштування, стилі та провайдери для всього застосунку;
- processes (процеси, застарілий шар) - складні сценарії, що покривають кілька сторінок. (наприклад, авторизація);
- pages (сторінки) - композиційний шар для складання повноцінних сторінок із сутностей, фіч і віджетів;
- widgets (віджети) - композиційний шар для з'єднання сутностей і фіч у самостійні блоки. (наприклад, IssuesList, UserProfile);
- features (фічі) - взаємодії з користувачем, дії, які несуть бізнес-цінність для користувача. (наприклад, SendComment, AddToCart, UsersSearch);
- entities (сутності) - бізнес-сутності. (наприклад, User, Product, Order);
- shared - перевикористовуваний код, що не має відношення до специфіки програми/бізнесу. (наприклад, UIKit, libs, API).

Слайси розділяють код за предметною областю. Вони групують логічно пов'язані модулі, що полегшує навігацію по кодовій базі. Слайси не можуть

використовувати інші слайси на тому ж шарі, що забезпечує високий рівень зв'язності (cohesion) при низькому рівні зачеплення (coupling).

Натомість кожен слайс складається із сегментів. Це маленькі модулі, головне завдання яких - розділити код всередині слайса за технічним призначенням. Найпоширеніші сегменти - ui, model (store, actions), api і lib (utils/hooks).

Таким чином, в нашій предметній області можна виділити наступні сутності: app, review, tag, ticket, alert. Кожна сутність має таблиці в БД та відповідні API методи для взаємодії із ними. PostgREST - це автономний веб-сервер, який перетворює базу даних PostgreSQL безпосередньо на RESTful API. Структурні обмеження та дозволи в базі даних визначають кінцеві точки та операції API.

Використання PostgREST є альтернативою ручному CRUD-програмуванню. Користувацькі сервери API страждають від проблем. Написання бізнес-логіки часто дублює, ігнорує або ускладнює структуру бази даних. Об'єктно-реляційне відображення є негерметичною абстракцією, що призводить до повільного імперативного коду. Філософія PostgREST встановлює єдине декларативне джерело істини: самі дані.

Для ініціалізації Nest.js додатку також скористаємось відповідним CLI (рис. 3.5). Nest CLI – це інструмент інтерфейсу командного рядка, який допомагає ініціалізувати, розробляти та підтримувати Nest-додатки. Він допомагає у багатьох аспектах, включаючи створення проекту, обслуговування його в режимі розробки, а також збірку та пакування програми для виробничого розповсюдження. Він втілює найкращі архітектурні патерни, що сприяють створенню добре структурованих додатків [21].

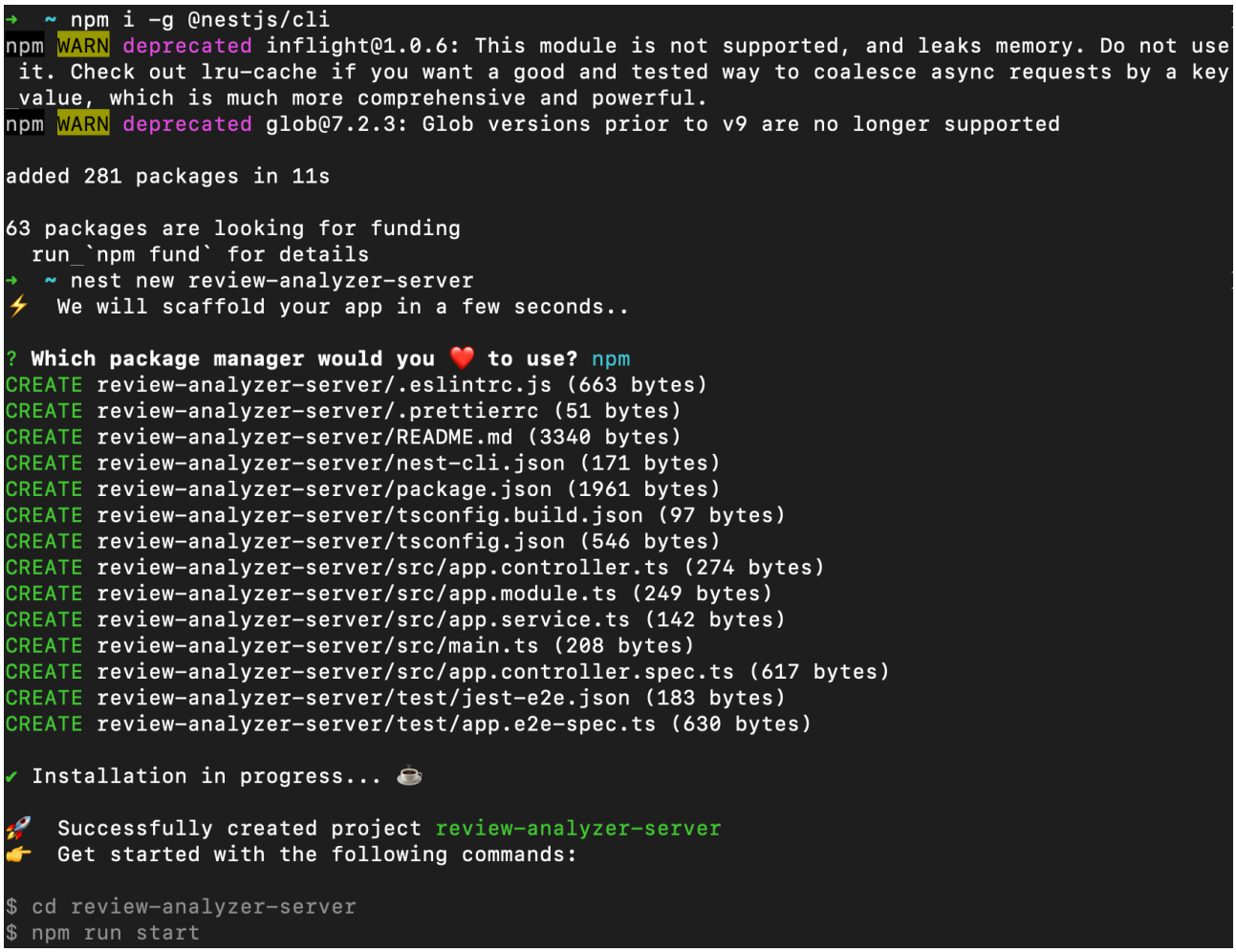


Рисунок 3.5 – Ініціалізація Nest.js додатку за допомогою CLI

Модуль "scraper" відповідає за збір даних, а саме відгуків користувачів з Google Play та App Store. Класичний веб-скрапінг, або парсинг, - це процес екстракції даних із веб-сайтів. Він включає наступні етапи:

1. веб-парсер відправляє HTTP-запит до специфічного URL;
2. сервер відповідає, відправляючи HTML-код веб-сторінки;
3. відбувається розбір HTML-коду, тобто парсер витягує потрібні дані з HTML-структури сторінки, використовуючи різні теги;
4. витягнуті дані зберігаються у потрібному форматі для подальшого аналізу.

Цей процес може повторюватися для багатьох веб-сторінок або веб-сайтів для збору великої кількості даних. Деякі веб-скрапери можуть також навігуватися між сторінками, переходячи за гіперпосиланнями.

Важливо зауважити, що деякі веб-сайти мають захист від веб-скрапінгу, наприклад, за допомогою файлів robots.txt, що обмежують доступ скраперів до певних частин сайту, або використовуючи спеціальні техніки, які роблять важким витягування даних.

Завдяки використанню таких Node.js бібліотек, як google-play-scraper та app-store-scraper цей процес значно спрощується, адже не потрібно писати власний web-scrapper чи використовувати публічний API, який надає кожен маркетплейс, адже в такому випадку потрібен приватний ключ доступу до конкретного додатку. Використовуючи open-source скрапери не можливо отримати всі мета-данні, які зберігаються на серверах маркетплейсу, лише відгуки, що є у відкритому доступі, однак навіть цього цілком достатньо для проведення якісного аналізу. На рисунку 3.6 наведено приклад імплементації методу збору відгуків на App Store.

```
13  async scrapReviews({ appStoreUrl }: ScrapAppDto) {
14    const reviews = await this.scrapAppStoreReviews(appStoreUrl);
15
16    for (const { text, updated, score } of reviews) {
17      await this.reviewsService.create({
18        source: Source.APP_STORE,
19        text,
20        score,
21        timestamp: new Date(updated),
22      });
23    }
24  }
25
26  private async scrapAppStoreReviews(url: string) {
27    return await store.reviews({
28      id: this.extractId(url, /id(\d+)/),
29      country: 'ua',
30    });
31  }
```

Рисунок 3.6 – Приклад методу для збору відгуків на App Store

Таким чином можна отримати результат у вигляді масиву даних з вмістом, приклад якого зображено на рисунку 3.7.

```
[ { id: '1472864600',
    userName: 'Linda D. Lopez',
    userUrl: 'https://itunes.apple.com/us/reviews/id324568166',
    version: '1.80.1',
    score: 5,
    title: 'Great way to pass time or unwind',
    text: 'I was a fan of Bejeweled many moons ago...',
    updated: '2021-07-26T18:26:24-07:00',
    url: 'https://itunes.apple.com/us/review?id=553834731&type=Purple%20Software' },,
  { id: '1472864708',
    userName: 'Jennamaxkidd',
    userUrl: 'https://itunes.apple.com/us/reviews/id223990784',
    version: '1.80.1',
    score: 1,
    title: 'Help! THE PROBLEM IS NOT FIXED!',
    text: 'STILL HAVING THE SAME ISSUE.  It\'s happening again...',
    updated: '2021-07-26T18:04:41-07:00',
    url: 'https://itunes.apple.com/us/review?id=553834731&type=Purple%20Software' },
  (...)
]
```


Рисунок 3.7 – Приклад зібраних даних з App Store

Отримані результати можна зберегти в базу даних для подальшого аналізу. Також, для зручності перегляду зібраних відгуків розроблено окрему сторінку (рис. 3.8).

