Державний торговельно-економічний університет

Кафедра комп'ютерних наук та інформаційних систем

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка автоматизованої системи парсингу даних»

Студента 4 курсу, 9 групи,
спеціальності
122 «Комп'ютерні науки» _____ *підпис студента* Головейчук Олександр Валерійович

Науковий керівник
кандидат фізико-математичних наук, доцент _____ *підпис керівника* Філімонова Тетяна Олегівна

Гарант освітньої програми
кандидат технічних наук, доцент _____ *підпис керівника* Демідов Павло Георгійович

Київ 2023

Державний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра комп'ютерних наук та інформаційних систем
Спеціальність 122 «Комп'ютерні науки»

Затверджую
Зав. кафедри _____Пурський О.І.
«12» грудня 2022р.

Завдання
на випускну кваліфікаційну роботу студенту

Головейчук Олександр Валерійович
(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи (проекту)
«Розробка автоматизованої системи парсингу даних»

Затверджена наказом ректора від *«09» грудня 2022 р. № 3332*

2. Строк здачі студентом закінченої роботи *30 травня2023 року*

3. Цільова установка та вихідні дані до роботи

Мета роботи: розробка автоматизованої системи парсингу даних

Об'єкт дослідження: процес автоматизованої системи парсингу даних

Предмет дослідження: засоби створення автоматизованої системи парсингу даних

4. Перелік графічного матеріалу_____

_____

_____

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Філімонова Т.О.	15.12.2022 р.	15.12.2022 р.
2	Філімонова Т.О.	15.12.2022 р.	15.12.2022 р.
3	Філімонова Т.О.	15.12.2022 р.	15.12.2022 р.

6. Зміст випускної кваліфікаційної роботи (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1 Парсинг даних

1.1 Парсинг та його ключові компоненти

1.2 Етапи парсингу

1.3 Функції та сфери застосування парсингу

РОЗДІЛ 2 Організація розробки автоматизованої системи парсингу даних

2.1 Загальна концепція

2.2 Модель автоматизованої системи парсингу даних

2.2.1 Модуль вилучення даних

2.2.2 Модуль обробки даних

2.2.3 Модуль візуалізації

2.2.4 Модуль інтерфейсу

2.3 Обґрунтування вибору мови програмування, бази даних і бібліотеки

2.3.1 Python

2.3.2 Django

2.3.3 PostgreSQL

2.3.4 Бібліотеки

РОЗДІЛ 3 Реалізація розробки автоматизованої системи парсингу даних

3.1.Програмна реалізація

3.1.1 Структура проекту

7. Календарний план виконання роботи

№ Пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускної кваліфікаційної роботи*	*04.10.2022*	*04.10.2022*
2	*Розробка та затвердження завдання на випускну кваліфікаційну роботу*	*15.12.2022*	*15.12.2022*
3	*Вступ*	*03.02.2023*	*03.02.2023*
4	*РОЗДІЛ 1 Парсинг даних*	*28.02.2023*	*28.02.2023*
5	*РОЗДІЛ 2*	*06.04.2023*	*06.04.2023*

	Організація розробки автоматизованої системи парсингу даних		
6	*РОЗДІЛ 3 Реалізація розробки автоматизованої системи парсингу даних*	*12.05.2023*	*12.05.2023*
7	*Висновки*	*15.05.2023*	*15.05.2023*
8	*Здача випускної кваліфікаційної роботи на кафедру науковому керівнику*	*30.05.2023*	*30.05.2023*
9	*Попередній захист випускної кваліфікаційної роботи*	*31.05.2023 -01.06.2023*	31.05.2023 -01.06.2023
11	*Виправлення зауважень, зовнішнє рецензування випускної кваліфікаційної роботи*	*02.06.2023*	*02.06.2023*
12	*Представлення готової зшитої випускної кваліфікаційної роботи на кафедру*	*05.06.2023*	*05.06.2023*
13	*Публічний захист випускної кваліфікаційної роботи*	*За розкладом роботи ЕК*	

8. Дата видачі завдання «15» грудня 2022 р.

Керівник випускної кваліфікаційної роботи

Філімонова Т.О.
(прізвище, ініціали, підпис)

Гарант освітньої програми Демідов П.Г.
(прізвище, ініціали, підпис)

Завдання прийняв до виконання студент-дипломник

Головейчук О.В.
(прізвище, ініціали, підпис)

9. Відгук керівника випускної кваліфікаційної роботи

У випускній кваліфікаційній роботі розроблено систему автоматизованого парсингу даних за допомогою мови програмування Python, система представлена у вигляді веб-сайту з використанням фреймворку Django. За допомогою цієї системи, користувачі мають можливість отримати глибокий аналіз певної теми чи бренду на платформі Reddit.

Автоматизована система парсингу даних дозволяє збирати та аналізувати дані шляхом автоматизації цього процесу. Система має важливе практичне значення.

Випускна кваліфікаційна робота відповідає всім вимогам до випускних кваліфікаційних робіт. Всі поставлені завдання виконані. Випускна кваліфікаційна

робота можу бути допущена до захисту.

Керівник випускної кваліфікаційної роботи
_____ 30.05.2022 р.
(підпис, дата)

10. Висновок про випускну кваліфікаційну роботу

Випускна кваліфікаційна робота студента _____
(прізвище, ініціали)
може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми_____Демідов П.Г.
(підпис, прізвище, ініціали)

Завідувач кафедри _____Пурський О.І.
(підпис, прізвище, ініціали)

«_____»_____2023 р.

Анотація

У випускній кваліфікаційній роботі розроблено автоматизовану систему парсингу даних для ефективного розв'язання проблем, пов'язаних з обробкою великих обсягів неструктурованих даних, що сприятиме підвищенню ефективності використання даних, покращенню процесів прийняття рішень та оптимізації бізнес-стратегій. Також було здійснено комплексне дослідження та всебічне вивчення технології парсингу даних. Крім того, в роботі представлено розроблену концепцію та модель автоматизованої системи, що забезпечує її ефективність та надійність при роботі з різноманітними даними. Підібрані відповідні технології для підтримки впровадження автоматизованої системи парсингу даних, враховуючи такі фактори, як масштабованість, ефективність та сумісність.

Ключові слова: парсинг, парсинг даних, автоматизована система, неструктуровані дані, вилучення інформації, синтаксичний аналіз.

Annotation

In the final qualification work, an automated data parsing system was developed to effectively solve problems associated with processing large volumes of unstructured data, which will help increase the efficiency of data use, improve decision-making processes, and optimize business strategies. A comprehensive study of data parsing technology was also carried out. In addition, the paper presents the developed concept and model of an automated system that ensures its efficiency and reliability when working with various data. Appropriate technologies have been selected to support the implementation of an automated data parsing system, taking into account such factors as scalability, efficiency, and compatibility.

Keywords: parsing, data parsing, automated system, unstructured data, information extraction, syntactic analysis.

ЗМІСТ

Вступ

Актуальність роботи. З входженням людства у цифрову епоху, інформація та дані стали не від'ємною частиною нашого життя. Дані широко використовують і генеруються майже у всіх видах бізнес-діяльності, і чим більше їх з'являється, тим сильніше постає питання їхньої ефективної обробки. Доволі часто дані являють собою набір необробленої інформації без чіткої структури. Тут і виникає потреба інструменті, що має можливість ефективно приводити дані в більш структурований та зрозумілий формат, саме для цього і була розроблена техніка парсингу даних.

Розробка автоматизованої системи парсингу даних допоможе компаніям чи просто фізичним особам ефективно видобувати релевантну інформацію з великих об'ємів не структурованих даних, заощаджуючи цим велику кількість часу та фінансів.

Мета роботи: розробка автоматизованої системи парсингу даних

Для досягнення цілей роботи, потрібно виконати низку завдань, а саме:

1. Провести загальне дослідження технології парсингу даних.
2. Розробити концепцію та модель системи, що буде створена в процесі.
3. Обрати відповідні технології для розробки програмного продукту.
4. Створити робочу систему автоматизованого парсингу даних.

Об'єкт дослідження: процес автоматизованої системи парсингу даних

Предмет дослідження: засоби створення автоматизованої системи парсингу даних.

Методи дослідження: Загальнонауковий аналітичний метод разом з системним підходом, ідеї провідних дослідників та сучасні дослідження в галузі парсингу даних сформували теоретичну основу дослідження. Інформаційну базу дослідження склали різні ресурси, в тому числі книжкові та онлайн матеріали, такі як документація на технології, що використовувалися при розробці системи.

Для практичного вирішення дослідницьких завдань і проблем були використані наступні методи дослідження:

- загальнонауковий аналітичний метод (розділ 1);
- метод системного моделювання для побудови моделі (розділ 2);
- методи функціонального та алгоритмічного програмування для реалізації системи парсингу даних (розділ 3)

Практичне значення. Автоматизовані системи парсингу набули великого практичного значення в сучасну цифрову епоху. Здатність системи отримувати релевантну інформацію з будь-яких об’ємів структурованих та неструктурованих джерел даних сприяє економії часу та інших ресурсів користувачів. Можливість проведення точного аналізу даних, розширює можливість для прийняття рішень у різноманітних галузях. Загалом, впровадження систем автоматизованого парсингу даних може надати компаніям конкурентну перевагу, дозволяючи їм ефективно обробляти різноманітні види даних.

Техніка парсингу даних широко використовується у різних сферах від фінансів до освіти, він охорони здоров’я до електронної комерції. Для прикладу, у сфері фінансів парсинг даних може використовується для аналізу кредитних звітів, інвестиційних портфелів, перевірки доходів, визначення процентних ставок і термінів погашення кредиту. А в електронній комерції, компанія часто доводиться аналізувати дані клієнтів, щоб краще зрозуміти свою клієнтську базу та їхні потреби. Це включає збір таких даних, як демографія, історія покупок та транзакцій і так далі. Надалі ці дані використовуються для створення більш персоналізованих пропозицій і послуг.

Загалом парсинг є важливою технологією, тому що дає компанія можливість краще використовувати дані та інформацію для ведення бізнесу.

Структура та обсяг випускної кваліфікаційної роботи. Випускна кваліфікаційна робота складається із вступу, трьох розділів, результатів і висновків, списку використаних джерел із 20 найменувань та містить 61 сторінки основного тексту і 29 рисунків.

РОЗДІЛ 1
ПАРСИНГ ДАНИХ

1.1 Парсинг та його ключові компоненти

Парсинг - це процес структурування лінійного представлення відповідно до заданої граматики.[1] Абстрактність визначення зумовлена його широкою інтерпретацією у різних сферах використання.

У лінгвістиці парсинг часто являє собою розбиття речення на складові частини для подальшого розуміння точного значення аналізованого тексту, у програмуванні це частина процесу компіляції коду для подальшого його використання комп’ютером, генерації нового коду чи його оптимізації. А у сфері даних, парсинг може використовуватися для вилучення релевантної інформації зі структурованих або напівструктурованих джерел даних.

Загалом процес парсингу містить розбиття вхідних даних на менші компоненти (токени), які надалі аналізуються за допомогою попередньо визначеного набору правил на основі заданої формальної граматики для розуміння структури та взаємозв'язків між цими токенами. Далі парсер на основі отриманого результату за допомогою алгоритмів парсингу створює абстрактні синтаксичні дерева, що відображають ієрархічну структуру вхідних даних. Специфіка подальшого використання отриманих структур даних залежить уже від цілей та сфери використання парсингу.

Перш ніж детальніше розбирати процес парсингу даних потрібно мати розуміння про фундаментальні концепції синтаксичного аналізу, з яких можна виділити такі:

1. Формальна граматика

Визначення формальної граматики бере свій початок у сфері формальної теорії мов. Його можна описати, як набір правил фіксованого розміру, що визначають спосіб генерації символів для побудови речення, відповідно до

синтаксису мови[2]. У сфері ж загального парсингу воно інтерпретується, як опис коректної побудови структури певного формату даних.

Згідно з ієрархією Чомскі вона поділяється на чотири різні типи, найважливішими з яких є:

1. Контекстно-вільна граматика – зазвичай використовується у мовах програмування та форматах даних.
2. Регулярна граматика – в основному застосовується у сфері регулярних мов.

Граматика в основному складається з набору правил породження для перетворення рядків. Кожне правило визначає заміну певного рядка (його лівої частини) на інший (його праву частину). Граматика додатково розрізняє два типи символів: нетермінальні та термінальні символи; кожна ліва частина повинна містити принаймні один нетермінальний символ. Вона також розрізняє спеціальний нетермінальний символ, який називається початковим символом.

Результат, породжений граматикою, визначається як множина всіх рядків без нетермінальних символів, які можуть бути згенеровані з рядка, що складається з одного стартового символу, шляхом (можливо, багаторазового) застосування її правил у будь-який можливий спосіб.

У формулі 1.1 наведено приклад контекстно-вільної граматики та її застосування для генерації рядка символів. Початок рядок - «S», якщо вибрати правило 1, то буде отримано рядок «aSb». Якщо знову обрати правило 1, то «S» заміниться на «aSb» і вийде рядок «aaSbb». Якщо обрати правило 2, то «S» заміниться на «ba» і вийде рядок «aababb»[3].

$$1.\ S \to aSb$$
$$2.\ S \to ba$$
$$S \to aSB \to aaSbb \to aababb \quad (1.1)$$

де a – термінальний символ;

b – термінальний символ;
S – стартовий символ;

Об’єм даних який можна згенерувати з даної граматики є нескінченною множиною, яку можна описати формулою 1.2.

$$\{a^k bab^k \mid n \geq 0\} = \{ba, abab, aababb, aaababbb, \dots\} \quad (1.2)$$

де a – термінальний символ;
b – термінальний символ;
S – стартовий символ;
k – кількість повторів символу;
n - кількість разів, коли було застосовано правило 1;

Важливість компонента формальної граматики полягає, не тільки в тому, що вона задає набір правил для коректної інтерпретації структури та генерації певного формату даних, але і в тому, що він слугує основою для алгоритмів парсингу, які напряму використовують задані набори правил для роботи з вхідними даними.