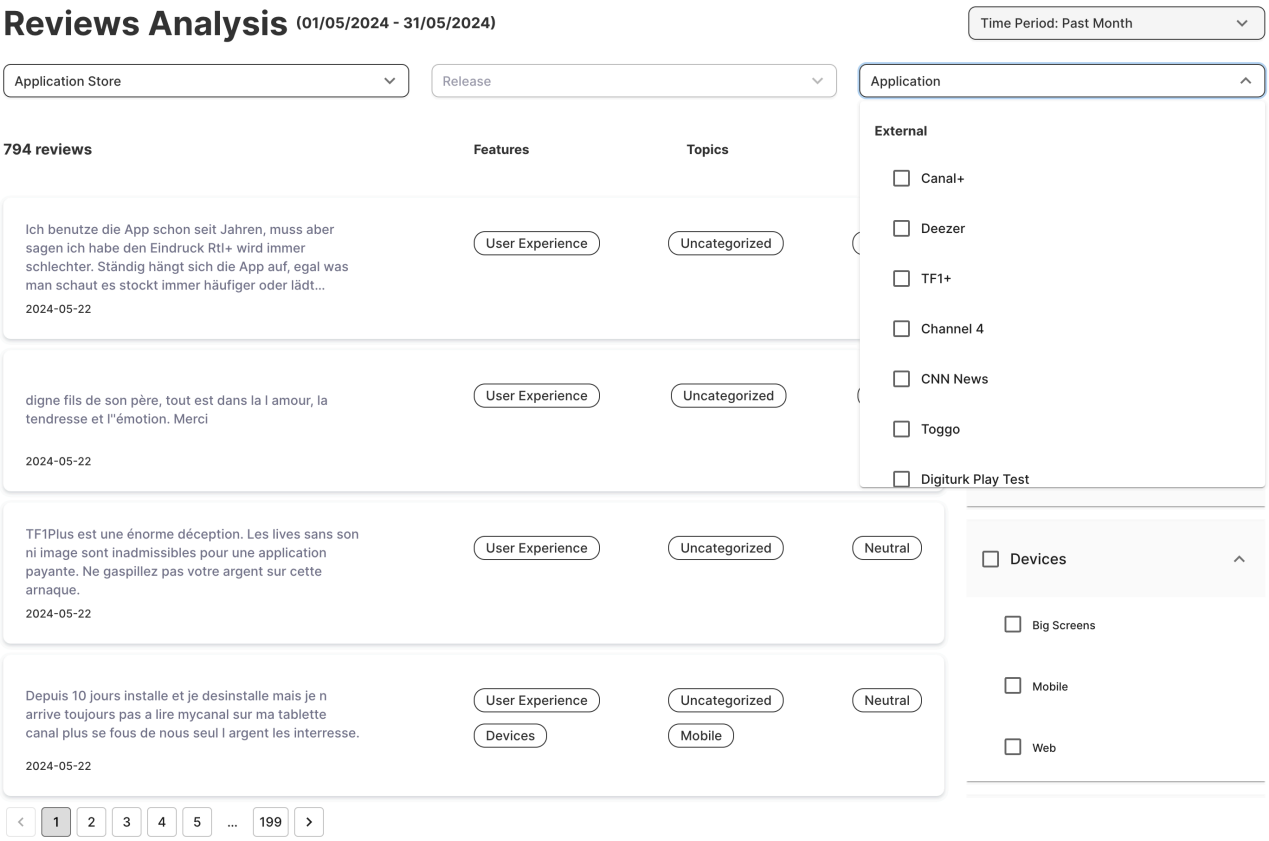


Рисунок 3.8 – Сторінка перегляду зібраних відгуків та їх фільтрації

Деталі кожного відгуку можна переглянути натиснувши на нього, тут же й можна згенерувати відповідь та прочитати переклад на англійську мову (рис. 3.9).

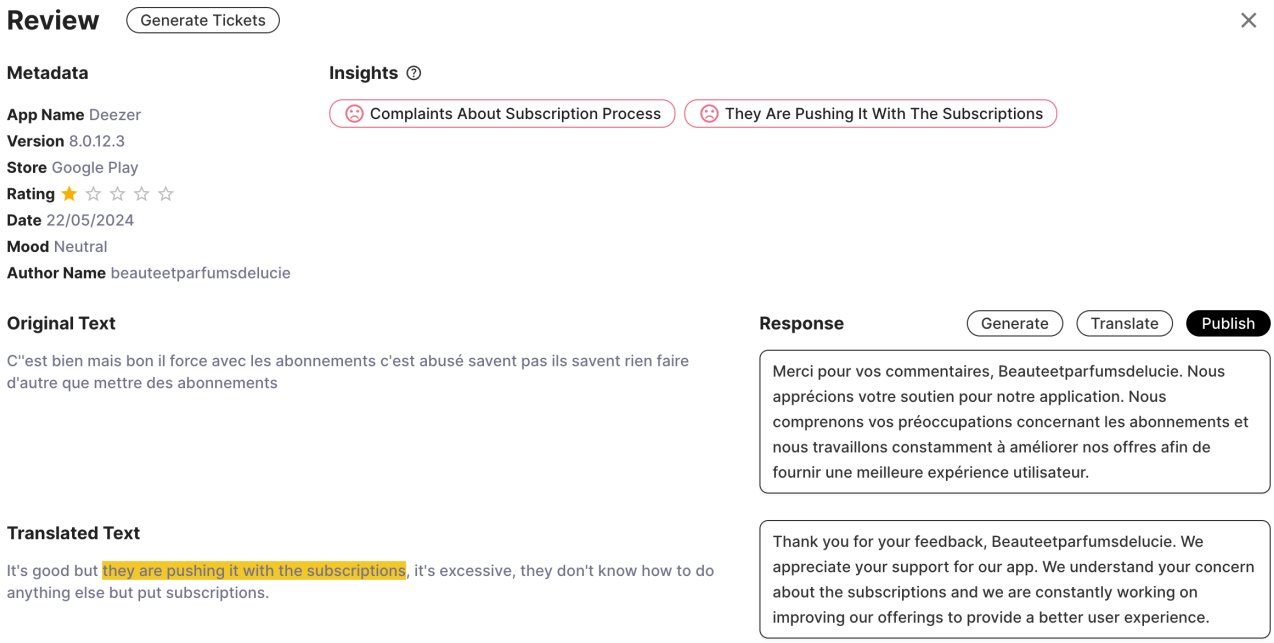


Рисунок 3.9 – Перегляд деталей на відгук та згенерована відповідь

Для генерації відповіді достатньо невеличкого запиту до LLM з використанням LangChain методів. На рисунку 3.10 зображеного приклад методу для генерації відповіді.

```
54    public async generateResponse(id: string) {
55      const review = await this.axiosClient
56        .get(`/review?id=eq.${id}`).then(item => item.data[0]);
57
58      if (review) {
59        const model = chat({ temperature: 0.3 });
60        const promptTemplate = new PromptTemplate({
61          template: `
62          You are a app owner, your goal is to answer for user review as app store response
63          title: {title}
64          review: {review}
65          answer should be for user name: {author_name}
66          use details
67          miss any mentions of owner
68          At most 3 sentences
69
70          Answer:
71          `,
72          inputVariables: ["review", "title", "author_name"]
73        })
74
75        const chain = promptTemplate.pipe(model);
76        const result = await chain.invoke({
77          author_name: (review.author_name || ""),
78          review: (review.text_translated || ''),
79          title: (review.title_translated || ''),
80        });
81
82        return result.content.toString()
83      }
84
85      return ""
86    }
```

Рисунок 3.10 – Метод для генерації відповіді на відгук

Однією з найцінніших функцій є перегляд відгуків, які містять негативні згадування щодо конкретної функціональності в додатку. Саме завдяки такому аналізу можна чітко визначити що найбільше турбує користувачів та відслідкувати можливий взаємозв'язок в залежності від обраного періоду та з огляду на джерело даних. Можливо, після дати виходу оновлення на для IOS, загальна кількість відгуків з негативним настроєм на App Store різко збільшилась. На рисунку 3.11 зображено сторінку, яка чітко це демонструє.

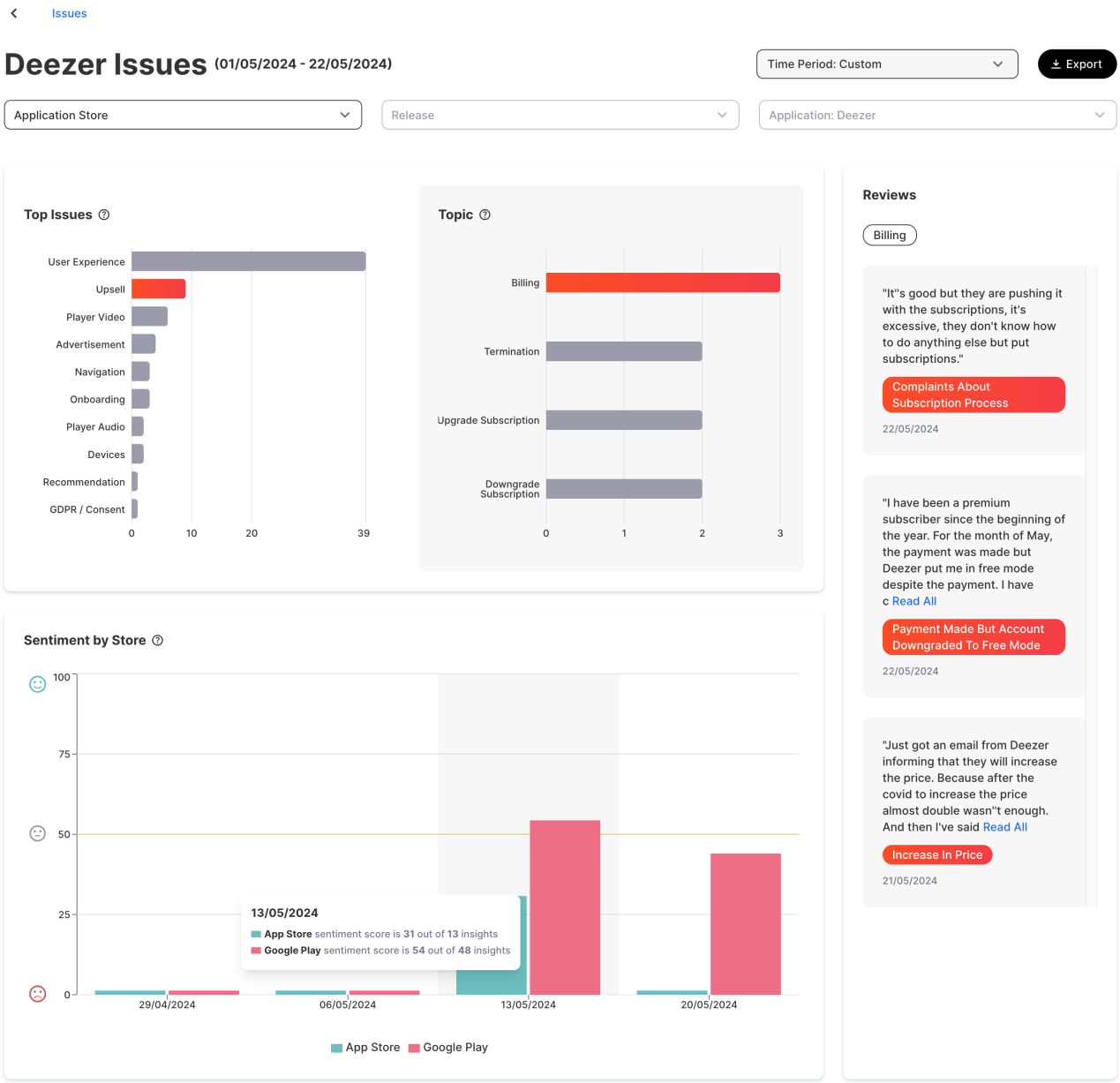


Рисунок 3.11 – Сторінка перегляду відгуків з негативними згадуваннями

Щоб провести подібний аналіз потрібно виокремити ключові фрази в тексті та виконати генерацію відповідних метаданих. Для цього в LangChain існує метод MetadataTagger. Він використовує конфігурований ланцюжок на основі OpenAI Functions та генерує метадані з кожного наданого документа відповідно до заданої схеми [1]. На рисунку 3.12 зображено приклад, як може виглядати імплементація методу для аналізу текстових відгуків.

```
  async tag(text: string) {
    const zodSchema = z.object({
      mood: z.enum(['delighted', 'happy', 'sad', 'angry', 'neutral']),
      insights: z.array(
        z.object({
          phrase: z.string(),
          tone: z.enum(['positive', 'neutral', 'negative']),
          category: z
            .string()
            .describe('propose an app feature depends on a phrase'),
        }),
      ),
    });

    const metadataTagger = createMetadataTaggerFromZod(zodSchema, {
      llm: this.chat({ temperature: 0.15 }),
      prompt: PromptTemplate.fromTemplate(`
        You received a feedback for a streaming application.

        Do the following:
        1. identify unique by meaning phrase
        2. identify tone of each phrase
        3. identify general mood of a reviewer

        Remove: simple communication phrases; just phrase
        Output: JSON format only

        Review: {input}
      `),
    });

    try {
      const taggedDocument = await metadataTagger._transformDocument(
        new Document({ pageContent: text }),
      );

      return taggedDocument.metadata as { mood: Mood; insights: Insight[] };
    } catch (error) {
      console.error('Something went wrong during tagging', error);
    }
  }
}
```

Рисунок 3.12 – Приклад аналізу відгуків з використанням LangChain

Коли дані представлені у вигляді графіків, може бути важко швидко зрозуміти головні тенденції або зробити порівняння. Текстове резюме подає інформацію в спрощеній формі, яка допомагає у перевірці інтерпретацій, отриманих з графіків. Резюме може надати інформацію, якої не видно на графіку,

таку як середнє значення, медіана, стандартне відхилення та інші статистичні показники. Особливо при роботі з великими наборами даних може бути важко візуалізувати всі деталі на одному графіку. Резюме допомагає усунути людські помилки, пов'язані з інтерпретацією візуальних даних. На рисунку 3.13 зображено приклад згенерованого резюме.

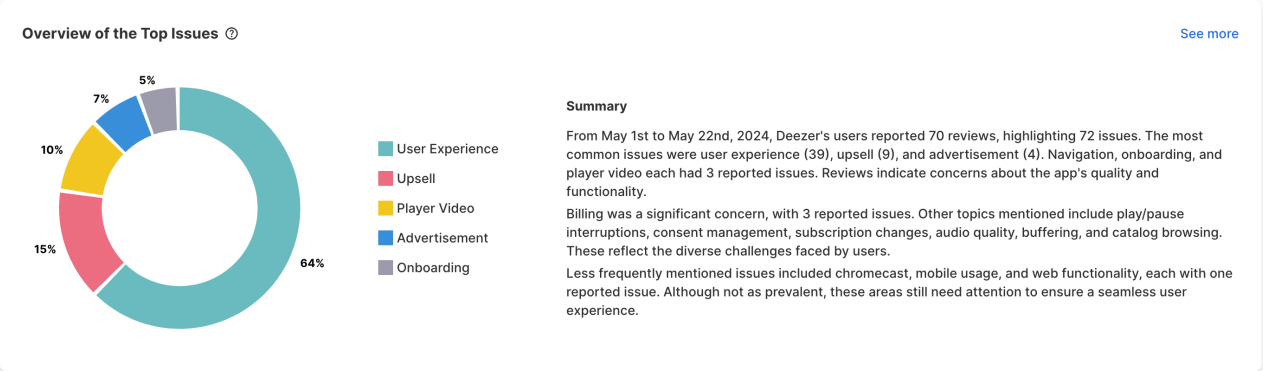
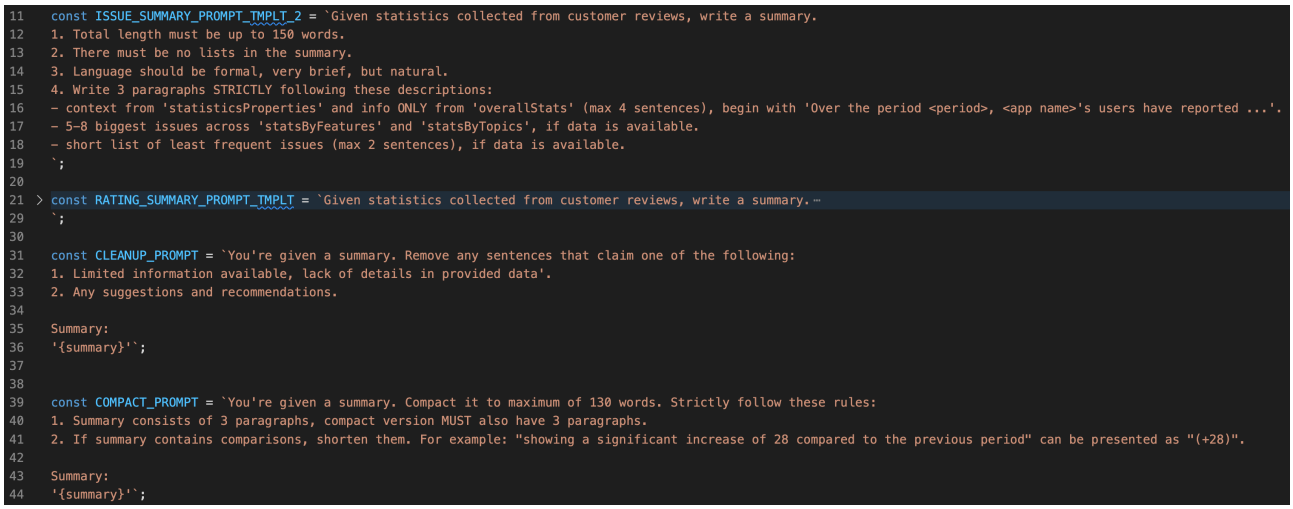


Рисунок 3.13 – Приклад текстового резюме

Для коректної генерації резюме за допомогою LLM від GPT, потрібно зробити наступне:

– сформувати початковий запит максимально виразно і чітко;
– обрати правильну модель, оскільки великі моделі зазвичай генерують більш точні та когерентні текстові відповіді, але вони також вимагають більше обчислювальної потужності та часу для генерації відповідей;
– налаштувати такі параметри моделі, як температуру та максимально допустиму довжину.

На рисунку 3.14 зображено приклади запитів для генерації резюме.



```
const ISSUE_SUMMARY_PROMPT_TMPLT_2 = `Given statistics collected from customer reviews, write a summary.
1. Total length must be up to 150 words.
2. There must be no lists in the summary.
3. Language should be formal, very brief, but natural.
4. Write 3 paragraphs STRICTLY following these descriptions:
- context from 'statisticsProperties' and info ONLY from 'overallStats' (max 4 sentences), begin with 'Over the period <period>, <app name>'s users have reported ...'.
- 5-8 biggest issues across 'statsByFeatures' and 'statsByTopics', if data is available.
- short list of least frequent issues (max 2 sentences), if data is available.
`;

const RATING_SUMMARY_PROMPT_TMPLT = `Given statistics collected from customer reviews, write a summary.…
`;

const CLEANUP_PROMPT = `You're given a summary. Remove any sentences that claim one of the following:
1. Limited information available, lack of details in provided data'.
2. Any suggestions and recommendations.

Summary:
'{summary}'`;


const COMPACT_PROMPT = `You're given a summary. Compact it to maximum of 130 words. Strictly follow these rules:
1. Summary consists of 3 paragraphs, compact version MUST also have 3 paragraphs.
2. If summary contains comparisons, shorten them. For example: "showing a significant increase of 28 compared to the previous period" can be presented as "(+28)".

Summary:
'{summary}'`;
```

Рисунок 3.14 – Приклади запитів для генерації резюме

На рисунку 3.15 наведено приклад використання цих запитів та імплементацію відповідних методів з використанням LangChain API.

```
export class SummaryGenerator {
  private summaryChain: StuffDocumentsChain;
  private cleanupChain: Runnable;
  private compactChain: Runnable;

  constructor(summaryModel: ChatOpenAI, compactModel: ChatOpenAI) {
    this.summaryChain = loadQAStuffChain(summaryModel);
    this.cleanupChain = PromptTemplate.fromTemplate(CLEANUP_PROMPT).pipe(summaryModel);
    this.compactChain = PromptTemplate.fromTemplate(COMPACT_PROMPT).pipe(compactModel);
  }

  async generateIssueSummary(stats: IssueSummaryStats): Promise<string> {
    print(
      `Generating issue summary based on statistics:
      ${JSON.stringify(stats, null, 2)}`,
    );

    const doc = new LCDocument({
      pageContent: JSON.stringify(stats),
      metadata: {},
    });

    const summaryResponse = await this.summaryChain.call({
      input_documents: [doc],
      question:  this.getSummaryPrompt(SummaryType.ISSUE, stats.statisticsProperties),
    });

    print(`Full issue summary:  ${summaryResponse.text}`);
    const filteredSummary = await this.cleanSummaryUp(summaryResponse.text);
    return await this.compactSummary(filteredSummary);
  }

  async generateRatingSummary(stats: RatingSummaryStats): Promise<string> {…
  }

  private async cleanSummaryUp(longSummary: string): Promise<string> {
    const cleanResponse = await this.cleanupChain.invoke({
      summary: longSummary,
    });
    print(`\nClean summary: \n ${cleanResponse.content}`);

    if (typeof cleanResponse.content === "string") {
      return cleanResponse.content;
    }
    throw new Error(
      `Clean response isn't string: ${JSON.stringify(cleanResponse.content)}`,
    );
  }

  private async compactSummary(longSummary: string): Promise<string> {
    const compactResponse = await this.compactChain.invoke({
      summary: longSummary,
    });
    print(`\nCompact summary:\n ${compactResponse.content}`);

    if (typeof compactResponse.content === "string") {
      return compactResponse.content;
    }
    throw new Error(
      `Compact response isn't string: ${JSON.stringify(compactResponse.content)}`,
    );
  }

  private getSummaryPrompt(summaryType: SummaryType, statsContext: StatsContext) {
    let prompt = summaryType == SummaryType.ISSUE ? ISSUE_SUMMARY_PROMPT_TMPLT_2 : RATING_SUMMARY_PROMPT_TMPLT;
    if (statsContext.application === undefined) {
      prompt = prompt.replace(/<app name>'s /g, '');
    }
    return prompt;
  }
}
```

Рисунок 3.15 – Методи для генерації резюме

Наявність системи сповіщень в інтелектуальній веб-системі аналізу відгуків допомагає вчасно реагувати на важливі аспекти, ідентифіковані в процесі аналізу відгуків.

Сповіщення можуть бути налаштовані для відправки при виявленні конкретних подій або умов, таких як негативні відгуки користувачів, згадування певних проблем або помилок у програмному забезпеченні, або виявлення важливих пропозицій щодо покращення.

Це дозволяє команді розробників оперативно реагувати на виявлені проблеми і питання користувачів, забезпечуючи високий рівень обслуговування клієнтів та постійне вдосконалення продукту.

Таким чином, система сповіщень допомагає покращувати відносини з користувачами та підтримувати високу якість програмного забезпечення. Гнучкі налаштування сповіщень допоможуть у виявленні помилки або проблем з процесом користування.

Для цього користувач повинен заповнити відповідні правила (рис. 3.16).