2. Дерева парсингу

Для репрезентації ієрархічної структури вхідних даних згідно з формальною граматикою у парсингу використовується структура даних під назвою абстрактне синтаксичне дерево.

Це дерево являє собою графічне представлення синтаксичної структури та складається з вузлів та гілок Рис. 1.1. Вузли відображають різноманітні елементи вхідних даних, а гілки відносини та залежності між цими даними. Дерево зазвичай складається з двох типів вузлів:

1. Вузли, що знаходяться в гілках і представляють нетермінальні символи.
2. Кінцеві вузли, що також називаються листковими листовими та представляють термінальні символи або токени вихідних даних.

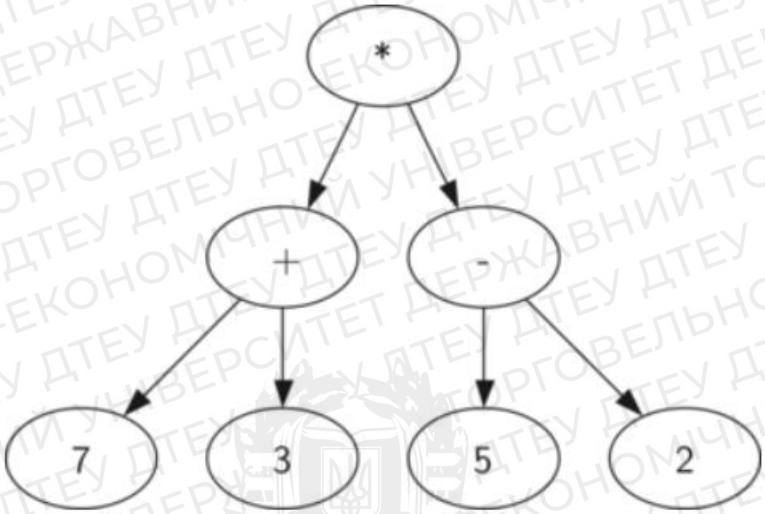

Рис. 1.1. Дерево парсингу яке відображає структура виразу ((7+3)∗(5−2))[3]

Термінальні символи - це літеральні символи, які можуть з'являтися у результатах виконання правил генерації формальної граматики і які не можуть бути змінені. А нетермінальні символи у свою чергу – це символи які також відомі як синтаксичні та вони можуть замінюватись групами термінальних символів згідно з правилами генерації[4].

Абстрактні синтаксичні дерева є важливою частиною парсингу, адже вони допомагають під чат етапу синтаксичного аналізу відображаючи ієрархічну структуру вхідних даних.

3. Алгоритми парсингу

Основним зв'язком між даними та граматикою, з якої вони походять, є дерево розбору, яке описує, як граматика була використана для їх створення. Для реконструкції цього зв'язку і потрібні алгоритми синтаксичного аналізу.

Існує величезна кількість алгоритмів парсингу, але з найважливіших можна виділити тільки два: низхідний та висхідні алгоритми.

Перший алгоритм намагається імітувати оригінальний процес генерації, шляхом відновлення набору даних з початкового елементу. Він називається низхідним, тому що дерево парсингу реконструюється зверху вниз.

Другий алгоритм намагається відкотити процес генерації назад і звести набір даних до початкового елементу. Цілком природно, що цей алгоритм називається висхідним.

4. Контекст

Дана концепція в парсингу стосується контекстної інформації, яка визначає значення вхідних даних, що аналізуються. Контекст може включати широкий спектр факторів, таких як формат даних, область даних та багато чого іншого.

Контекст дуже важливий для парсингу, оскільки він може впливати на інтерпретацію вхідних даних. Наприклад, одна і та ж послідовність елементів може мати різне значення залежно від контексту, в якому вони використовуються. Розглянемо наступні два речення:

1. Я зіткнувся зі своїми друзями вчора.
2. Я зіткнувся з незнайомцем на вході до метро.

У першому реченні слово "зіткнутися" можна інтерпретувати як дієслово, що означає несподівано зустрічатися з ким-небудь. У другому реченні слово "зіткнутися" можна інтерпретувати, як дієслово, що означає ударятися об когось, щось. Контекст, у якому вживається слово, допомагає уточнити його значення.

5. Невизначеність

Цей компонент ставить таке питання: що, якщо на певному етапі синтаксичного аналізу можна зробити більш ніж один вибір?

Не надання повних даних, дані які суперечать самим собі, або декілька правильних інтерпретацій одних і тих самих даних можуть призводити до неоднозначності в процесу парсингу.

Наприклад поширеним прикладом невизначеності в парсингу мов програмування є проблема елемента «else». У деяких мовах цей елемент же не обов’язковим, том що умовні вирази можна записати у двох коректних формах: у формі «if-then» та у формі «if-then-else» роблячи елемент «else» не обов’язковим для написання. Це все призводить до того, що вкладені умовні оператори можуть

розпізнаватись по різному[5].

У різних мовах програмування це вирішують по різному. Іноді граматику роблять однозначною вимагаючи оператор «endif» або роблячи «else» обов'язковим. В інших випадках граматику не змінюють, а роблять її залежною від контексту, як, наприклад асоціюючи «else» з найближчим «if».

Невизначеність є значною проблемою парсингу, оскільки вона може призвести до помилок, не правильних результатів та неправильної інтерпретації вхідних даних. Однак не завжди є можливість обійти її, і тоді синтаксичному аналізатору може знадобитися втручання людини аби розв'язувати проблему неоднозначності.

Знання фундаментальних компонентів парсингу дозволяє повною мірою розуміти сам процес та надалі мати можливість використовувати його реалізувати його основну функцію аналізу строки символів, комп'ютерної мови або будь-якого іншого набору даних відповідно до правил формальної граматики, наприклад з метою вилучення корисної інформації або кращого розуміння загальної структури чи значення цих даних.

1.2 Етапи парсингу

Парсинг зазвичай складається з кількох етапів, які можуть відрізнятися залежно від конкретного типу програми, що використовується, і характеру вхідних даних. Заведено виділяти три етапи парсингу: лексичний, синтаксичний та семантичний аналіз, нижче на Рис. 1.2. наведено ілюстрацію процесу парсингу від початку до кінця.

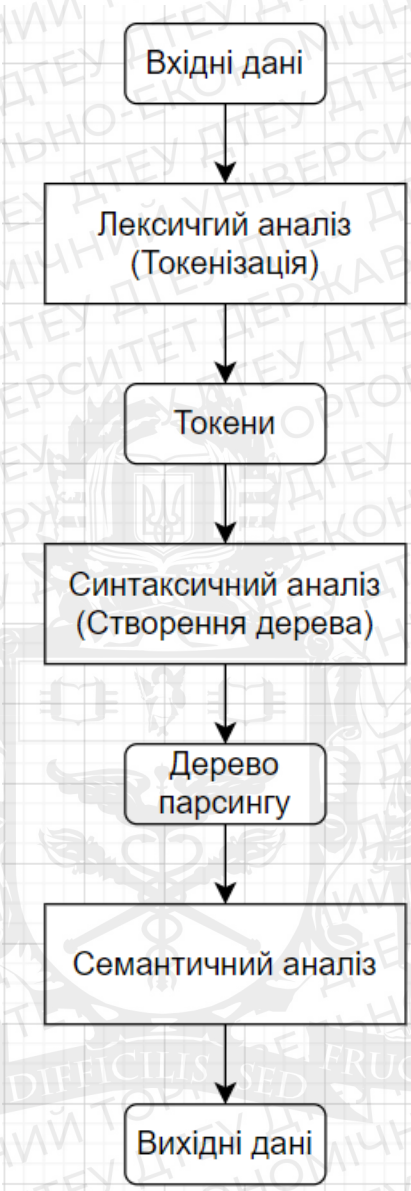

Рис. 1.2. Процес парсингу

1. Лексичний аналіз

Перший етап парсингу, також відомий як лексичний аналіз або генерація токенів, - це процес розбиття вхідних даних певного формату на послідовність токенів.

Токени - це найменші одиниці даних, що мають певне значення та утворюються в результаті процесу токенізації, під час етапу синтаксичного аналізу.[6]

На цьому етапі сканер зчитує вхідні дані символ за символом і групує їх у значущі одиниці, такі як ідентифікатори, ключові слова, оператори, літерали або розділові знаки в залежності від вхідного типу даних. Отримані токени передаються на наступний етап парсингу, де вони аналізуються відповідно до формальної граматики формату даних, що аналізується.

Наприклад, програма-калькулятор розгляне вхідні дані типу "12 * (3 + 4)^2" і розділить їх на токени 12, *, (, 3, +, 4,), ^, 2, кожен з яких є значущим символом у контексті арифметичного виразу. Сканер міститиме правила, які вказуватимуть йому, що символи *, +, ^, (і) позначають початок нового токену, тому безглузді токени на кшталт "12*" або "(3)" не будуть згенеровані[7].

2. Синтаксичний аналіз

Наступною складовою частиною парсингу є синтаксичний аналіз, який ґрунтується на аналізі токенів, згенерованих на попередньому етапі та побудові дерев парсингу. Протягом цього етапу відбувається перевірка на відповідність послідовності токенів щодо правил заданої граматики, і у разі валідності вхідних даних будується дерево парсингу.

Зазвичай у на цьому етапі граматика визначається за допомогою контекстно-вільних граматичних нотацій, таких як BNF або EBNF. Синтаксичний аналізатор генерує дерево розбору, яке є ієрархічним представленням вхідних даних, що показує, як токени згруповані разом на основі правил граматики.

Під час цього етапу аналізатор здатний генерувати синтаксичні помилки, в тому випадку, коли вхідні дані не відповідають очікуваним синтаксичним або граматичним специфікаціям, а отже, не можуть бути належним чином оброблені. Ці помилки можуть містити інформацію про їх характер та місце, де вони були виявлені. Крім того, повідомлення про помилку може містити детальну інформацію про очікуваний токен або синтаксичне правило.

3. Семантичний аналіз

Останній етап парсингу відомий, як семантичний аналіз, його функція являє

собою аналіз та інтерпретацію вхідних даних за межами їхньої синтаксичної структури. Цей етап виявляє та усуває двозначності, невідповідності та будь-які інші можливі джерела некоректної або неочікуваної поведінки у вихідних даних.

Застосування семантичного аналізу відрізняється в залежності від сфери його використання. У контексті компіляції програмного коду, на цьому етапі виконуються семантичні перевірки, такі як перевірка типів (перевірка на наявність помилок типу), зв'язування об'єктів (зв'язування посилань на змінні та функції з їхніми визначеннями), певне присвоєння (вимога ініціалізації всіх локальних змінних перед використанням), відкидання некоректних програм або видача попереджень.[8]

В той самий час, у сфері обробки природної мови, він включає такі задачі, як: аналіз сентиментів, розпізнавання об'єктів і визначення взаємозв'язків між словами або поняттями. А при роботі з даними, на цьому етапі може відбуватись вилучення значення зі структурованих або неструктурованих даних, виявлення закономірностей або кореляцій, а також виконання операцій на основі семантичного контексту.

1.3 Функції та сфери застосування парсингу

Окрім головної функції аналізу даних, парсери можуть виконувати й інші функції, такі як:

- Валідація:

Парсери можна використовувати для валідації вхідних даних шляхом перевірки правильності їхньої семантичної та синтаксичної структури. Це відбувається за допомогою перевірки відповідності щодо формальної граматки певного формату вхідних даних. Ця функція широко використовується у комп’ютерному програмуванні, де забезпечує валідність написаного програмного коду. За допомогою її можна перевіряти на наявність таких помилок як:

синтаксичні помилки, помилки в типах, посиланнях і т.д..

- Перетворення:

Однією з функцій парсингу, яка особливо сильно відноситься до парсингу даних ніж до парсингу загалом є функція перетворення. За допомогою її синтаксичний аналізатор може перетворювати дані одного формату в зовсім інший. Наприклад, модна трансформувати дані з файлу CSV у формат JSON

- Оптимізація:

За допомогою парсингу можна досягати оптимізації та покращення продуктивності певного формату даних шляхом покращення структури, представлення чи використання вхідних даних. Це особливо корисно в таких галузях, як комп'ютерне програмування, адже парсинг часто є частиною ширшого процесу оптимізації коду компілятором.

- Нормалізація:

Нормалізація має безліч значень в залежності у сфери її використання. У парсингу говорячи нормалізація зазвичай мається на увазі перетворення даних у стандартний формат. Ця функція допомагає у забезпечення коректної послідовності та структури даних чи узгодженості даних та їх відповідності певній схемі.

Завдяки переліченим вище функціям парсинг успішно використовується в різних сферах таких як:

- Лінгвістика

Парсинг має широке застосування у сфери лінгвістки, особливо в обробці природної мови. Основне його застосування полягає в аналізі структури та граматики речень, з метою вилучення з тексту точного його значення. Окрім обробки натуральної мови парсинг також застосовується для машинного переводу, сентиментального аналізу, класифікації текстів і т.д..