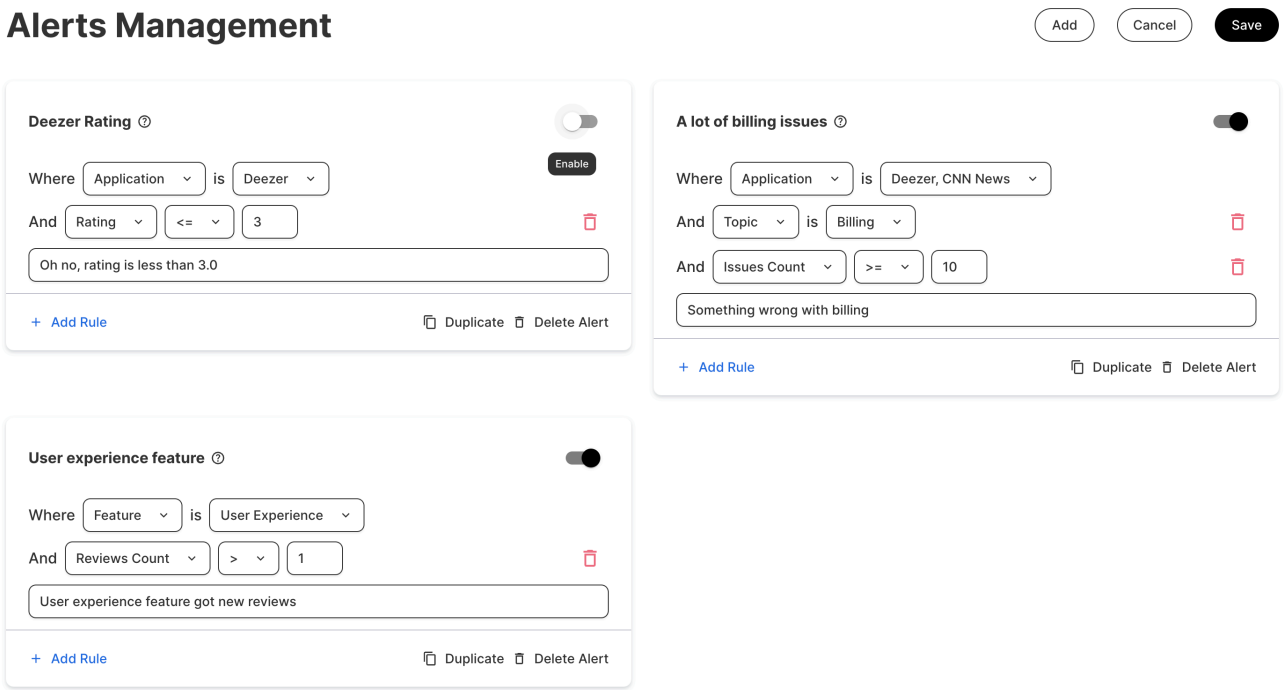


Рисунок 3.16 – Налаштування сповіщень

Після чого в базі даних буде збережено конфігурації, приклад яких зображено на рисунку 3.17.

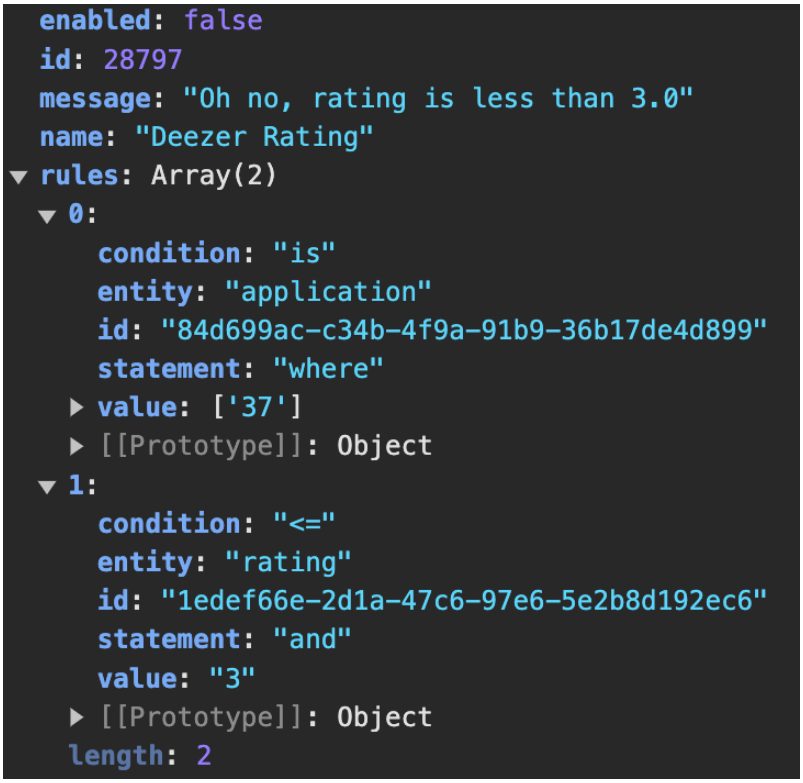


Рисунок 3.17 – Приклад об’єкту з конфігурацією сповіщень

Щоденно сервер автоматично перевіряє усі нові аналізи відгуків на наявність збігу з конфігурацією збережених сповіщень. Якщо виявлено збіг – користувач отримує сповіщення у відповідному Slack каналі. Програмну реалізацію вищеописаної логікі представлено на рисунку 3.18.

```
    rules.map(item => {
        if (item.entity === "sentiment" || item.entity === "rating" || item.entity === "reviews"){
            includesCountableEntity++;
        }else {
            uncountableEntities.push(item.entity)
        }

        if (item.entity === "sentiment") {
            select.push(" avg(insight.tone) as avg_tone");
        } else if (item.entity === "rating") {
            select.push(" avg(review.rating) as avg_rating");
        } else if (item.entity === "reviews") {
            select.push(" count(review.id) as review_count");
        } else if (item.entity === "application") {
            select.push(" review.app_id as app_id")
            group = "app_id";
        } else if (item.entity === 'topic' || item.entity === 'feature') {
            select.push(" insight.topic_id as topic_id")
            group = "topic_id";
        }
    });

    if (includesCountableEntity === 0 && uncountableEntities.length > 0){
        select = ['count(*)']
    }

    repo = repo.select(select.join(","));
    if (group !== '') {
        repo = repo.groupBy(group);
    }

    for (const rule of rules) {
        if (rule.entity == 'topic') {
            // @ts-ignore
            repo = repo.andWhere(`insight.topic_id ${rule.condition == 'is' ? 'in' : 'not in'} (:...topicId)`, { topicId: rule.value })
        } else if (rule.entity == "application") {
            // @ts-ignore
            repo = repo.andWhere(`app_id ${rule.condition == 'is' ? 'in' : 'not in'} (:...appId)`, { appId: rule.value})
        } else if (rule.entity == "feature") {
            // @ts-ignore
            repo = repo.andWhere(`insight.topic_id  ${rule.condition == 'is' ? 'in' : 'not in'} (:...topics)`, { topics:  rule.value })
        }
    }

    try {
        const result = await repo.execute();

        const alerts: string[] = [];

        if (result.length === 1) {
            const { avg_tone, avg_rating, review_count } = result[0]
            for (const rule of rules) {
                console.log(rule)
                print(result)
                if (rule.entity == "sentiment") {
                    if (updateCondition(avg_tone, rule.condition, Number(rule.value))) {
                        alerts.push("sentiment")
                    }
                } else if (rule.entity == "rating") {
                    if (updateCondition(avg_rating, rule.condition, Number(rule.value))) {
                        alerts.push("rating")
                    }
                } else if (rule.entity == "reviews") {
                    if (updateCondition(review_count, rule.condition, Number(rule.value))) {
                        alerts.push("reviews")
                    }
                }
            }
        }
        print(`${alerts}`)

        if (alerts.length > 0) {
            sendSlackNotification(alert.name, alert.message, alerts)
```

Рисунок 3.18 – Опрацювання сповіщень

На першому етапі обробки даних, для кожного відгуку, було виявлено ключові фрази (insights) та виконано їх тегування відповідно до заданої схеми (рис. 3.19).

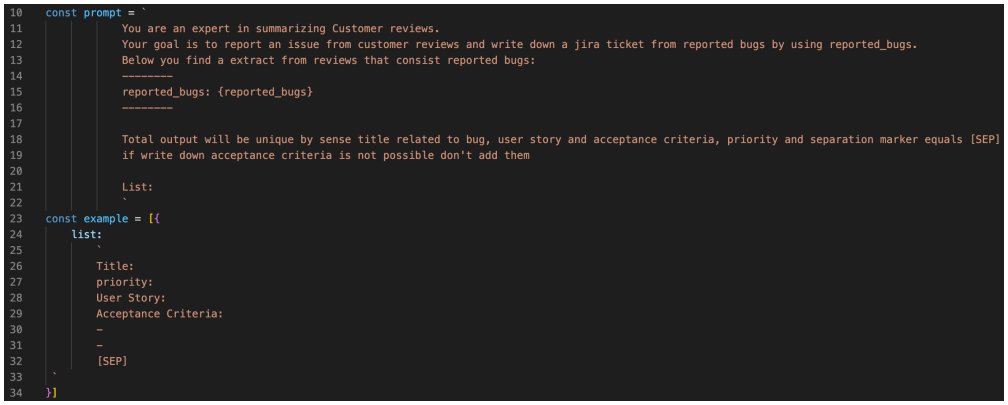
```
const zodSchema = z.object({
  mood: z.enum(["delighted", "happy", "sad", "angry", "neutral"]),
  insights: z
    .array(
      z.object({
        thought: z.string(),
        tone: z.enum(["positive", "neutral", "negative"]),
        usage: z.enum([
          "specific bug",
          "current app feature",
          "ads",
          "update",
          "content",
          "complaint",
          "mention about device",
          "add new",
          "just phrase",
          "compliment"
        ]),
        topic: z.string(),
        issueScore: z.number()
      })
    )
});
```

Рисунок 3.19 – Схема тегування відгука

Відгуки, які містили ключові фрази (insights) з типом застосування (usage) рівним “add new”, було використано для генерації потенційних задач для розробки функціональності (рис. 3.20), а ті що помічені як “specific bug” – потенційних завдань для виправлення (рис. 3.21).

```
const prompt = `
You are an expert in summarizing Customer reviews.
Your goal is to create a feature request from customer reviews and write down a jira ticket from feature requests.
Below you find a extract from reviews that consist of feature request:
--------
request: {review}
--------

Total output will be list of title related to feature request,  user story and acceptance criteria
if write down acceptance criteria or user story is not possible don't add them
Return only one ticket

List:
`

const example = [{
  list:
    `
      priority:
      Title:
      User Story: As a user,
      Acceptance Criteria:
      -
      -
`
}]

export default async (
  reviews: { reviewId: string, description: string, topic_id: string, app_id: string, version: string, platform:string }[],
  topic: { [x: string]: string },
  app: { [x: string]: string },
) => {
  print("Generating feature ticket started");
  const model = chat({ temperature: 0.6 });
  const output: Ticket[] = []
  const currentDate = new Date().toISOString();
  const ticketApp: {[x:string]: string} = {};

  for (const review of reviews) {
    print(`Generating feature ticket for review started reviewId: ${review.reviewId}`);
    const promptTemplate = new PromptTemplate({
      template: `{list}`,
      inputVariables: ["list"],
    });
    const fewShotPrompt = new FewShotPromptTemplate({
      examplePrompt: promptTemplate,
      examples: example,
      prefix: prompt,
      inputVariables: ["review"],
    });

    const chain = fewShotPrompt.pipe(model);
    const result = await chain.invoke({ review: review.description });

    const ticket = result.content.toString()

    if (ticket) {
      const priorityReference = ticket.match(/[Pp]riority: \w+/g);
      const reference = ticket.match(/User Story: [\s\w,."' +]+\n/g) || " ";
      const title = ticket.match(/[Tt]itle: [\s\w,."' +]+\n/g)

      output.push({
        priority: (priorityReference || "").toString().replace(/[Pp]riority: /g, ''),
        type: "feature",
        version: `${app[review.app_id]}: ${review.version}`,
        label: topic[review.topic_id],
        title: (title || "").toString(),
        description: ticket.replace(/List:/g, '').replace(/[Pp]riority: \w+\n/g, '').replace(/[Tt]itle: [\s\w,."' +]+\n/g, '').trim(),
        reference: reference[0].toString(),
        timestamp: currentDate,
        app_id: [review.app_id],
        platform: review.platform,
      })
    }
```

Рисунок 3.20 – Код генерації потенційних задач



```
const prompt = `
            You are an expert in summarizing Customer reviews.
            Your goal is to report an issue from customer reviews and write down a jira ticket from reported bugs by using reported_bugs.
            Below you find a extract from reviews that consist reported bugs:
            --------
            reported_bugs: {reported_bugs}
            --------

            Total output will be unique by sense title related to bug, user story and acceptance criteria, priority and separation marker equals [SEP]
            if write down acceptance criteria is not possible don't add them

            List:
            `
const example = [{
    list:
      `
       Title:
       priority:
       User Story:
       Acceptance Criteria:
       -
       -
       [SEP]
`
}]
```

Рисунок 3.21 – Запит для генерації завдань на виправлення

На рисунку 3.22 представлено сторінку для перегляду потенційних задач.