- Комп'ютерне програмування

Парсинг є невіддільним (частина/ознака) компонентом у сфері

комп’ютерного програмування. Він бере участь у роботі компіляторів, інтерпретаторів мов програмування, та серіалізації даних. В контексті компіляторів та інтерпретаторів парсинг використовується для аналізу синтаксису та конвертації вхідного коду у структуровані представлення такі, як парсингові дерева для його подальшого виконання[9].

- Data Science

Парсинг відіграє важливу роль у сфері Data Science і в основному використовується для задач попередньої обробки даних. Оскільки у цій сфері часто потрібно працювати з великими об’ємами не структурованих даних обійтись без засобу, що здатний ефективно вилучати релевантну інформацію, аналізувати структуру та перетворювати дані з одного формату в більш відповідний майже не можливо.

- Комп’ютерний зір

Будь-яка технологія комп’ютерного зору потребує можливості в розпізнаванні об’єктів, класифікації та сегментації зображень і так далі. Серед технік та методів для реалізації цих функцій знаходиться парсинг., Аналізуючи картинки, алгоритми комп’ютерного зору здатні розрізняти такі властивості об’єктів, як: краї, кути, текстури або кольори. Одним з прикладів використання парсингу в комп’ютерному зорі на практиці, можна виділити його застосування для виявлення незвичної активності на відеозаписах з камер спостереження, шляхом ідентифікації закономірностей та аномалій у вхідних візуальних даних,

Згадані вище сфери застосування парсингу є тільки не великою частиною. Парсинг є фундаментальною технікою, для роботи з будь-якими видами даних, що дозволяє їй мати широкий спектр застосувань у різних областях.

РОЗДІЛ 2
ОРГАНІЗАЦІЯ РОЗРОБКИ АВТОМАТИЗОВАНОЇ СИСТЕМИ ПАРСИНГУ ДАНИХ

2.1 Загальна концепція

Мета цього підрозділу - надати огляд загальної концепції розробки автоматизованої системи парсингу даних, спрямованої на аналіз конкретних брендів або тем на платформі Reddit. Reddit - американська соціальна мережа, сайт для агрегації, рейтингування та обговорення контенту, де користувачі можуть надсилати контент, наприклад, посилання або текстові пости, до певного сабредіту (спільноти, присвяченої певній темі). Станом на лютий 2023 року, за даними Semrush, Reddit є 10-м найбільш відвідуваним веб-сайтом у світі та 6-м найбільш відвідуваним веб-додатком у США. Завдяки своїй великій базі користувачів, яка налічує понад 330 мільйонів активних учасників щомісяця, Reddit пропонує багате джерело даних, які можуть бути використані аналітиками даних, дослідниками та просто зацікавленими людьми для отримання різноманітної інформації на майже будь-які теми.

Основною метою роботи та очікуваним результатом є успішна розробка автоматизованої системи парсингу даних, яка здатна вилучати великі обсяги даних та може ефективно обробляти та аналізувати їх, що дозволить користувачам отримувати аналіз та статистику щодо конкретних тем або брендів на платформі Reddit. Система використовуватиме Reddit API для вилучення таких даних як: пости та коментарі, пов’язані з заданою користувачем темою або брендом, їхня користувацька оцінка та час створення. За допомогою різних методів парсингу буде отримана інформація, шляхом обробки та аналізу вилучених даних. Ця інформація для прикладу включатиме кількість постів та коментарів за певний часовий проміжок, аналіз тональності їх тексту, вилучення ключових слів та іншу релевантну інформацію. Подання цих результатів користувачам передбачається у

вигляді текстової статистики та інтерактивних візуалізацій, для полегшення їхнього розуміння.

Для реалізації автоматизованої системи парсингу даних було вирішено розробити її у формі веб-сайту, який дозволяє користувачам у зручному вигляді отримувати аналіз певного бренду або теми на платформі Reddit.

Цільовою аудиторією цієї системи є широке коло користувачів, включаючи дослідників, аналітиків даних та маркетологів і т.д., які бажають отримати інформацію про конкретні бренди або теми. Окрім професіоналів програмний продукт може зацікавити також і звичайних користувачів мережі Internet, охочих дізнатись про популярність та настрої людей, щодо певних тем чи брендів. За допомогою простої у використанні платформи, користувачі зможуть отримати цінну інформацію з великої кількості даних, доступних на Reddit.

Ефективність додатка буде вимірюватись через залучення користувачів, зворотний зв'язок та точність і швидкість алгоритмів аналізу даних.

Підсумовуючи, загальна концепція цієї роботи полягає в розробці автоматизованої системи парсингу даних, яка отримує вхідні дані з API Reddit та надає користувачам всебічний аналіз конкретних брендів або тем. Завдяки використанню сучасних методів парсингу даних та інструментів візуалізації, очікується, що ця система забезпечить цінну інформацію про тенденції, настрої та думки користувачів Reddit щодо різних тем. Зрештою, ця система може використовуватися компаніями, маркетологами, дослідниками та окремими особами для прийняття обґрунтованих рішень на основі інформації, отриманої в результаті аналізу вилучених даних.

2.2 Модель автоматизованої системи парсингу даних

Розробка надійної та ефективної моделі програмного забезпечення має вирішальне значення перед реалізацією будь-якого проекту. Добре спроектована

модель дозволяє розробникам планувати, аналізувати та перевіряти поведінку і продуктивність програми перед її впровадженням. Моделі полегшують співпрацю та зменшують ризики, виявляючи потенційні проблеми на ранніх стадіях розробки програмного проекту[10]. Модель автоматизованої системи парсингу даних побудована за модульною архітектурою, яка включає різні компоненти, що працюють разом для вилучення, зберігання, парсингу та аналізу даних з платформи Reddit.

Першим компонентом системи є модуль вилучення даних. Цей модуль відповідальний за отримання даних з Reddit API шляхом парсингу даних формату JSON за допомогою бібліотек на основі заданих користувачем параметрів. Цими даними в основному є пости та коментарі, які відразу зберігаються у базі даних для подальшої їх обробки. Модуль призначений для ефективного вилучення великих обсягів даних.

Другим компонентом системи є модуль аналізу даних. Цей модуль відповідальний за обробку, аналіз та парсингу даних у текстовому форматі, що були попередньо вилучені та збережені у базі даних. Модуль використовує методи обробки природної мови та інші передові аналітичні інструменти для виявлення тенденцій, настроїв та закономірностей у вхідних даних. Результати аналізу зберігаються в базі даних для використання модулем візуалізації даних.

Третім компонентом системи є модуль візуалізації даних. Цей модуль візуалізації даних представляє проаналізовані дані в інтерактивному та зручному вигляді. Він генерує візуалізації, такі як графіки та діаграми, які допомагають користувачам зрозуміти ідеї та тенденції, виявлені системою. Він використовує веб-технології для створення динамічних та адаптивних візуалізацій.

Четвертим компонентом системи є модуль інтерфейсу користувача. Цей модуль відповідальний за веб-інтерфейс, який дозволяє користувачам взаємодіяти з системою та отримувати доступ до інформації, згенерованої модулями аналізу та візуалізації даних. Цей модуль складається з набору представлень, які обробляють

запити користувачів, відображають шаблони та повертають відповіді. Модуль інтерфейсу користувача включає форми та інші компоненти, які дозволяють користувачам вводити дані або вибирати параметри для аналізу.

Останнім компонентом системи є модуль бази даних. Цей модуль бази даних забезпечує серверну частину бази даних для зберігання та отримання даних. У ній зберігаються дані, витягнуті з Reddit, та результати аналізу даних. Модуль зберігає елементи задані наборами моделей в інших компонентах, які визначають структуру таблиць бази даних і зв'язки між ними. Система розроблена таким чином, щоб бути масштабованою і може бути адаптована для аналізу різних брендів або тем на Reddit на основі вимог користувача.

2.2.1 Модуль вилучення даних

Для вилучення та парсингу даних з API Reddit цей модуль використовуватиме різні кінцеві точки API, надані Reddit. Для цього він використовуватиме Reddit REST API, який дозволяє отримувати дані з сабредитів, тематичних потоків, окремих дописів та будь-яких інших джерел. Використовуючи Reddit API модуль здатний отримувати інформацію в автоматичному режимі, без втручання людини.

Усі отримані дані зберігаються в локальній базі даних, яка слугує централізованим сховищем зібраної інформації, модуль буде періодично оновлювати вилучені в минулому дані. Регулярне оновлення локальної бази даних гарантує, що система завжди має доступ до найсвіжішої інформації, та зберігає актуальність та релевантність отриманих даних.

Модуль вилучення даних призначений для ефективної обробки великих обсягів даних, що робить його придатним для аналізу інформації з такої популярної платформи, як Reddit. Він безперебійно працює у фоновому режимі з мінімальним

втручанням користувача, забезпечуючи безперервний потік релевантних і актуальних даних.

Цей модуль відіграє важливу роль в спроектовані моделі автоматизованої системи парсингу даних, надаючи надійний і автоматизований механізм для парсингу та зберігання даних з Reddit. Вилучені дані слугують основою для наступних компонентів моделі, які покладаються на вичерпну та актуальну інформацію для аналізу, візуалізації та інших завдань з обробки даних.

2.2.2 Модуль обробки даних

Модуль обробки даних - важливий компонент системи, який відповідає за парсингу, обробку та аналіз даних, що зберігаються в базі даних. Він використовує низку методів парсингу для аналізу даних і виявлення значущої інформації.

Модуль передбачає використання методів обробки природної мови, такі як розпізнавання іменованих об'єктів, для ідентифікації та вилучення таких об'єктів, як люди, організації, продукти та події. Аналізуючи вміст дописів і пов'язані з ними коментарі, модуль фіксує повний набір іменованих об'єктів.

Аналіз тональності тексту - ще один важливий аспект модуля обробки даних. Цей аналіз дає уявлення про загальний настрій і полярність тексту. Модуль також фокусується на виявленні значущих фраз у даних. Цей процес дозволяє виявити фрази, що часто зустрічаються, і пов'язані з ними оцінки тональності.

Результати, отримані за допомогою модуля обробки даних, є важливими для подальшого аналізу та візуалізації. Виявляючи іменовані об'єкти, оцінки настроїв, важливі фрази та популярні теми, модуль сприяє глибшому розумінню даних і допомагає процесам прийняття рішень.

Таким чином, модуль обробки даних відіграє життєво важливу роль у вилученні цінної інформації зі збережених даних. Завдяки методам синтаксичного аналізу, таким як розпізнавання іменованих об'єктів, аналіз настроїв і моделювання

тем, він дозволяє перетворити необроблені дані на змістовну інформацію, надаючи користувачам можливість приймати обґрунтовані рішення на основі проаналізованих даних.

2.2.3 Модуль візуалізації

Модуль візуалізації даних автоматизованої системи парсингу даних є невіддільною частиною для представлення інформації користувачам. Модуль використовуватиме різні бібліотеки та інструменти для створення інтерактивних візуалізацій та графіків на основі проаналізованих даних.

Модуль також включає функціонал для візуалізації кількості постів і коментарів за день. Він використовує бібліотеку Plotly для створення гістограм, які показують щоденну кількість постів і коментарів за певний період часу. Діаграми дають чітке уявлення про активність на платформі Reddit щодо конкретних тем.

Аналіз тональності тексту у часі - ще одна ключова функція візуалізації. Модуль обчислює оцінки настроїв для коментарів і візуалізує їх у вигляді діаграми розсіювання. Аналіз настроїв можна проводити за різні періоди часу, наприклад, за день, тиждень, місяць або рік, що дозволяє користувачам спостерігати тенденції настроїв, пов'язані з конкретними темами.

Модуль візуалізації даних надає користувачеві зручний інтерфейс для взаємодії з візуалізаціями та отримання цінної інформації. Використовуючи можливості таких бібліотек, як Plotly, система надає користувачам більш зручніший та інтуїтивно зрозумілий спосіб розуміння результатів аналізу.

2.2.4 Модуль інтерфейсу

Цей модуль орієнтований на користувача та дозволяє йому взаємодіяти з системою та отримувати доступ до проаналізованих даних. Загалом система веб-

сайт буде складатись з двох сторінок: головної сторінки та сторінки результатів, саме за і відповідає модуль інтерфейсу.

Головна сторінка слугує точкою входу для користувачів, де вони можуть ввести бажану тему для аналізу. Ця сторінка має зручний інтерфейс, що дозволяє користувачам легко вводити свої запити та вказувати діапазон дат для аналізу. Дизайн і макет головної сторінки зосереджені на забезпеченні безперешкодної роботи користувача, гарантуючи, що користувачі можуть легко орієнтуватися в системі та взаємодіяти з нею.

Після створення запиту, користувачі потрапляють на сторінку аналізу, де вони можуть ознайомитися з проаналізованими даними та їхніми висновками. На сторінці аналізу результати представлені в чіткій та організованій формі, що дозволяє користувачам ефективно розуміти та інтерпретувати дані. Ціль полягає в тому, щоб надати користувачам інтуїтивно зрозумілий та інформативний інтерфейс, який покращує їхнє розуміння проаналізованих даних.

Сторінка аналізу може містити різні компоненти, такі як список проаналізованих дописів, пов'язаних із зазначеною темою, відповідну статистику та ключові висновки, отримані в процесі аналізу даних. Мета полягає в тому, щоб надати користувачам всебічний огляд проаналізованих даних та інтуїтивно зрозумілий та інформативний інтерфейс, для кращого розуміння результатів обробки даних.

Важливо зазначити, що модуль інтерфейсу зосереджений на представленні проаналізованих даних користувачам і полегшенні їхньої взаємодії з системою. Складні завдання з обробки та візуалізації даних, виконуються іншими модулями системи. Модуль інтерфейсу виступає сполучною ланкою між користувачами та основними функціональними можливостями системи, забезпечуючи безперебійну та зручну роботу з нею.

2.3 Обґрунтування вибору мови програмування, бази даних і бібліотеки

Вибір мови програмування, бази даних та бібліотеки має вирішальне значення при розробці будь-якого програмного продукту. Ці технології складають основу системи та безпосередньо впливають на її продуктивність, масштабованість і функціональність. Для програмної реалізації системи автоматизованого парсингу даних було вирішено обрати мову програмування Python та її веб-фреймворк Django, що дозволить використовувати такі бібліотеки, як: PRAW, SpaCy, NetworkX та інші. Для збереження даних використовуватиметься система управління базами даних PostgreSQL.

2.3.1 Python

Python - це інтерпретована мова програмування високого рівня, яка набула широкого розповсюдження з моменту свого випуску в 1991 році. Розроблена Гвідо ван Россумом, Python відома своєю простотою, читабельністю та легкістю у використанні. Python - це універсальна мова загального призначення, яка знаходить своє застосування в різних галузях, включаючи веб-розробку, наукові обчислення, аналіз даних, штучний інтелект і машинне навчання.

Кросплатформність Python дозволяє йому працювати на різних операційних системах, таких як Windows, Linux та macOS. Зрозумілий синтаксис мови та простота використання роблять її ідеальним вибором для проектів з веб-розробки. Крім того, універсальні веб-фреймворки Python, такі як Flask, Django та Pyramid, надають розробникам необхідні інструменти для створення веб-додатків з легкістю та ефективністю.

Значний внесок Python в аналіз даних зробив її популярним вибором для додатків, орієнтованих на роботу з даними, таких як, наприклад системи парсингу даних. Її велика бібліотека та інструменти для аналізу даних дозволяють розробникам виконувати складні завдання аналізу даних з мінімальними

зусиллями. Крім того, потужні бібліотеки Python, такі як NumPy, SciPy та Pandas, роблять її привабливим вибором для розробників, які прагнуть впровадити у свої програми передові технології, такі як машинне навчання та штучний інтелект[11].

Універсальність, простота використання та багатий набір бібліотек для аналізу даних та веб розробки роблять Python оптимальним вибором для побудови автоматизованих систем аналізу даних. Спільнота розробників і користувачів, що підтримує цю мову, створює міцну основу для навчання нових розробників і сприяє зростанню екосистеми Python. Популярність Python також означає, що розробникам доступна велика кількість ресурсів, включаючи навчальні посібники, форуми та проекти з відкритим вихідним кодом, що ще більше підвищує привабливість її вибору, як мови програмування для автоматизованої системи аналізу даних.

2.3.2 Django

Для розробки веб-сайту було обрано фреймворк Django. Рішення використовувати Django як основний веб-фреймворк для розробки веб-додатку було прийнято після ретельного аналізу доступних веб-фреймворків.

Django - це веб-фреймворк з відкритим вихідним кодом, написаний на мові Python і побудований за архітектурною схемою "модель-вид-контролер" (MVC). Розроблений у 2005 році Адріаном Головатим та Саймоном Віллісоном, Django став одним з найпопулярніших веб-фреймворків для розробки складних веб-додатків. Філософія дизайну Django підкреслює принцип "Не повторюй себе" (DRY) і сприяє швидкій розробці, чистому і прагматичному дизайну та надійності.

Django відомий своєю вбудованою підтримкою таких важливих функцій, як автентифікація, міграція схем баз даних та шаблонування. Його об'єктно-реляційне відображення (ORM) спрощує процес інтеграції баз даних для розробників. Крім

того, механізм шаблонів Django зменшує дублювання коду, дозволяючи створювати шаблони для багаторазового використання.

Активна спільнота розробників створила багату екосистему сторонніх пакетів та розширень, які додали Django ще більшої привабливості. Наприклад, Django REST Framework дозволяє розробникам створювати RESTful API, які є надійними та масштабованими з легкістю. Крім того, Django CMS - це популярна система управління контентом, яка надає розробникам гнучкість та можливості кастомізації для створення високо адаптивних веб-сайтів[12]. В поєднанні з вбудованими функціями, інструменти та легкістю в освоєнні роблять Django ідеальним веб-фреймворком для розробки. автоматизованої системи парсингу даних у вигляді веб-додатку.

2.3.3 PostgreSQL

Для зберігання даних використовуватиметься PostgreSQL

Рішення використовувати PostgreSQL для зберігання даних базується на кількох факторах, які роблять її привабливим вибором для сучасної розробки програмного забезпечення. PostgreSQL - це об'єктно-реляційна система управління базами даних з відкритим вихідним кодом, яка може запропонувати безліч можливостей, включаючи високу масштабованість, надійність і продуктивність. Система добре зарекомендувала себе в середовищі розробників програмного забезпечення і має велику базу користувачів, які постійно отримують активну підтримку, часті оновлення та численні розширення для аналізу даних та інтеграції з іншими інструментами.

Однією з головних особливостей, які роблять PostgreSQL хорошим вибором для цього проекту, є її здатність ефективно обробляти великі обсяги даних. PostgreSQL забезпечує надійну підтримку складних типів даних, індексів та методів оптимізації запитів, які дозволяють швидко і точно отримувати дані. Крім

того, розширені можливості PostgreSQL, такі як збережені процедури, тригери та обмеження зовнішніх ключів, полегшують керування складними структурами даних та забезпечують їх цілісність[13].

На додаток до своїх технічних можливостей, PostgreSQL має кілька інших переваг над іншими базами даних. Наприклад, PostgreSQL повністю сумісна зі стандартом ACID, що забезпечує узгодженість і надійність даних. Крім того, PostgreSQL сумісна з широким спектром мов програмування, включаючи C# та Python, що дозволяє легко інтегрувати її з іншими частинами системи.

Однак, однією з потенційних проблем використання PostgreSQL є її встановлення та конфігурація. PostgreSQL вимагає певного рівня знань з адміністрування баз даних, і може бути складно налаштувати базу даних для оптимальної продуктивності. Крім того, PostgreSQL може бути не найкращим варіантом для проектів, які вимагають високого рівня гнучкості, оскільки вона може бути менш поблажливою, ніж інші бази даних, коли мова йде про зміни схеми.

2.3.4 Бібліотеки

Система використовуватиме низку бібліотек для вилучення даних з Reddit, парсингу, аналізу та графічному відображенню. Основні бібліотеки такі:

1. SpaCy

SpaCy - популярна бібліотека з відкритим вихідним кодом для обробки природної мови (NLP) мовою Python. Вона розроблена для ефективного, швидкого та простого використання, що робить її популярним вибором для розробників, яким потрібно обробляти великі обсяги тексту. SpaCy надає широкий спектр інструментів для таких завдань, як токенізація, тегування частин мови, розпізнавання іменованих сутностей та синтаксичний аналіз залежностей[14]. Ці можливості роблять його чудовим вибором для аналізу та вилучення інформації з текстових даних, таких як коментарі та пости на Reddit.

2. NetworkX

NetworkX - це бібліотека Python для створення, маніпулювання та дослідження складних мереж[15]. Він надає інструменти для побудови графів, їх візуалізації та виконання складних графових алгоритмів. NetworkX широко використовується в галузях науки про дані та машинного навчання, оскільки є чудовим інструментом для аналізу та моделювання складних систем. З його допомогою можна створити візуалізації на кшталт діаграм розсіювання, гістограм та іншого.

3. PRAW

PRAW, або Python Reddit API Wrapper - це бібліотека Python для взаємодії з API Reddit[16]. Він забезпечує простий та інтуїтивно зрозумілий спосіб взаємодії з Reddit, дозволяючи розробникам отримувати такі дані, як дописи, коментарі та інформацію про користувачів. PRAW надає широкий спектр функціональних можливостей, включаючи автентифікацію, обмеження швидкості та кешування, що робить його чудовим вибором для створення додатків, керованих даними, які покладаються на дані Reddit.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ РОЗРОБКИ АВТОМАТИЗОВАНОЇ СИСТЕМИ ПАРСИНГУ ДАНИХ

3.1 Програмна реалізація

Система складається з декількох елементів, включаючи головну сторінку з пошуковим рядком, де також відображаються популярні пости, статистика та їхній аналіз та сторінку аналізу за певною темою або брендом.

Для розробки програми було використано Visual Studio Code, яке пропонує інтегроване середовище розробки (IDE) з розширеними можливостями, такими як налагодження, завершення коду та рефакторинг коду. Реалізація цієї системи вимагала декількох кроків, включаючи встановлення Python, Django, PostgreSQL, SpaCy, NetworkX і PRAW, а також конфігурацію необхідних налаштувань і залежностей. Програма також потребувала створення численних кастомних моделей, представлень і функцій для забезпечення бажаної функціональності.

У цьому розділі надано детальний опис реалізації програми, включаючи структуру проекту, використання бібліотек та фреймворків, деталі реалізації й так далі.

3.1.1 Структура проекту

Система автоматизованого парсингу даних має дану структуру Рис. 3.1.

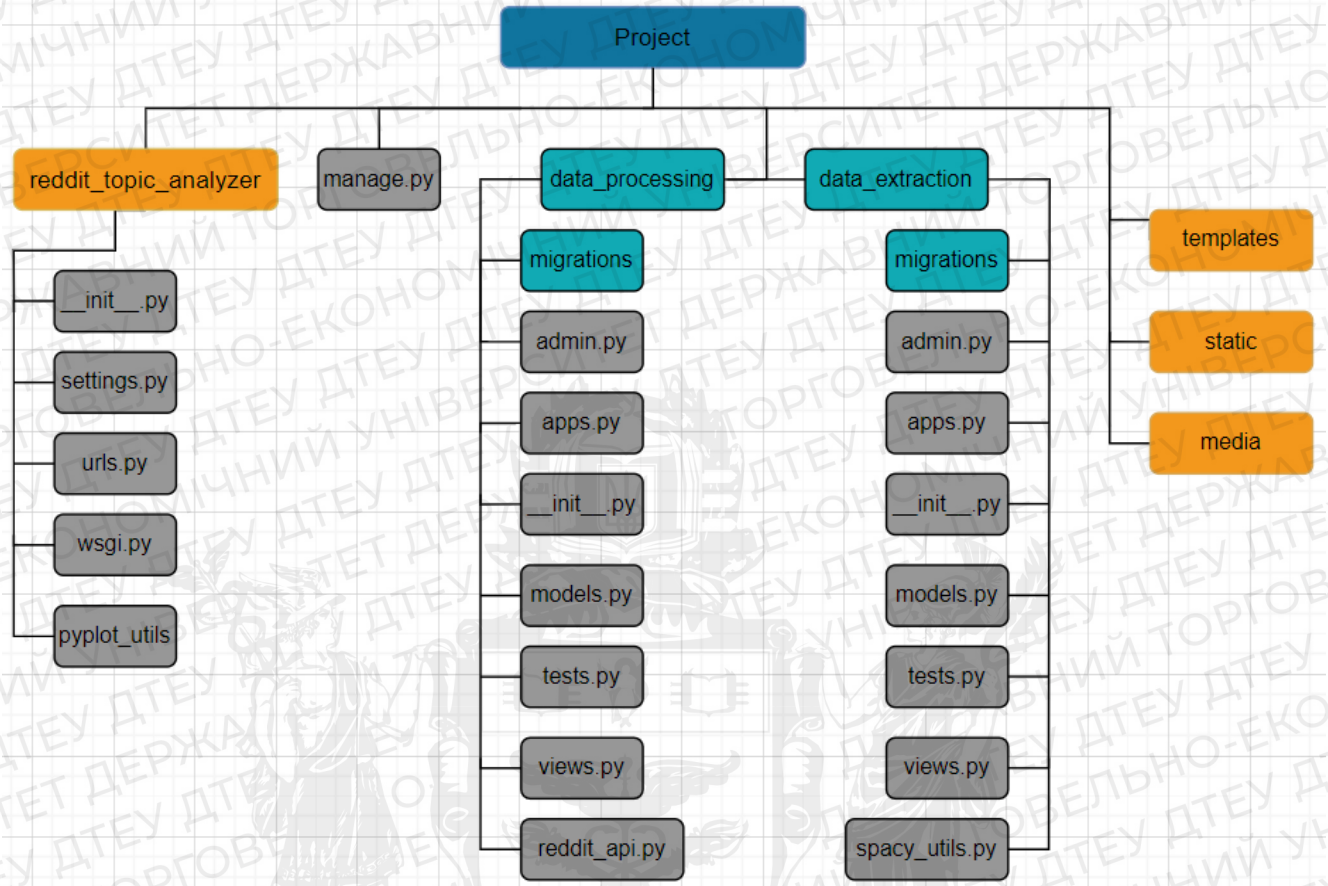

Рис. 3.1 Структура проекту

Основними компонентами структури системи є три додатки:

1. Веб-додаток (reddit_topic_analyzer)

Цей компонент відповідальний за візуалізацію, конфігурацію та виведення результатів на графічний інтерфейс веб-сайту.

2. Додаток вилучення даних (data_extraction)

Цей компонент відповідає за вилучення та парсинг даних з платформи Reddit, використовуючи бібліотеку PRAW та збереження їх у базі даних.