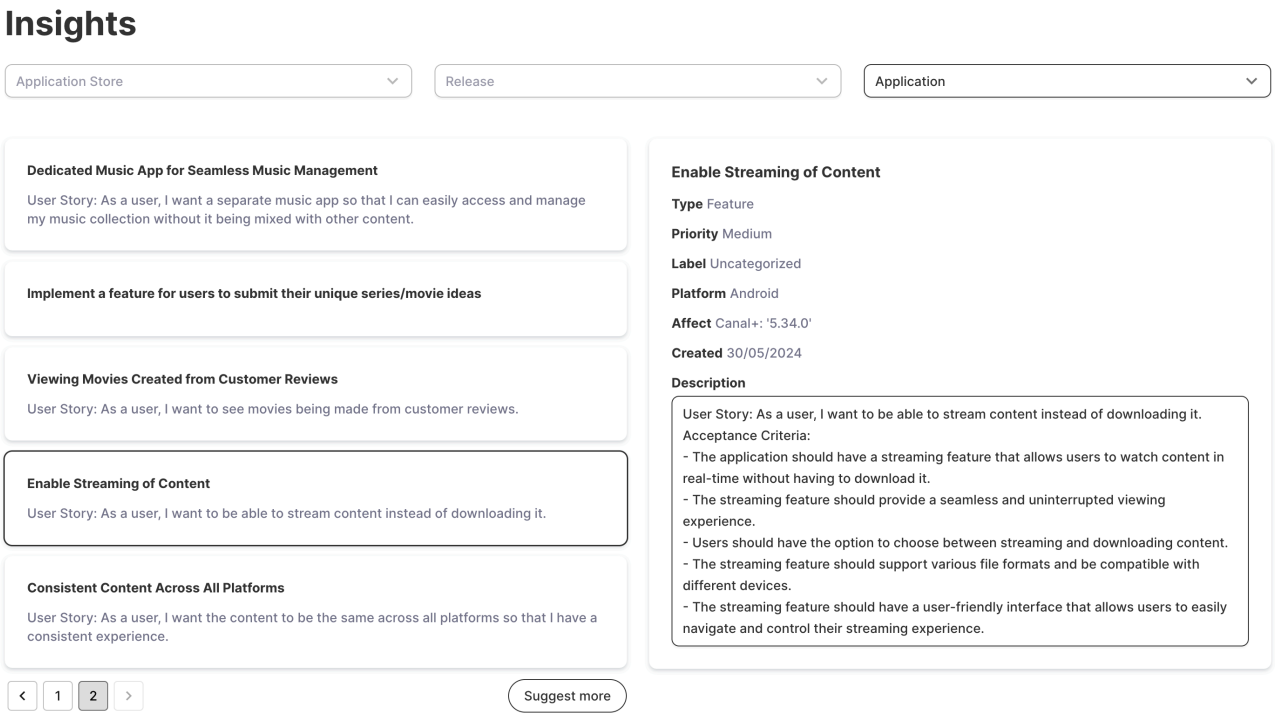


Рисунок 3.22 – Сторінка перегляду потенційних задач на основі відгуків

Отже, розроблена інтелектуальна веб-система використовує технології NLP від GPT-4 для:

– розуміння текстових відгуків, включаючи семантику, контекст і тон використання мови;
– класифікації відгуків по темах, що дозволяє групувати і аналізувати їх за спільними категоріями;
– неупередженого аналізу емоційного настрою відгуків, що дозволяє зрозуміти загальне ставлення користувачів до програмного застосунку;
– генерації коротких резюме, дозволяючи користувачам швидко ознайомитися з результатами аналізу відгуків;
– генерації відповіді на відгуки відповідною мовою;
– автоматичної ідентифікації помилок та пропозицій щодо покращення програмного застосунку.

Також, оскільки процес збору та аналізу відбувається щодня в автоматичному режимі, інтелектуальна веб-система може відправляти сповіщення в Slack, відповідно до заданих правил та сценаріїв, що в свою чергу допомагає вчасно помітити важливі зміни в настрої чи оцінках користувачів та вжити відповідних заходів, щоб попередити різке падіння кількості користувачів.

3.3 Висновки

В даному розділі було розроблено архітектуру інтелектуальної веб-системи аналізу відгуків на програмні застосунки. Наведено Use-Case UML діаграму, здійснено програмну реалізацію веб-додатку відповідно до FSD методології. Наведено приклади імплементації методів для збору й аналізу текстових відгуків. Розглянуто запити для генерації резюме, відповіді на відгук та потенційних задач для команди розробників. Продемонстровано інтерфейс користувача для відображення опрацьованих та згенерованих даних. Описано функціональні можливості інтелектуальної веб-системи.

ВИСНОВКИ

Дипломна робота освітлює важливість використання відгуків користувачів для оптимізації продуктів в контексті зростаючого ринку програмних застосунків. В рамках роботи зроблено акцент на необхідності розробки інтелектуальної веб-системи для ефективного аналізу цих відгуків, що допомагає вивчити досвід користувачів і вдосконалити програмне забезпечення.

Розробка системи включає в себе ризик помилки в алгоритмах обробки природної мови, які можуть вплинути на точність аналізу відгуків та призвести до збільшення витрат на розробку проєкту та зменшення його ефективності. З цього приводу, було запропоновано використати існуючу LLM. Цей підхід дозволить зменшити зусилля та витрати на розробку та тренування власної моделі, а також дозволить сфокусуватися на розробці ефективних алгоритмів та запитів для аналізу відгуків користувачів.

Для розробки веб-системи було обрано мову програмування TypeScript і фреймворки Next.js, Nest.js та LangChain з використанням GPT-4 – LLM від OpenAI.

Дипломна робота демонструє, що розроблена інтелектуальна веб-система значно покращує процес аналізу відгуків на програмні застосунки. Завдяки використанню технологій обробки природної мови (NLP) та GPT-4, система глибоко аналізує текст відгуків, розуміє їх семантику, контекст та емоційний тон. Це дозволяє точно класифікувати відгуки за темами для подальшого аналізу.

Система також автоматично визначає емоційний настрій відгуків, створює короткі резюме, генерує відповіді на відгуки, ідентифікує помилки та пропозиції щодо покращення програмного застосунку. Також система надсилає сповіщення у Slack згідно з налаштованих правил та сценаріїв.

У цілому, використання інтелектуальної веб-системи підвищує ефективність аналізу відгуків, дозволяючи швидше і точніше виявляти проблеми, розуміти настрої користувачів і вчасно реагувати на зміни.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Побідаш В. В., Крижановський Є. М. Інформаційна система аналізу відгуків на програмні застосунки. *Міжнародна науково-практична інтернет-конференція «Молодь в науці: дослідження, проблеми, перспективи»*. Вінниця, 2024. [Електронний ресурс]. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/21628

2. Angèle L.M. Cavaye. User participation in system development revisited. *Information & Management*. 1995. [Електронний ресурс]. DOI: https://doi.org/10.1016/0378-7206(94)00053-L

3. J Drew Procaccino, June M Verner, Scott P Overmyer, Marvin E Darter. Case study: factors for early prediction of software development success. *Information and Software Technology*. 2002. [Електронний ресурс]. DOI: https://doi.org/10.1016/S0950-5849(01)00217-8

4. Mark I. Hwang, Ron G. Thorn. The effect of user engagement on system success: A meta-analytical integration of research findings. *Information & Management*. 1999. [Електронний ресурс]. DOI: https://doi.org/10.1016/S0378-7206(98)00092-5

5. Daniela S. Cruzes, Tore Dybå. Research synthesis in software engineering: A tertiary study. *Information and Software Technology*. 2011. [Електронний ресурс]. DOI: https://doi.org/10.1016/j.infsof.2011.01.004

6. Maalej, W., Kurtanović, Z., Nabil, H. et al. On the automatic classification of app reviews. *Requirements Engineering*. 2016. [Електронний ресурс]. DOI: https://doi.org/10.1007/s00766-016-0251-9

7. Мокін В. Б., Дратований М. В. Наука про дані: машинне навчання та інтелектуальний аналіз даних. Вінниця : ВНТУ, 2024. 258 с.

8. Software Market Size, 2024. [Електронний ресурс]. URL: https://www.precedenceresearch.com/software-market

9. Business Software Market Size, 2024. [Електронний ресурс]. URL: https://www.mordorintelligence.com/industry-reports/global-business-software-market

10. UserVoice, 2024. [Електронний ресурс]. URL: https://www.uservoice.com/

11. ReviewTrackers, 2024. [Електронний ресурс]. URL: https://www.reviewtrackers.com/

12. Minimum Viable Product, 2020. [Електронний ресурс]. URL: https://www.techopedia.com/definition/27809/minimum-viable-product-mvp

13. M. Rahmouni, M. N. Lakhoua, State of the art of enterprise modeling. *International Conference on Logistics*. 2011. [Електронний ресурс]. DOI: https://doi.org/10.1109/LOGISTIQUA.2011.5939308

14. Sotnik S., Shakurova T., Lyashenko V. Development Features Web-Applications. *International Journal of Academic and Applied Research (IJAAR)*. 2023. [Електронний ресурс]. URL: https://openarchive.nure.ua/handle/document/21600

15. What is Full Stack, 2024. [Електронний ресурс]. URL: https://www.w3schools.com/whatis/whatis_fullstack.asp

16. Dan Vanderkam. Effective TypeScript: 83 Specific Ways to Improve Your TypeScript. O'Reilly Media; 2nd edition, 2024. 401 с.

17. Michele Riva. Real-World Next.js: Build scalable, high-performance, and modern web applications using Next.js, the React framework for production. Packt Publishing, 2022. 366 с.