3. Додаток обробки даних (data_processing)

Цей компонент відповідає за обробку та парсинг даних вилучених з Reddit, результат обробки зберігається у базі даних або напряму потрапляє у веб-додаток для візуалізації.

Окрім додатків у системі присутні статичні інші файли та директорії:

1. Static - скрипти, Java Script код, CSS та інше.
2. Templates - HTML-файли.
3. Media - зображення і медіа-контент.
4. Manage.py - утиліта командного рядка, яка дозволяє взаємодіяти з проектом Django різними способами

3.1.2 Формування бази даних

Для функціонування веб-додатка потрібно створити базу даних. База даних призначена для зберігання інформації та для здійснення доступу до цих даних. Вона містить у собі зв’язані одна з одною таблиці, при цьому кожна зберігає в собі певну інформацію. Структура бази даних зображена на Рис. 3.2.

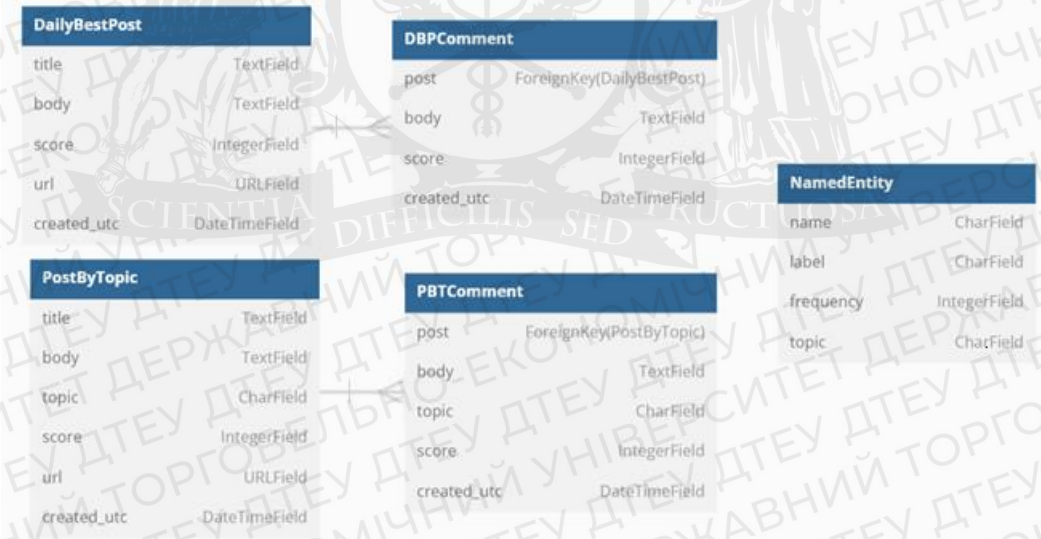

Рис. 3.2 Структура бази даних

Для того, щоб з даними працювати, кожен раз, коли на сервер буде надіслано запит відповідно до сервера буде надісланий запит у форматі мови запитів SQL, для кожної таблиці буде надісланий персональний запит

Для зберігання даних про користувачів курси, завдання, рішення та контент розроблена система використовує реляційну базу даних, створену на базі

системи управління базами даних PostgreSQL засобами Python веб-фреймворку Django.

Для комунікації із базою даних Django використовує засоби моделей та представлень архітектури MTV, яка є основою фреймворку.

База даних складається із 10 таблиць:

- DailyBestPost - використовується для представлення найпопулярніших щоденних дописів на Reddit. Таблиця містить наступні поля: "title", "body", "score", "URL" і "created_utc".
- PostByTopic - використовується для зберігання інформації про дописи на Reddit, пов'язані з певними темами. Вона має ті ж поля, що і таблиця "DailyBestPost", включаючи "title", "body", "score", "url" і "created_utc" окрім поля "topic", яке зберігає тему, яка асоціюється з дописом, що дозволяє легко фільтрувати та організовувати дописи за темами.
- DBPComment - представляє коментар до найпопулярніших щоденних дописів на Reddit. Вона містить наступні поля: "post", "body", "score" і "created_utc".
- PBTComment - використовується для зберігання інформації про коментарі до дописів на Reddit, які пов'язані з певними темами. Вона має ті ж поля, що і модель "DBPComment", включаючи "post", "body", "score", "created_utc" та “topic”.
- NamedEntity - використовується для зберігання інформації про іменовані сутності, знайдені в дописах і коментарях на Reddit. Вона має чотири поля: "name", "label", "frequency" і "topic".

3.1.3 Створення додатку вилучення даних

Цей додаток містить функції, які використовують PRAW Reddit API для вилучення та парсингу даних з Reddit, а потім створюють і зберігають об'єкти в Django ORM для представлення вилучених даних.

Модуль починається з імпорту необхідних бібліотек, включаючи бібліотеку PRAW для взаємодії з API Reddit, бібліотеку datetime для обробки часових даних і відповідні моделі з додатка data_extraction для збереження даних. Він також ініціалізує об'єкт Reddit за допомогою PRAW, який використовується для доступу до API. Модуль містить дві основні функції: get_daily_top_posts() та get_posts_by_topic()

1. Функція get_daily_best_posts()

Перша функція - get_daily_best_posts() Рис. 3.3. Вона використовує PRAW Reddit API для отримання найпопулярніших дописів на сабредіті "all" за певний день. Потім вона проводить парсинг даних отриманих з Reddit та зберігає результати у вигляді моделей в Django ORM як для поста, так і для коментарів до нього.

Функція отримує один параметр - 'date', який є рядком, що представляє дату, за яку потрібно отримати найпопулярніший допис. Спочатку функція видаляє всі наявні об'єкти DailyBestPost, щоб забезпечити чистий аркуш для нових даних.

Потім вона знаходить найпопулярніший допис за вказану дату за допомогою методу subreddit.top() з часовим фільтром, встановленим на 'day', і лімітом, встановленим на 1. Вона витягує відповідні дані, такі як назва допису, URL, тіло, оцінка і час створення, і додає їх до списку дописів.

Далі він створює об'єкт DailyBestPost для знайденого допису за допомогою методу bulk_create() Django ORM. Потім вона витягує коментарі до посту і зберігає їх як об'єкти DBPComment за допомогою того ж методу.

Нарешті, функція повертає словник, що містить пост і коментарі до нього, а також кількість знайдених коментарів.

Загалом, функція get_daily_top_posts() надає зручний спосіб отримати та проаналізувати найпопулярніші дописи та коментарі до них за певний день у "всіх" сабредітах.

```python
def get_daily_best_posts():
    subreddit = reddit.subreddit('all')
    posts = []
    DailyBestPost.objects.all().delete()

    for post in subreddit.top('day', limit=1):
        data = {
            'title': post.title,
            'url': 'https://www.reddit.com' + post.permalink,
            'body': post.selftext,
            'score': post.score,
            'created_utc': datetime.datetime.fromtimestamp(post.created_utc),
        }
        posts.append(data)

    daily_best_posts = [DailyBestPost(**post) for post in posts]
    DailyBestPost.objects.bulk_create(daily_best_posts)

    comments = []
    for post in daily_best_posts:
        submission = reddit.submission(url=post.url)
        submission.comments.replace_more(limit=0)
        post_comments = submission.comments.list()
        for comment in post_comments:
            comment_data = {
                'post': post,
                'body': comment.body,
                'score': comment.score,
                'created_utc': datetime.datetime.fromtimestamp(comment.created_utc),
            }
            comments.append(comment_data)

    daily_best_posts_comments = [DBPComment(**comment) for comment in comments]
    DBPComment.objects.bulk_create(daily_best_posts_comments)

    num_posts = len(posts)
    num_comments = len(comments)

    return {'posts': posts, 'comments': comments, 'num_posts': num_posts, 'num_comments': num_comments}
```

Рис. 3.3 Функція get_daily_best_posts()

2. Функція get_posts_by_topic()

Друга функція - get_posts_by_topic() Рис 3.4. Ця функція використовує PRAW Reddit API для пошуку дописів у сабредіті "all" на основі заданої теми та діапазону даних. Потім вона проводить парсинг отриманих даних та зберігає результати у вигляді моделей Django ORM, як для самих дописів, так і для коментарів до них.

Загалом ця функція майже ідентична до першої, основна відмінність полягає у використанні змінних “topic”, “data_range”, “post_number”. Для визначення теми або бренду для пошуку, часового діапазону та кількості постів для аналізу.

```python
def get_posts_by_topic(topic, date_range, post_number):
    subreddit = reddit.subreddit('all')
    keyword = topic.lower()

    PostByTopic.objects.filter(topic=topic).delete()
    PBTComment.objects.filter(topic=topic).delete()

    posts = [
        PostByTopic(
            title=post.title,
            url='https://www.reddit.com' + post.permalink,
            body=post.selftext,
            score=post.score,
            topic=topic,
            created_utc=datetime.datetime.fromtimestamp(post.created_utc)
        )
        for post in subreddit.search(keyword, time_filter=date_range, limit=int(post_number))
        if keyword in post.title.lower()
    ]
    PostByTopic.objects.bulk_create(posts, batch_size=1000)

    comments = [
        PBTComment(
            post_id=post.id,
            body=comment.body,
            score=comment.score,
            topic=topic,
            created_utc=datetime.datetime.fromtimestamp(comment.created_utc)
        )
        for post in PostByTopic.objects.filter(topic=topic)
        for comment in reddit.submission(url=post.url).comments.list()
        if not isinstance(comment, MoreComments)
    ]
    PBTComment.objects.bulk_create(comments, batch_size=1000)

    num_posts = len(posts)
    num_comments = len(comments)

    return {'posts': posts, 'comments': comments, 'num_posts': num_posts, 'num_comments': num_comments}
```

Рис. 3.4. Функція get_posts_by_topic()

3.1.4 Створення додатку обробки даних

Додаток data_processing є критично важливим компонентом веб-сайту, що відповідає за парсинг та аналіз даних, пов'язаних з пошуковими запитами користувачів. Ця програма використовує різні технології, зокрема бібліотеку обробки природної мови SpaCy, бібліотеку аналізу тексту TextBlob і колекції бібліотек Python, щоб виокремлювати іменовані сутності з дописів і коментарів, обчислювати їхню частоту і надавати уявлення про їхні настрої.

Основним модулем програми є spacy_utils.py, який містить набір функцій для аналізу та парсингу текстів постів й коментарів. Модуль spacy_utils.py виконує чотири основні функції:

- Вилучення іменованих сутностей з найпопулярніших постів за день за допомогою функцій update_daily_named_entities і get_top_daily_entities.
- Вилучення іменованих сутностей з постів на обрану тему за допомогою функцій update_named_entities і get_top_entities.
- Аналіз тональності вибраного тексту за допомогою функції get_sentiment.

Для виконання цих завдань код завантажує попередньо навчену модель (en_core_web_sm) і додає до неї компонент “spacytextblob”. Крім того, код використовує бібліотеку TextBlob для аналізу тональності тексту. Клас NamedEntity та Phrase з models.py використовується для зберігання вилучених іменованих сутностей, а клас Counter з модуля collections - для підрахунку входжень сутностей.

1. Функція update_daily_named_entities() та get_top_daily_entities() Рис. 3.5.

Функція update_daily_named_entities() відповідає за вилучення іменованих сутностей з найпопулярніших дописів за день. Спочатку вона очищає наявні іменовані сутності, які не мають теми, а потім перебирає всі дописи моделі DailyBestPost. Потім функція використовує spacy для вилучення сутностей типу "PERSON", "ORG", "PRODUCT" і "EVENT" з заголовка і тіла кожного допису. Аналогічно, вона витягує іменовані сутності з коментарів, пов'язаних з кожним дописом. Всі ці сутності підраховуються, і отриманий список використовується для оновлення моделі NamedEntity, де кожна сутність зберігається з її назвою, міткою, частотою і порожньою темою.

Функція get_top_daily_entities(n) повертає список n найпопулярніших сутностей за частотою, відсортованих у порядку спадання. Ця функція спочатку викликає функцію update_daily_named_entities(), щоб переконатися, що модель NamedEntity є актуальною. Потім вона отримує n сутностей з найвищою частотою з моделі NamedEntity і повертає їх у вигляді списку кортежів, що містять назву сутності та її частоту.

```python
def update_daily_named_entities():
    # return all entities even with topic FIX
    NamedEntity.objects.filter(topic='').all().delete()  # clear existing entities
    entities = []
    for post in DailyBestPost.objects.all():
        doc = nlp(post.title + ' ' + post.body)
        entities += [(ent.text, ent.label_) for ent in doc.ents if ent.label_ in ('PERSON', 'ORG', 'PRODUCT', 'EVENT')]
        for comment in DBPComment.objects.all():
            doc = nlp(comment.body)
            entities += [(ent.text, ent.label_) for ent in doc.ents if ent.label_ in ('PERSON', 'ORG', 'PRODUCT', 'EVENT')]
    counter = collections.Counter(entities)
    for (name, label), freq in counter.items():
        entity, created = NamedEntity.objects.get_or_create(name=name, label=label, topic='')
        entity.frequency += freq
        entity.save()

def get_top_daily_entities(n=15):
    update_daily_named_entities()
    entities = NamedEntity.objects.all().order_by('-frequency')[:n]
    return [(entity.name, entity.frequency) for entity in entities]
```

Рис. 3.5. Функція update_daily_named_entities() та get_top_daily_entities()

2. Функція update_named_entities() та get_top_entities() Рис. 3.6.