18. Daniel Correa, Greg Lim. Practical Nest.js: Develop clean MVC web applications. Independently published, 2022. 190 с.

19. Oguzhan Topsakal, Tahir Cetin Akinci. Creating Large Language Model Applications Utilizing LangChain: A Primer on Developing LLM Apps Fast. *International Conference on Applied Engineering and Natural Sciences*. 2023. [Електронний ресурс]. DOI: https://doi.org/10.59287/icaens.1127

20. FSD Overview, 2024. [Електронний ресурс]. URL: https://feature-sliced.design/docs/get-started/overview

21. Nest CLI, 2024. [Електронний ресурс]. URL: https://docs.nestjs.com/cli/overview

Додаток А

(обов’язковий)

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
“____”_____ 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на бакалаврську дипломну роботу

ІНТЕЛЕКТУАЛЬНА ВЕБ-СИСТЕМА АНАЛІЗУ ВІДГУКІВ
НА ПРОГРАМНІ ЗАСТОСУНКИ

08-34.БДР.014.00.000 ТЗ

Керівник: к.т.н., доц.
_____ Євгеній КРИЖАНОВСЬКИЙ
«__» _____ 2024 р.
Розробив студент гр. 2ІСТ-20б
_____ Владислав ПОБІДАШ
«__» _____ 2024 р.

Вінниця 2024

1. Підстава для проведення робіт.

Підставою для виконання роботи є наказ №__ по ВНТУ від «__» _____2024р., та індивідуальне завдання на БДР, затверджене протоколом №__ засідання кафедри САІТ від «__» _____ 2024р.

2. Джерела розробки:

1) Мокін В. Б., Дратований М. В. Наука про дані: машинне навчання та інтелектуальний аналіз даних. Вінниця : ВНТУ, 2024. 258 с.

2) Oguzhan Topsakal, Tahir Cetin Akinci. Creating Large Language Model Applications Utilizing LangChain: A Primer on Developing LLM Apps Fast. *International Conference on Applied Engineering and Natural Sciences*. 2023. [Електронний ресурс]. DOI: https://doi.org/10.59287/icaens.1127

3. Мета і призначення роботи.

Метою дослідження є підвищення ефективності аналізу відгуків на програмні застосунки шляхом використання великих мовних моделей

4. Вихідні дані для проведення робіт:

Відгуки користувачів програмних застосунків, розміщених на Google Play та App Store.

5. Методи дослідження:

a) огляд наукової літератури;
b) навчання з перенесенням (transfer learning);
c) інженерія підказок (prompt engineering).

6. Етапи роботи і терміни їх виконання:

a) Аналіз проблематики відгуків на програмні застосунки _____ – _____
b) Вибір оптимальних інформаційних технологій _____ – _____
c) Розроблення інтелектуальної веб-системи аналізу відгуків на програмні застосунки _____ – _____
d) Оформлення матеріалів до захисту БДР _____ – _____

7. Очікувані результати та порядок реалізації

Отримання інтелектуальної веб-системи аналізу відгуків на програмні застосунки.

8. Вимоги до розробленої документації

Текстова та ілюстративна частини роботи оформлені у відповідності до вимог «Методичних вказівок до виконання бакалаврських кваліфікаційних робіт для студентів спеціальностей: 124 «Системний аналіз», 126 «Інформаційні системи та технології» (освітня програма «Прикладні інформаційні технології»)».

9. Порядок приймання роботи

Публічний захист «___» _____ 2024 р.
Початок розробки «___» _____ 2024 р.
Граничні терміни виконання БДР «___» _____ 2024 р.

Розробив студент групи 2ІСТ-20б _____ Владислав ПОБІДАШ

Додаток Б

(обов'язковий)

Протокол перевірки бакалаврської дипломної роботи на наявність текстових запозичень

Назва роботи: «Інтелектуальна веб-система аналізу відгуків на програмні застосунки»
Тип роботи: бакалаврська дипломна робота
Підрозділ: кафедра САІТ

Показники звіту подібності Unicheck

Оригінальність 96% Схожість 4%

Аналіз звіту подібності (відмітити потрібне)

• Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
○ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.
○ 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 _____ (підпис) Сергій ЖУКОВ

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

Автор роботи _____ (підпис) Владислав ПОБІДАШ

Керівник роботи _____ (підпис) Євгеній КРИЖАНОВСЬКИЙ

Додаток В

(довідниковий)

Фрагмент лістингу програми

Код для агрегації параметрів та отримання даних з API:

```
import { DefinedUseQueryResult } from "@tanstack/react-query";
import { type AxiosHeaderValue } from "axios";

import { apiInstance } from "@/shared/api/api-instance";
import { QueryParamKeys } from "@/shared/lib/use-query-params";

export enum TagSelector {
  feature = "...feature(feature_id:id,name,removed)",
  topic = "topic_id,name:topic",
}

export enum TimePeriod {
  day = "review_date",
  week = "review_week",
  month = "review_month",
  year = "review_year",
}

interface FetchParams {
  [QueryParamKeys.id]: string | number;
  [QueryParamKeys.app_id]: string;
  [QueryParamKeys.feature_id]: string;
  [QueryParamKeys.topic_id]: string;
  [QueryParamKeys.platform]: string;
  [QueryParamKeys.version]: string;
  [QueryParamKeys.from]: string;
  [QueryParamKeys.to]: string;
  tag: TagSelector;
```

```
  period: TimePeriod[];
  select: string;
}

export type FetchParamsType = Partial<FetchParams>;

export const queryFn = <T>(
  url: string,
  fetchParams: FetchParamsType = {},
  headers: {
    [key: string]: AxiosHeaderValue;
  } = {},
) => {
  const params = Object.entries(fetchParams)
    .filter(([key, value]) => value)
    .reduce(
      (updatedParams, [key, value]) => {
        switch (key) {
          case QueryParamKeys.id:
          case QueryParamKeys.app_id:
          case QueryParamKeys.feature_id:
          case QueryParamKeys.topic_id:
          case QueryParamKeys.platform:
          case QueryParamKeys.version:
            updatedParams[key] =
              (value as string).split(",").length > 1 ? `in.(${value})` : `eq.${value}`;
            break;
          case QueryParamKeys.from:
          case QueryParamKeys.to:
            updatedParams["review_date"] = [
              `gte.${fetchParams[QueryParamKeys.from]}`,
              `lte.${fetchParams[QueryParamKeys.to]}`,
            ];
            break;
          case "select":
```

```
        const tagSelector = fetchParams["tag"];
        const period = fetchParams["period"];

        const select = [value];

        if (tagSelector) select.unshift(tagSelector);
        if (period) select.unshift(period);

        updatedParams.select = select.join(",");
       default:
        break;
      }

      return updatedParams;
    },
    {} as FetchParamsType & { review_date: string[] },
  );

 return () =>
   apiInstance
    .get(url, {
      params,
      headers,
      paramsSerializer: {
        indexes: null, // by default: false
      },
    })
    .then(({ data }) => data as T);
};
```

Код скрапера для отримання даних з App Store:

```
// @ts-ignore
import store from 'app-store-scraper';

type Data = {
```

```
    [key: string]: string
}

type LocalizationSettings = {
    countries?: string[]
    langs?: string[]
}

type LocalizationList = { [key: string]: string }[]

export default class AppStoreScrapper {

    private _applicationId: string;
    private _localizationSettings: LocalizationSettings;
    private _formatParams: { [key: string]: string }
    private _dateLimit: Date | null
    private limitReached: boolean

    finalData: Data[] = []

    constructor() {
        this._applicationId = ''
        this._localizationSettings = {}
        this._formatParams = {}
        this._dateLimit = null
        this.limitReached = false
    }

    public set applicationId(appId: string) {
        this._applicationId = appId
    }

    public set localizationSettings(settings: LocalizationSettings) {
        this._localizationSettings = settings
    }
```

```
preparelocalsList(): LocalizationList {
   const result: LocalizationList = []
   this._localizationSettings.countries?.map(country => {
      result.push({
         country
      })
   })
   return result
}

formatData(entity: any): any {
   const recordDate = new Date(entity.updated)

   if (this._dateLimit && recordDate <= this._dateLimit) {
      this.limitReached = true
      return {}
   }
   return {
      external_id: entity.id,
      app_id: this._formatParams.app_id,
      title: entity.title,
      text: entity.text,
      platform: this._formatParams.platform,
      rating: entity.score,
      version: entity.version,
      author_name: entity.userName,
      timestamp: entity.updated,
   }
}

async getData(dataManager: any, params: any) {
   this.finalData = []
   this._formatParams = params
```

```
    this._dateLimit = await dataManager.getLastSavedDate(params.app_id, params.platform)

    Promise.all(
      this.preparelocalsList().map(
        locale => this.getDataByCountryAndLang(locale.country)
      )
    ).then(res => {
      dataManager.saveReviews(this.finalData, params)
    })
  }

  async getDataByCountryAndLang(country: string, page: number = 1) {
    const res = await store.reviews({
      id: this._applicationId,
      country,
      page
    })

    // @ts-ignore
    const preparedData = res.map(el => this.formatData(el));

    this.finalData.push(...preparedData)
    if (res.length >= 0 && page < 10 && !this.limitReached) {
      await this.getDataByCountryAndLang(country, page + 1)
    } else {
      return Promise.resolve(true)
    }
  }

  async getRating(dataManager: any, params: any) {
    const { app_id } = params;
    const app = await dataManager.getAppById(app_id)
    const prev_data = app.global_ratings ?? {}

    const data = await store.app({
```

```
        id: this._applicationId,
        country: this._localizationSettings.countries?.[0] || 'us'
      })

      const new_ratings = {
        ...prev_data,
        ios: {
          ratingsCount: data.reviews,
          rating: data.score
        }
      }
      dataManager.updateRatingByApp(app_id, new_ratings)
    }
}
```