Функція update_named_entities витягує іменовані сутності з дописів у вибраній темі та створює або оновлює відповідні об'єкти в моделі NamedEntity. Спочатку вона очищає наявні сутності для теми, а потім обробляє всі дописи та коментарі, пов'язані з цією темою.

Функція get_top_entities знаходить найкращі n іменованих сутностей для вибраної теми, запитуючи модель NamedEntity і впорядковуючи результати за частотою. Вона викликає внутрішній виклик update_named_entities, щоб переконатися, що доступні найсвіжіші сутності.

Загалом функції схожі до перших двох з однією лиш різницею у використанні “topic” змінної та роботи з іншими моделями даних.

```python
def update_named_entities(topic):
    NamedEntity.objects.filter(topic=topic).delete()  # clear existing entities
    entities = []
    for post in PostByTopic.objects.filter(topic=topic):
        doc = nlp(post.title + ' ' + post.body)
        entities += [(ent.text, ent.label_) for ent in doc.ents if ent.label_ in ('PERSON', 'ORG', 'PRODUCT', 'EVENT')]
        for comment in PBTComment.objects.filter(topic=topic):
            doc = nlp(comment.body)
            entities += [(ent.text, ent.label_) for ent in doc.ents if ent.label_ in ('PERSON', 'ORG', 'PRODUCT', 'EVENT')]
    counter = collections.Counter(entities)
    for (name, label), freq in counter.items():
        entity, created = NamedEntity.objects.get_or_create(name=name, label=label, topic=topic)
        entity.frequency += freq
        entity.save()

def get_top_entities(topic):
    update_named_entities(topic)
    entities = NamedEntity.objects.filter(topic=topic).order_by('-frequency')[:15]
    return [(entity.name, entity.frequency) for entity in entities]
```

Рис. 3.6. Функція update_named_entities() та get_top_entities()

3. Функція get_sentiment() Рис. 3.7.

Функція get_sentiment виконує аналіз тональності заданого тексту за допомогою бібліотеки TextBlob. Вхідний текст спочатку обробляється конвеєром “nlp” з SpaCy для створення об'єкта “Doc”. Потім результат аналізу настроїв TextBlob витягується з атрибутом “_.blob.polarity” об'єкта “Doc”, який представляє полярність настроїв тексту за шкалою від -1 до 1, де від'ємні значення представляють негативні настрої, додатні - позитивні, а 0 - нейтральні. Функція повертає оцінку полярності настрою.

```python
def get_sentiment(text):
    doc = nlp(text)
    sentiment_score = doc._.blob.polarity
    return sentiment_score
```

Рис. 3.7. Функція get_sentiment()

3.1.5 Створення основного веб-додатку

Reddit_topic_analyzer є основним додатком у проекті Django і відповідає за конфігурацію проекту, включаючи підключення до бази даних та схеми маршрутизації URL-адрес. Крім того, він відповідає за візуалізацію діаграм, які

представляють оброблені дані користувачам в інтуїтивно зрозумілий і візуально привабливий спосіб, а також обробляє запити користувачів і контролює потік даних від бекенду до фронтенду. Додаток складається з трьох основних модулів: views.py, pyplot_utils.py та settings.py.

1. Модуль views.py

Модуль views.py імпортує різні модулі та функції, необхідні для вилучення та обробки даних, зокрема json, reddit_api, spacy_utils та pyplot_utils.

Функція home у views.py отримує дані з API Reddit за допомогою функції reddit_api.get_daily_best_posts(), а потім передає отримані дані функції spacy_utils.get_top_daily_entities() для вилучення найкращих об'єктів з щоденних найкращих дописів. Отримані дані потім використовуються для рендерингу шаблону home.html, який відображає найкращі дописи за день і витягнуті сутності.

Функція search_results відповідає за отримання та обробку даних, пов'язаних з конкретною темою, введеною користувачем. Спочатку вона отримує тему і діапазон дат з GET-запиту, а потім викликає функцію reddit_api.get_posts_by_topic(), щоб отримати дописи і коментарі, пов'язані з заданою темою і діапазоном дат. Потім функція передає отримані дані різним функціям, таким як spacy_utils.get_top_entities(), pyplot_utils.get_word_cloud_data(), pyplot_utils.posts_comments_per_day_chart() і spacy_utils.get_top_phrases(), щоб витягти і обробити відповідну інформацію, таку як топ-об'єкти, щоденні підрахунки дописів і коментарів, аналіз настроїв з плином часу і тд. Нарешті, функція рендерить шаблон search_results.html з обробленими даними.

2. Модуль pyplot_utils.py

Модуль pyplot_utils.py надає кілька утиліт для побудови графіків і візуалізацій за допомогою популярних бібліотек Python, таких як matplotlib і plotly. Ці функції призначені для роботи з даними, отриманими та обробленими іншими компонентами проекту.

Функція posts_comments_per_day_chart() та генерує гістограму, яка показує кількість постів і коментарів за день для заданої теми в межах заданого діапазону дат. Ця функція використовує дані з моделей PostByTopic і PBTComment, які є частиною програми data_extraction.

Функція get_sentiment_over_time() генерує діаграму розсіювання, що показує результати аналізу настроїв для заданої теми за вказаний діапазон дат. Оцінки аналізу настроїв обчислюються за допомогою модуля spacy_utils, який використовує бібліотеку обробки природної мови spaCy.

3. Модуль settings.py

Модуль "settings.py" є частиною кожного будь-якого проекту Django, в тому числі і проекту "reddit_topic_analyzer". Цей модуль містить різноманітні конфігурації та налаштування для проекту, такі як налаштування бази даних, встановлені програми, проміжне програмне забезпечення, шаблони, статичні файли тощо.

Проект "reddit_topic_analyzer" використовує PostgreSQL як бекенд бази даних за замовчуванням. Налаштування бази даних вказано у словнику "DATABASES", який містить рушій бази даних, ім'я, користувача, пароль, хост і порт. Ці параметри мають вирішальне значення для встановлення з'єднання з базою даних та виконання операцій над нею Рис. 3.8.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'pytestdb',
        'USER': 'postgres',
        'PASSWORD': 'admin',
        'HOST': 'localhost',
        'PORT': '5432'
    }
}
```

Рис. 3.8. Змінна DATABASES

Змінна INSTALLED_APPS - це список усіх додатків, які встановлено у проекті. Сюди входять додатки Django за замовчуванням, такі як admin, auth, contenttypes, sessions і messages, а також сторонні додатки, які були встановлені. У цьому проекті також встановлено два користувацьких додатки: data_extraction та data_processing, а також сторонній додаток bootstrap5, який забезпечує інтеграцію Bootstrap 5 для проектів Django. Порядок програм у списку визначає порядок їхнього завантаження, тому важливо переконатися, що програми перелічено у правильному порядку Рис. 3.9.

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'data_extraction',
    'data_processing',
    'bootstrap5',
]
```

Рис. 3.9. Змінна INSTALLED_APPS

Проект також включає різні класи проміжного програмного забезпечення, які обробляють вхідні запити та вихідні відповіді. Ці класи проміжного програмного забезпечення визначені у списку "MIDDLEWARE" і включають проміжне програмне забезпечення безпеки, проміжне програмне забезпечення сеансу, проміжне програмне забезпечення автентифікації тощо. Ці класи проміжного програмного забезпечення необхідні для застосування політик безпеки, обробки сеансів і автентифікації користувачів Рис. 3.10.

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

Рис. 3.10. Змінна MIDDLEWARE

Список "TEMPLATES" містить різноманітні параметри конфігурації шаблонів проекту, зокрема каталоги для пошуку шаблонів, контекстні процесори та механізми шаблонів. У проекті також встановлено додаток "bootstrap5", який надає набір готових компонентів інтерфейсу користувача і шаблонів Рис. 3.11.

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

Рис. 3.11. Змінна TEMPLATES

У файлі налаштувань Django список "STATICFILES_DIRS" визначає директорії, в яких веб-додаток буде шукати статичні файли, включаючи CSS, JavaScript і файли зображень Рис 3.12. Ці статичні файли мають вирішальне значення для створення привабливого користувацького інтерфейсу та покращення загального користувацького досвіду. "STATIC_URL" визначає базову URL-адресу

для обслуговування статичних файлів. Разом ці налаштування дозволяють Django ефективно обслуговувати статичні ресурси у браузері користувача.

```
STATIC_URL = 'static/'
STATICFILES_DIRS = [BASE_DIR / "static"]
```

Рис. 3.12 Змінна INSTALLED_APPS

Загалом, модуль "settings.py" містить різноманітні конфігурації та налаштування, необхідні для належного функціонування проекту "reddit_topic_analyzer". Ці налаштування мають вирішальне значення для встановлення з'єднань з базами даних, забезпечення дотримання політик безпеки, обробки автентифікації користувачів і надання зручного інтерфейсу.

3.2 Графічний інтерфейс

Графічний інтерфейс – це тип інтерфейсу, який дає змогу користувачам взаємодіяти з електронними пристроями через графічні зображення та візуальні вказівки. Його мета полягає у забезпеченні ефективного, простого та інтуїтивно зрозумілого шляху взаємодії користувача та програмного продукту.

Якісний графічний інтерфейс є важливою частиною будь-якого веб-сайту, оскільки він на пряму поліпшує користувацький досвід. Користувацький досвід - це сукупність досвіду взаємодії людини з продуктом, послугою або системою. Він охоплює всю взаємодію між користувачем і системою від початку до кінця[17]. Поганий користувацький досвід може призвести до зниження задоволеності, лояльності і навіть відмови від продукту чи послуги.

При розробці графічного інтерфейсу необхідно враховувати багато різних аспектів. Однак одним з найважливіших факторів є забезпечення простоти його використання. Якщо у користувачів виникають труднощі з розумінням або

використанням інтерфейсу, вони, швидше за все, відмовляться від нього в пошуках простішого варіанту.

Тому головною ціллю при розробці графічно інтерфейсу для автоматизованої системи парсингу даних у вигляді веб-сайту було досягнення максимальної простоти та зручності у використанні.

3.2.1 Дизайн інтерфейсу та його загальний вигляд

Перша річ яку потрібно враховувати при дизайні інтерфейсу це аудиторія та ціль з якою буде використовуватись програмний продукт. Враховуючи, те що проект являє собою систему аналізу певних тем чи брендів на платформи Reddit, то потенційними групами користувачів будуть особи, наприклад, які збирають та аналізують інформацію про присутність та сприйнятті певних брендів чи тем в мережі Інтернет. Також зацікавленими в системі можуть бути різноманітні професіонали чи ентузіасти у сфері аналітики даних разом з звичайними користувачами платформи Reddit, які хочуть дослідити популярність та настрою людей до певних брендів чи тем.

Отже, інтерфейс веб-сайту повинен бути максимально простим, бей зайвих відволікаючих елементів та використовувати комфортну гаму кольорів.

Кольори відіграють недооцінену роль у тому як людина сприймає навколишній світ. Від темних похмурих хмар, що вказують на можливу зливу, до червоного та зеленого кольорів які повідомляють учасникам дорожнього руху про можливість руху. Це все також стосується і дизайну графічного інтерфейсу. Колірна гама може сильно впливати на сприйняття програмного продукту, наприклад яскраві кольори такі як червоний будуть негативно сприяти на користувача, якщо він вимушений довгий час користуватись програмою.

Колірна гама - це поєднання кольорів, що використовуються в різних дисциплінах дизайну, від образотворчого мистецтва до графічного дизайну. Кожен

колір чи їх поєднання має певне значення, чи викликає певні емоції в користувачів, і не правильне поєднання кольорів може коштувати набагато більше чим час затрачений на їх вибір.

Тому для графічного інтерфейсу було обрано використовувати таку гаму кольорів Рис. 3.13:

- Для бекграунду – звичайний білий колір #FFFFFF
- Для елементів інтерфейсу та для виділення полів аналізу – Aqua Spring колір #E7F7F3

Рис. 3.13 Кольорова гама веб-сайту

Щодо структури графічного інтерфейсу системи, було вирішено дотримуватись максимально простого та мінімалістичного дизайну. Веб-сайт складається з двох основних сторінок:

- main_page – являє собою головну сторінку веб-сайту. На ній розташований аналіз найпопулярніших дописів за день на платформі Reddit. Аналіз складається з текстової статистики та діаграм які у зручному вигляді подають інформацію користувачеві. Вище аналізу у верхній частині сторінки розташований її головний елемент –

пошукове поле, яке дозволяє отримати аналіз щодо заданої теми або бренду. Рис. 3.14.

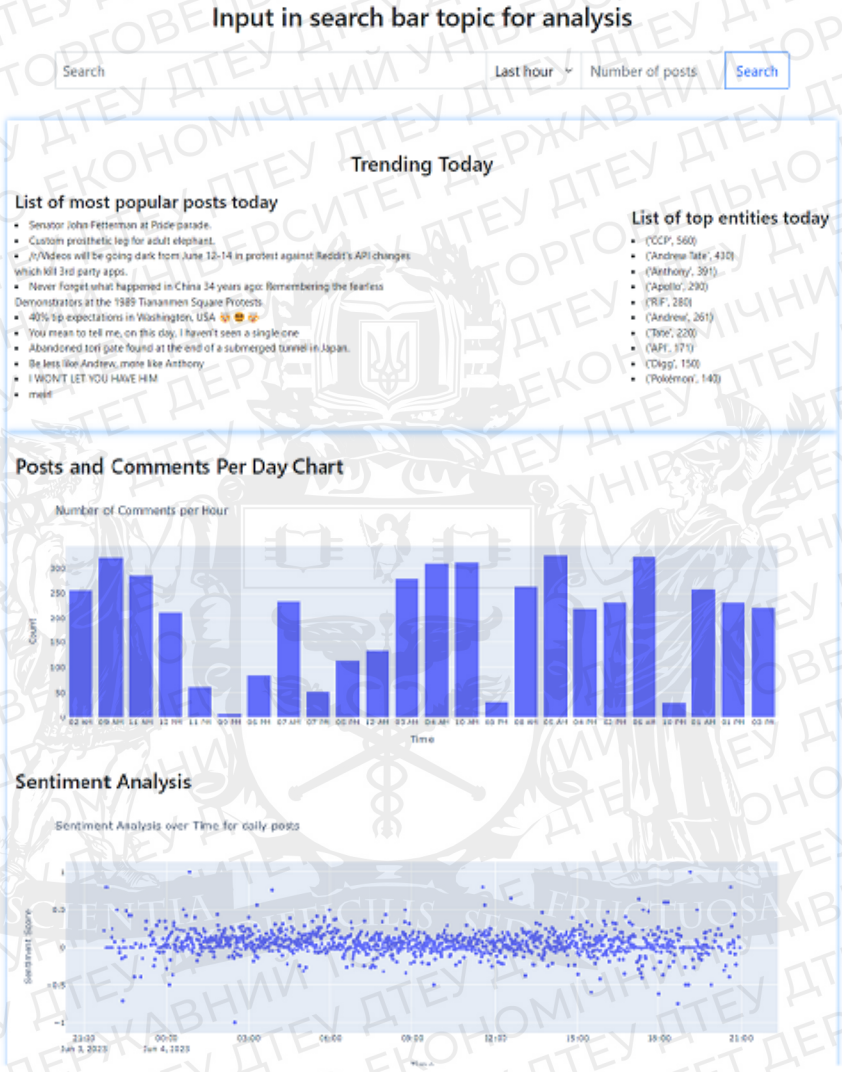

Рис. 3.14 Головна сторінка

- search_result_page – виконує головну ціль системи, видаючи аналіз щодо заданого бренду чи теми, сторінка являє собою текстовий та візуальний аналіз з використанням різноманітних діаграм. Рис. 3.15.

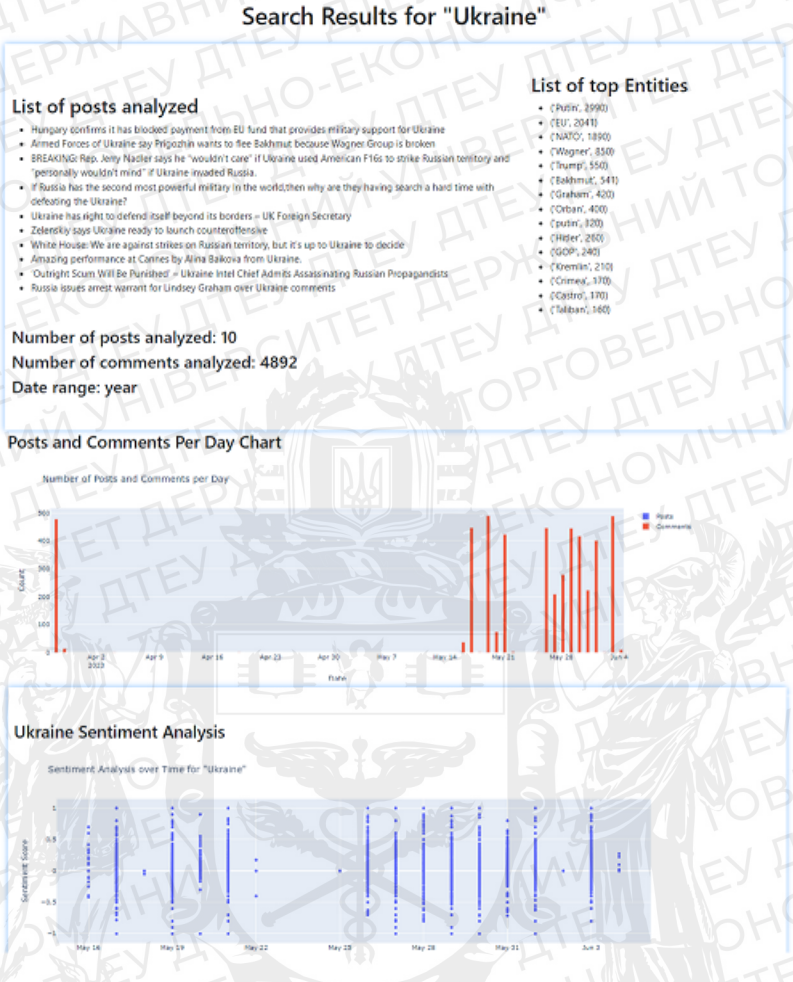

Рис. 3.15. Сторінка результатів

В нижній частині сторінки результатів представлено аналіз тональності тексту для всіх постів загалом та для кожного поста окремо Рис. 3.1.6

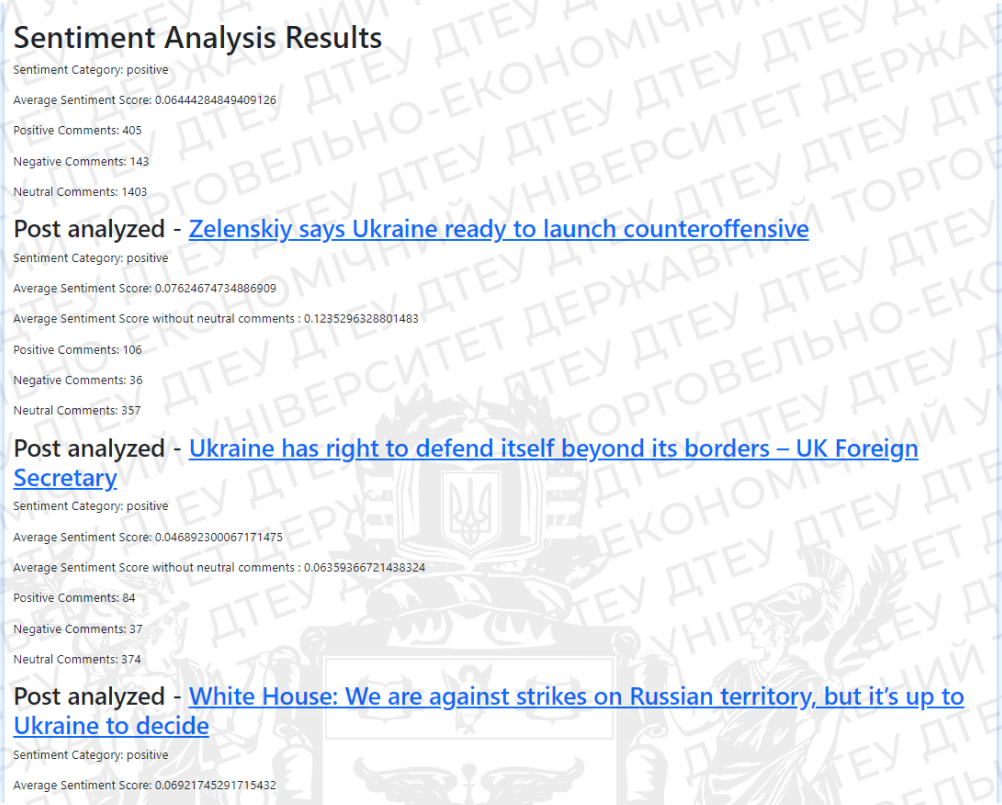

Рис. 3.16. Аналіз тональності тексту

3.2.2 Програмна реалізація вебсторінок

Веб фреймворк Django для динамічної генерації HTML сторінок використовує шаблонізаторами. Шаблон містить у собі статичні частини самого HTML, а також деякий спеціальний синтаксис, який відповідає за динамічну генерацію вмісту сторінки.

Проект Django може бути сконфігурований з одним або декількома шаблонізаторами (або навіть нульовим, якщо ви не використовуєте шаблони). Django постачається з вбудованим бекендом для власної системи шаблонів, яку називають Django Template Language (DTL)

Файли сторінок проекту містяться у папці templates Рис. 3.17.

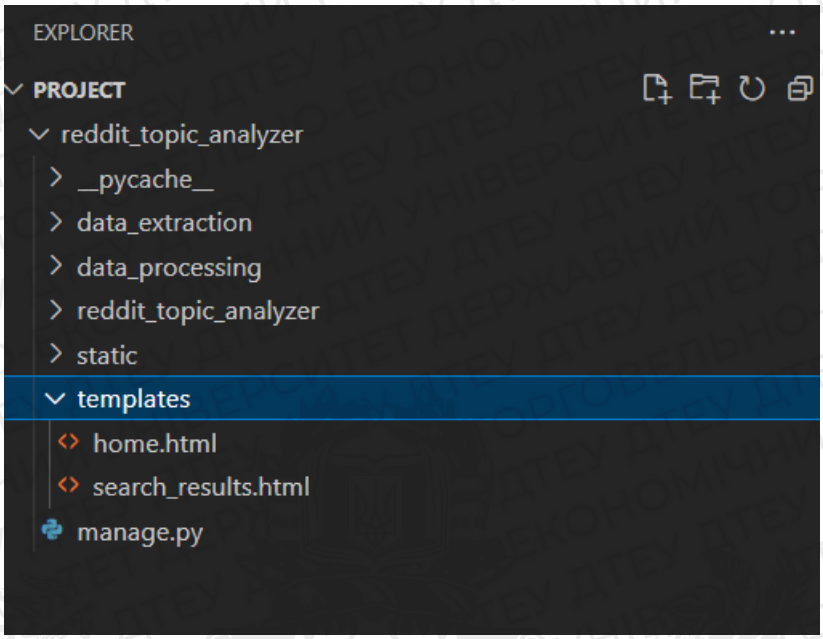

Рис. 3.17 Папка templates у директорії проекту

1. Сторінка home.html

Обидві сторінки починаються з завантаження фреймворку Bootstrap 5 Рис. 3.18. Bootstrap - це безкоштовний CSS-фреймворк з відкритим вихідним кодом, призначений для адаптивної, орієнтованої на мобільні пристрої веб-розробки[18]. Вибір Bootstrap замість чистого CSS обумовлений укороченням часу розробки проекту, узгодження стилів, уникнення написання JavaScript коду та можливість використання SASS або SCSS.

```
<html>
    <head>
        <title>Brand Analysis App</title>
        {% load bootstrap5 %}
        {% bootstrap_css %}
        {% bootstrap_javascript %}
    </head>
```

Рис. 3.18 Завантаження бібліотек Bootstrap 5

Основним компонентом головної сторінки є пошуковий рядок Рис. 3.19. Елементи панелі пошуку обгорнені в контейнер div з декількома Bootstrap класами, такими як: d-flex – перетворю контейнер в гнучкий, justify-content-center та align-

items-center - центрує вміст по горизонталі та вертикалі, flex-column – вертикально розташовує вміст.

Перший елемент в даному контейнері це заголовок <h1> з текстом " Input in search bar topic for analysis". Він відцентрований по горизонталі за допомогою класу text-center.

Другий елемент – це сам пошуковий рядок. Він являє собою елемент form, який складається з трьох компонентів:

- Два елементи input.

Перший елемент є самим пошуковим рядком, а другий елемент є полем вводу кількості постів для аналізу.

- Елемент select

Цей елемент створює випадаюче меню і складається з групи під-елементів option, кожен з який відповідає певному часовому діапазону.

- Елемент button

Цей клас являє собою звичайну кнопку для запуску пошуку. Для її дизайну використовуються набір класів btn та btn-outline-primary.

```
<div class="container d-flex justify-content-center align-items-center flex-column py-5">
    <div class="mb-4">
        <h1 class="text-center">Input in search bar topic for analysis</h1>
    </div>
    <div>
        <form class="d-flex" action="{% url 'search_results' %}" method="GET">
            <div class="input-group w-auto">
            <input type="search" name="topic" class="form-control rounded-start" placeholder="Search" aria-label="Search"
            aria-describedby="search-addon" style="width: 675px; font-size: 1.4rem; padding: 0.7rem;">
            <select class="form-select rounded-end" name="date_range" style="width: 150px; font-size: 1.4rem;">
                <option value="hour">Last hour</option>
                <option value="day">Last day</option>
                <option value="week">Last week</option>
                <option value="month">Last month</option>
                <option value="year">Last year</option>
                <option value="all">All time</option>
            </select>
            <input type="number" class="form-control rounded-end" name="post_number" style="width: 225px; font-size: 1.4rem;" placeholder="Number of posts" min="1">
            <button class="btn btn-outline-primary" type="submit" style="font-size: 1.4rem; padding: 0.7rem 1rem;">Search</button>
            </div>
        </form>
    </div>
</div>
```

Рис. 3.19 Код пошукового рядка

Наступним елементом після пошукового рядка, є поле з текстовими даними, де відображається імена оброблених дописів, та найбільш часті власні назви, які використовують користувачі форуму під даними постами. Рис. 3.20.