Код аналізу з використанням LangChain:

```
import { HumanMessage } from "langchain/schema";
import z from "zod";
import { Document as LCDocument } from "langchain/document";
import { IReview } from "./types.js";
import { portion } from "../utils/portion.js";
import { createMetadataTaggerFromZod } from "langchain/document_transformers/openai_functions";
import { chat } from "../models/chat.js";
import { PromptTemplate } from "langchain/prompts";
import { DocumentInterface } from "@langchain/core/documents";
import { jsonWrite } from "../utils/fileReader.js";
import { print } from '../../../utils/logs.js';

export const phrasesEnum: [string, ...string[]] = [
  "specific bug",
  "current app feature",
  "ads",
  "update",
  "content",
```

```
  "complaint",
  "mention about device",
  "add new",
  "just phrase",
  "compliment"
]

export const taggingChainTemplate = `
  You received a feedback for tv streaming app from app google play. You need to create sentiment for phrase:
  1. According to usage parameter filter main thoughts about unique problems or unique benefits of the app
  2. Identify issue score, that should identify how mention relates to issue
  3. Provide polarity

  Remove: simple communication phrases; just phrase
  Output in JSON Format only
  Review:
  {input}
`;

export const translationPrompt = `"You are translator. Translate text into English. If text is already in English, return the same text. If there is not text, just your translation is "no input". If there is unrecognizable text, your translation is "hard to translate". Your answer is just a translation.\n\n"`

export async function translateReview(reviewText: string) {
  try {
    const result = await chat({ temperature: 0, useCallback: false, model: 'gpt-4' }).predictMessages([
      new HumanMessage(
        translationPrompt +
        reviewText
      ),
    ]);
    return result.content;
  } catch (error) {
    print(`error ${error}`);
    return "";
  }
```

```
}

export async function identifyProblems(reviews: IReview[], chunkSize: number) {
 const modifiedReviews = [...reviews];
 const problemsZodSchema = z.object({
  mood: z.enum(["delighted", "happy", "sad", "angry", "neutral"]),
  insights: z
   .array(
    z.object({
     thought: z.string(),
     tone: z.enum(["positive", "neutral", "negative"]),
     usage: z.enum(phrasesEnum),
     topic: z.string(),
     issueScore: z.number()
    })
   )
 });

 const metadataTagger = createMetadataTaggerFromZod(problemsZodSchema, {
  llm: chat({ temperature: 0.15, useCallback: false }),
  prompt: PromptTemplate.fromTemplate(taggingChainTemplate),
 });

 const chunks = portion(reviews, chunkSize);
 print(`chunks.length ${chunks.length}`);
 print(`chunks.size ${chunkSize}`);

 const promises = [];

 for await (const chunk of chunks) {
  const input_documents = chunk.map(
   (c) =>
    new LCDocument({
     pageContent: `
     Title: ${(c.title_translated?.toString()?.includes('ard to translate') ? "Uncategorized": c.title_translated) ||
"Uncategorized"};
```

```
      Review: ${(c.text_translated?.toString()?.includes('ard to translate') ? "Uncategorized": c.text_translated)||
"Uncategorized"};`,
       metadata: {
        external_id: c.external_id,
       },
      })
  );

  try {
   promises.push(metadataTagger.transformDocuments(
    input_documents
   ).then((resolved) => {print(`Chunk execution finished`); return resolved}));
  } catch (error) {
   jsonWrite(modifiedReviews, `local-data/llm/current-state-crashed.json`)
   print(`error ${error}`);
  }

}
try {
  const taggedDocuments = await Promise.allSettled(promises)

  const filteredTaggedDocument = taggedDocuments.flatMap(
   (item) => {
    if(item.status === 'fulfilled'){
     return item?.value;
    }
    print(`Can't tag the documents reason: ${item.reason}`)

   return []
   })

  processTaggedDocuments(filteredTaggedDocument, modifiedReviews);
 } catch (error) {
  print(`error ${error}`);
 }
 return modifiedReviews;
```

```
}

function processTaggedDocuments(
  taggedDocuments: DocumentInterface<Record<string, any>>[],
  modifiedReviews: IReview[]
) {
  taggedDocuments.forEach((td) => {
    const review = modifiedReviews.find(
      (mr) => mr.external_id === td.metadata.external_id
    );
    if (review) {
      review.insights = [];

      let mapper = (
        { thought,  tone, usage, issueScore, topic}:
        {
          thought: string;
          tone: string;
          usage: string;
          issueScore: number
          topic:string,
        }) => {
          return {
            description: thought,
            tone,
            usage: usage,
            issueScore,
            topic
          }
        };
      review.insights.push(...td.metadata.insights.map(mapper));
      review.mood = td.metadata.mood;
    }
  });
}
```

Додаток Г

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНТЕЛЕКТУАЛЬНА ВЕБ-СИСТЕМА АНАЛІЗУ ВІДГУКІВ
НА ПРОГРАМНІ ЗАСТОСУНКИ

Нормоконтроль: к.т.н., доцент

_____ Сергій ЖУКОВ

«___» _____ 2024 р.

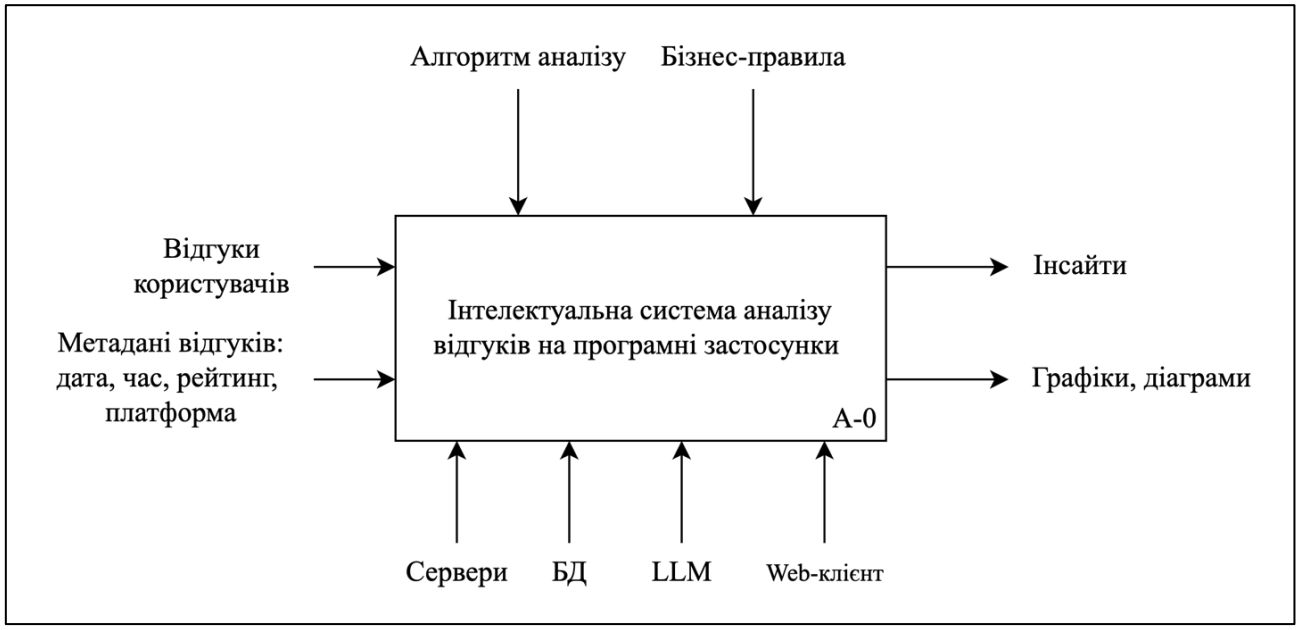


Рисунок Г.1 – Контекстна діаграма A-0

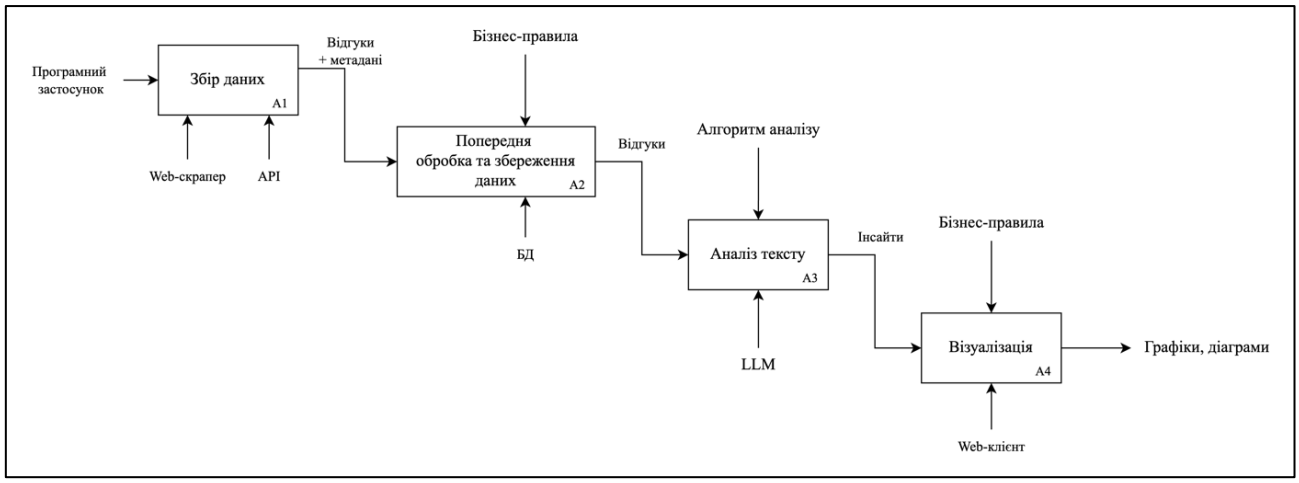


Рисунок Г.2 – Діаграма модулів системи

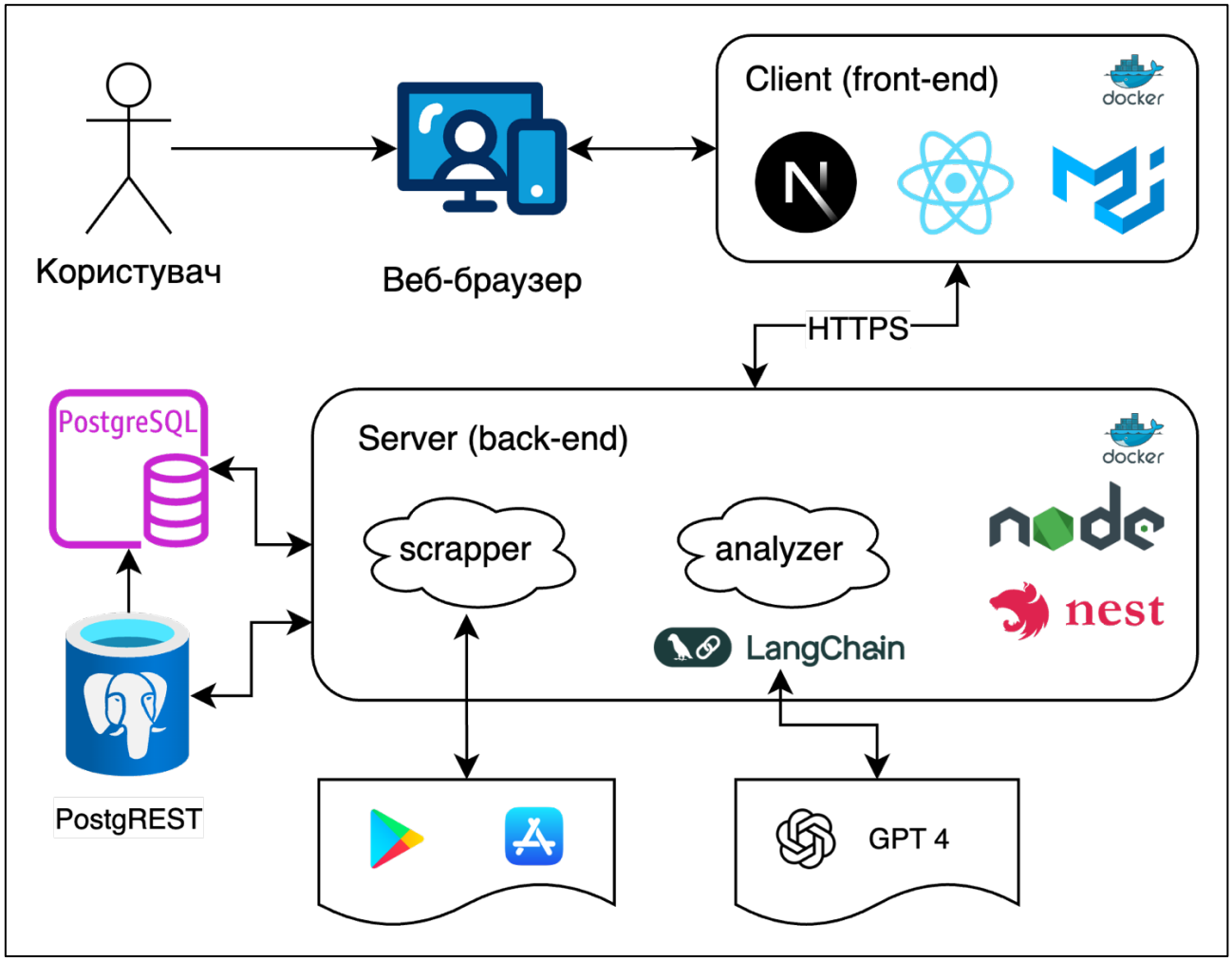


Рисунок Г.3 – Архітектура веб-додатку інтелектуальної системи

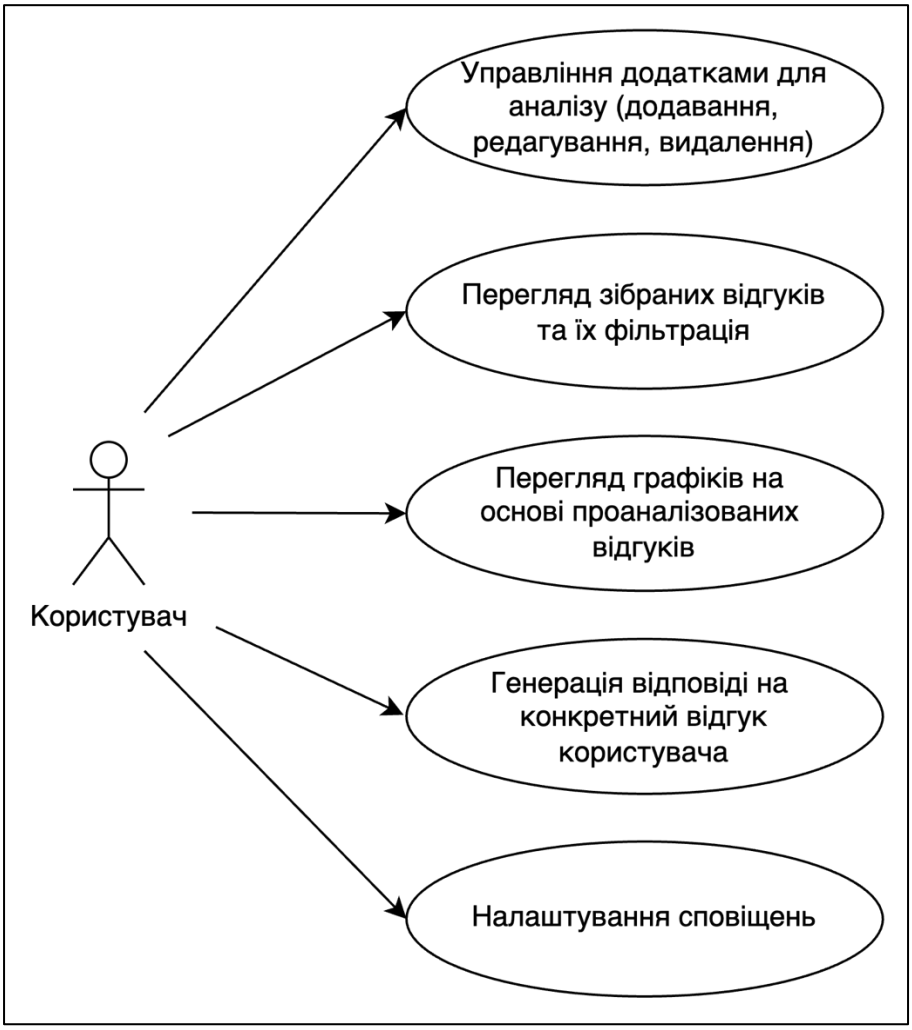


Рисунок Г.4 - Діаграма Use-Case інтелектуальної системи

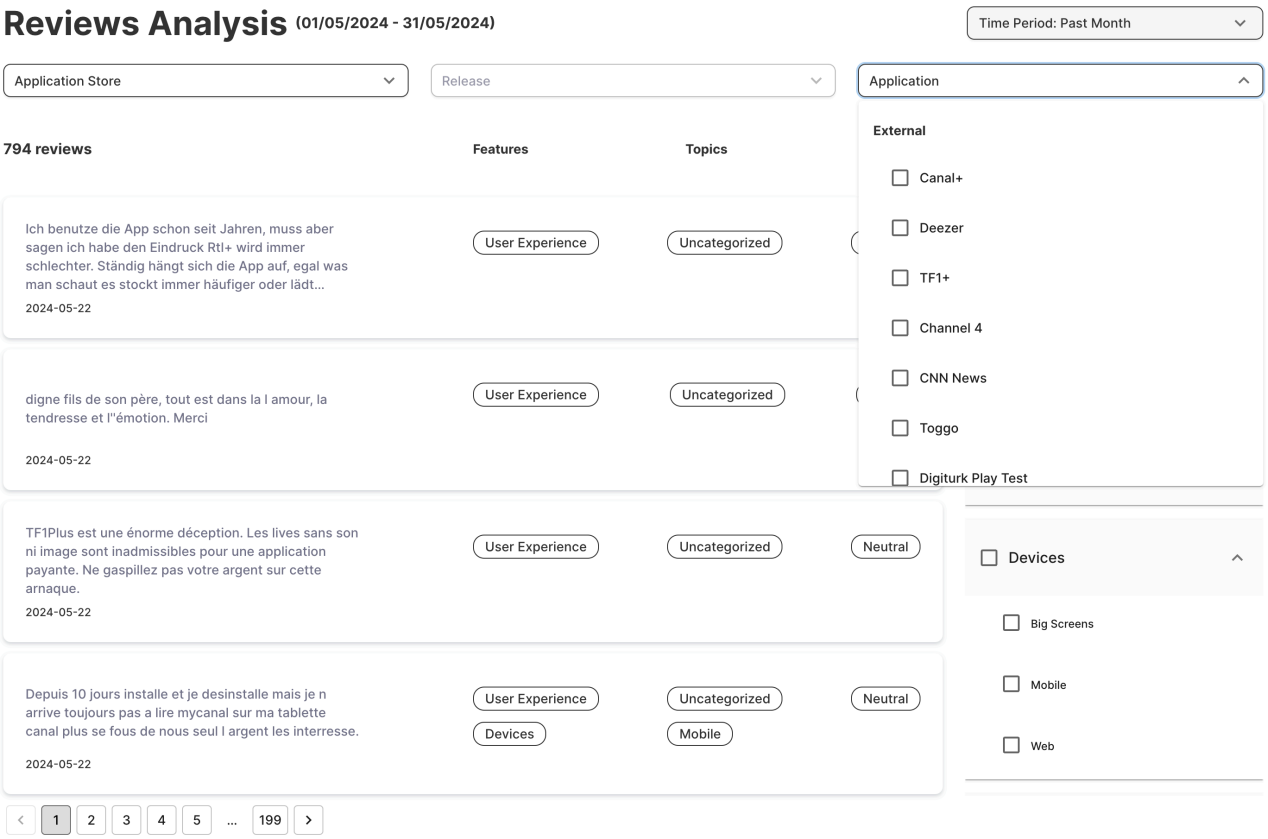


Рисунок Г.5 – Сторінка перегляду зібраних відгуків та їх фільтрації

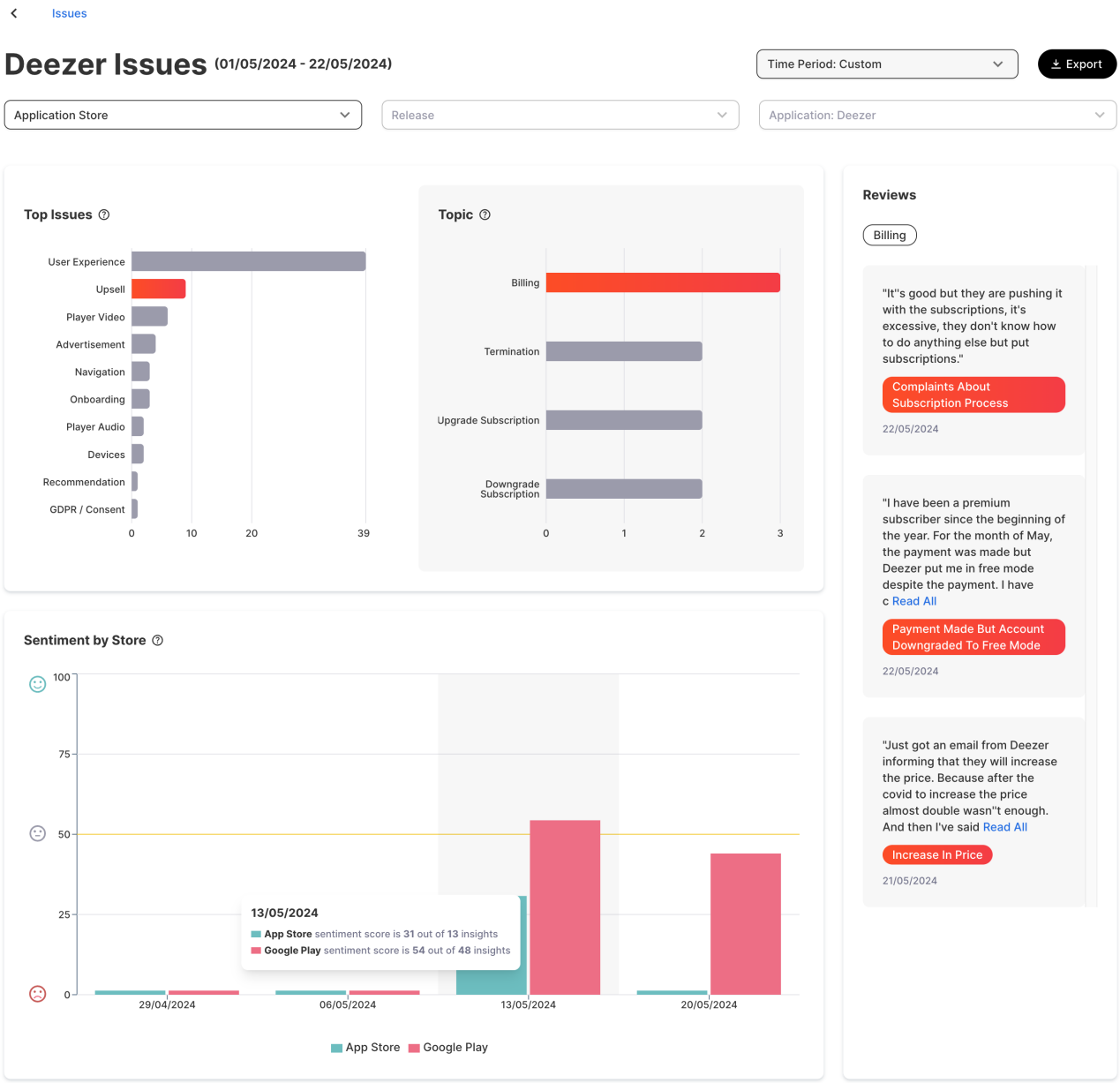


Рисунок Г.6 – Сторінка перегляду відгуків з негативними згадуваннями