Досягається це шляхом створення основного контейнера div, з атрибутом стилю, який задає тінь певного кольору, для всього елементу. Всередині розташовано інший елемент div у вигляді рядка, та два дочірні div елементи з класом «col», у яких за допомогою DTL динамічно виводиться текстовий аналіз.

```
<div class="container py-5" style="box-shadow: 0 0 10px rgba(0, 123, 255, 0.5);">
    <h2 class="text-center mb-4">Trending Today</h2>
    <div class="row justify-content-between align-items-center">
        <div class="col-md-6">
            <div>
                <h3>List of most popular posts today</h3>
                {% for x in posts %}
                <li>{{ x.title }}</li>
                {% endfor %}
            </div>
        </div>
        <div class="col-md-6">
            <div class="d-flex justify-content-end">
                <div class="col-md-6"></div>
                <div>
                    <h3>List of top entities today</h3>
                    {% for x in top_entities %}
                    <li>{{ x }}</li>
                    {% endfor %}
                </div>
            </div>
        </div>
    </div>
</div>
```

Рис. 3.20 Код поля текстового аналізу

Фінальними код сторінки являє собою ще один контейнер, який вміщує собі заголовки та DTL код для генерації діаграм Рис. 3.21.

```
<div class="container py-5" style="box-shadow: 0 0 10px rgba(0, 123, 255, 0.5);">
    {% block content %}
        <h2>Posts and Comments Per Day Chart</h2>
        {{ plot_div|safe }}
        <h2>{{ topic }} Sentiment Analysis</h2>
        {{ chart|safe }}
    {% endblock %}
</div>
```

Рис. 3.21 Код для генерації діаграм

2. Сторінка search_results.html

Сторінка результатів аналізу, містить майже однаковий код для генерації діаграм, єдина відмінність це наявність додатковою статистики, яка відображає

кількість аналізованих дописів, коментарів, та заданий часовий діапазон та статистики по тональності тексту коментарів аналізованих постів Рис. 3.22.

```
<div class="container py-5" style="box-shadow: 0 0 10px rgba(0, 123, 255, 0.5);">
  <h2>{{ topic }} Sentiment Analysis</h2>
  {{ chart|safe }}

  <h1>Sentiment Analysis Results</h1>
  <p>Sentiment Category: {{ sentiment_results.sentiment_category }}</p>
  <p>Average Sentiment Score: {{ sentiment_results.average_score }}</p>
  <p>Average Sentiment Score without neutral comments : {{ result.average_score_2 }}</p>
  <p>Positive Comments: {{ sentiment_results.positive_count }}</p>
  <p>Negative Comments: {{ sentiment_results.negative_count }}</p>
  <p>Neutral Comments: {{ sentiment_results.neutral_count }}</p>

  {% for result in sentiment_results_posts %}
    <h2>Post analyzed - <a href="{{ result.post_url }}">{{ result.post_title }}</a></h2>
    <p>Sentiment Category: {{ result.sentiment_category }}</p>
    <p>Average Sentiment Score: {{ result.average_score }}</p>
    <p>Average Sentiment Score without neutral comments : {{ result.average_score_2 }}</p>
    <p>Positive Comments: {{ result.positive_count }}</p>
    <p>Negative Comments: {{ result.negative_count }}</p>
    <p>Neutral Comments: {{ result.neutral_count }}</p>
  {% endfor %}
</div>
```

Рис. 3.22 Додаткова статистка

3.3 Тестування системи

По мірі зростання програмного продукту, ручне тестування все більше і більше ускладняється. Окрім появи нових компонентів, також збільшується комплексність їхньої взаємодії між собою. Усе це призводить до необхідності в автоматизації тестування. Перевага автоматизованого тестування полягає у тому, що вона набагато швидше за ручне, стабільно виконує задані задачі та має більший рівень деталізації[19].

Існує безліч підходів до тестування, та видів самих тестів, найбільш поширеними є:

1. Модульні тестування – тестує поведінку окремих елементів, таких як окремі модулі, класи чи функції

2. Регресійні тестування – тести, які гарантують, що раніше розроблене і протестоване програмне забезпечення все ще працює як очікувалося після внесення змін
3. Інтеграційні тестування – об’єднує окремі програмні компоненти у групи, та тестує взаємодію між ними.
4. Наскрізне тестування – оцінює весь процес роботи програмного продукту з початку до кінця.

Враховуючи розміри та складність розробленої системи парсингу даних, доцільно буде застосувати три види тестування. Модульне тестування, для оцінки коректності поведінки моделей даних та окремих функцій. Інтеграційне тестування, для перевірки інтеграції між представленнями, шаблонами та даними. Та наскрізне тестування для перевірки цілісності робочого процесу всієї системи[20].

3.3.1 Модульне тестування

Модульне тестування має ціль перевірити коректність роботи моделей даних та функцій модулів системи.

Тестування моделей даних включає у себе, тести для перевірки значення полів даних, валідаційні тести та тести CRUD операцій, ці тести можна побачити на прикладі тесту «DailyBestPostTestCase» який відповідає за перевірку моделі «DailyBestPost» з додатка «data_processing». Рис. 3.23.

```
> class DailyBestPostTestCase(TransactionTestCase):···
```

Рис. 3.23 Клас «DailyBestPostTestCase»

Тести на прикладі моделі даних «DailyBestPost»:

- Тестування полів даних для перевірки коректності значень які зберігаються у полях даних конкретної моделі. Рис. 3.24.

```python
class DailyBestPostTestCase(TransactionTestCase):
    def setUp(self):
        self.post = DailyBestPost.objects.create(
            title='Test Post',
            body='Lorem ipsum dolor sit amet.',
            score=10,
            url='https://example.com',
            created_utc=timezone.now()
        )

    def test_post_title(self):
        self.assertEqual(self.post.title, 'Test Post')

    def test_post_body(self):
        self.assertEqual(self.post.body, 'Lorem ipsum dolor sit amet.')

    def test_post_score(self):
        self.assertEqual(self.post.score, 10)

    def test_post_url(self):
        self.assertEqual(self.post.url, 'https://example.com')

    def test_post_created_utc(self):
        self.assertIsNotNone(self.post.created_utc)
```

Рис. 3.24 Тестування полів даних

- Валідаційні тести для перевірки коректності застосування обмежень та правил валідації на поля даних моделі. Рис. 3.25.

```python
def test_post_validation(self):
    with self.assertRaises(IntegrityError):
        # Перевірка відсутності обов'яхкових полів даних
        DailyBestPost.objects.create()

    with self.assertRaises(ValueError):
        # Тестування введення не коректного значення score
        DailyBestPost.objects.create(title='Test', body='Lorem ipsum', score='score', url='https://example.com')
```

Рис. 3.25 Валідаційні тести

- Тести CRUD операцій для перевірки коректного застосування операцій: CREATE, READ, UPDATE, DELETE. Рис. 3.26.

```python
def test_post_crud_operations(self):
    # CREATE
    post = DailyBestPost.objects.create(title='Test', body='Lorem ipsum', score=10, url='https://example.com')
    self.assertIsNotNone(post.id)

    # READ
    retrieved_post = DailyBestPost.objects.get(id=post.id)
    self.assertEqual(retrieved_post.title, 'Test')

    # UPDATE
    retrieved_post.title = 'Updated Test'
    retrieved_post.save()

    updated_post = DailyBestPost.objects.get(id=post.id)
    self.assertEqual(updated_post.title, 'Updated Test')

    # DELETE
    updated_post.delete()

    with self.assertRaises(DailyBestPost.DoesNotExist):
        DailyBestPost.objects.get(id=post.id)
```

Рис. 3.26 Тести CRUD операцій

3.3.2 Інтеграційне тестування

Для проведення інтеграційного тестування було створено клас «RedditDataExtractionTestCase», який тестує представлення «home» та «search_result» «views.py». Наприклад функція «test_home_view» імітує HTTP-запит до представлення «home» та перевіряє коректність повернутого вмісту Рис. 3.27.

```python
class RedditDataExtractionTestCase(TestCase):
    def setUp(self):
        # Створення фабрик запитів
        self.factory = RequestFactory()

    def test_home_view(self):
        # Створення GET запиту до "home" представлення
        request = self.factory.get('/')
        response = home(request)

        # Перевірка чи 200 (OK)
        self.assertEqual(response.status_code, 200)

        # Перевірка чи відповідь містить очікуваний конетент
        self.assertIn('posts', response.content.decode())
        self.assertIn('num_posts', response.content.decode())
        self.assertIn('num_comments', response.content.decode())
        self.assertIn('top_entities', response.content.decode())
        self.assertIn('word_cloud_data', response.content.decode())
        self.assertIn('plot_div', response.content.decode())
        self.assertIn('chart', response.content.decode())
        self.assertIn('top_phrases', response.content.decode())
```

Рис. 3.27 Тести CRUD операцій

Це тестування допомагає перевірити, що представлення у «views.py», їхня взаємодія з даними та шаблонами функціонує коректно.

3.3.3 Наскрізне тестування

Наскрізне тестування повторює поведінку користувача в системі. Для цього тест проводить імітацію дій користувача, виконує пошук і перевіряє елементи сторінки результатів аналізу, аби переконатись в коректності робочого процесу програмного продукту.

За тестування відповідає клас «MyApplicationEndToEndTest», який спочатку налаштовує Selenium WebDriver для Edge, відкриває додаток у браузері, вводить деяку тему або бренд в пошуковий рядок після чого перевіряє коректність повернутою сторінки результатів аналізу Рис. 3.28.

```
class MyApplicationEndToEndTest(LiveServerTestCase):
    @classmethod
    def setUpClass(cls):
        super().setUpClass()
        # Налашутвання Selenium WebDriver для Edge
        cls.selenium = Edge()

    @classmethod
    def tearDownClass(cls):
        # Закриття Selenium WebDriver
        cls.selenium.quit()
        super().tearDownClass()

    def test_end_to_end_flow(self):
        # Відкриття додатку в браузері
        self.selenium.get(self.live_server_url)

        # Знаходження пошукового елемента та введення topic
        search_input = self.selenium.find_element_by_name("topic")
        search_input.send_keys("python")
        search_input.send_keys(Keys.ENTER)

        # Верифікаця сторінки результату
        self.assertIn("Search Results", self.selenium.title)

        # Перевірка елементу заголовку
        header = self.selenium.find_element_by_tag_name("h1")
        self.assertEqual(header.text, 'Search Results for "python"')

        # Перевірка списку дописів
        posts_list = self.selenium.find_element_by_tag_name("ul")
        posts = posts_list.find_elements_by_tag_name("li")
        self.assertGreater(len(posts), 0)

        # Перевірка списку найчастіших власних назв
        entities_list = self.selenium.find_elements_by_xpath("//div/h2[contains(text(), 'List of top Entities')]/following-sibling::ul")
        entities = entities_list[0].find_elements_by_tag_name("li")
        self.assertGreater(len(entities), 0)
```

Рис. 3.28 Клас «MyApplicationEndToEndTest»

РЕЗУЛЬТАТИ І ВИСНОВКИ

1. Було успішно розроблено систему автоматизованого парсингу даних за допомогою мови програмування Python, система представлена у вигляді веб-сайту розробленому на веб-фреймворку Django. За допомогою цієї системи, користувачі мають можливість отримати глибокий аналіз певної теми чи бренду на платформі Reddit.
2. Розроблений програмний продукт за допомогою бібліотеки PRAW ефективно вилучає дані з платформи Reddit, зберігає їх у базі даних PostgresSql та обробляє їх, використовуючи такі бібліотеки як SpaCy та NetworkX. Готові результати надаються користувачам у вигляді аналізу з використанням діаграмі та інших ілюстрацій для більш наочного відображення статистики по конкретній темі або бренду.
3. Автоматизована система парсингу даних з платформи Reddit дозволяє ефективно збирати та аналізувати дані, шляхом автоматизації цього процесу. Користувачі позбавляються необхідності в ручному зборі та аналізі даних, заощаджуючи на цьому дорогоцінний час та зусилля.
4. Система має важливе значення для різноманітних бізнесів, адже вона дозволяє їм на основі отриманих даних розуміти настрої, дискусії та загальне відношення користувачів мережі Інтернет до певного бренду чи теми. Це розуміння у свою чергу, уможливлює прийняття більш точних та ефективних бізнес-рішень.
5. На основі висновків та результатів отримання під час та після розроблення програмного продукту, можна зробити декілька рекомендацій: інтегрувати машинне навчання, покращити графічний інтерфейс, створити системи облікових записів користувачів та додати сумісності з іншими платформами, подібними до Reddit.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Dick Grune. Parsing Techniques: A Practical Guide / Dick Grune, Ceriel J. H. Jacobs. — Springer : New York, 1998. — 326 с.
2. Christopher Manning. Foundations of Statistical Natural Language Processing / Christopher Manning, Hinrich Schutze. — MIT Press, 1999. — 720 с.
3. Harry Bunt. Trends in Parsing Technology: Dependency Parsing, Domain Adaptation, and Deep Parsing / Harry Bunt, Paola Merlo, Joakim Nivre. — Springer : Dordrecht, 2010. — 297 с.
4. Alfred V. Aho, Jeffrey D. Ullman - The Theory of Parsing, Translation, and Compiling / Alfred V. Aho, Jeffrey D. Ullman. — Parsing-Prentice-Hall, 1973. — 490 с.
5. Andrew Carnie. Syntax: A generative introduction, 3rd edition. — Malden, MA: Wiley-Blackwell, 2013. — 544 с.
6. Alfred V. Aho. Compilers: Principles, techniques, & tools / Alfred V. Aho , Sethi, Ravi; Ullman, Jeffrey D. — MA: Addison-Wesley, 2006. — 796 с.
7. Robert C. Berwick. Minimalist Parsing / Berwick Robert C. Stabler Edward P. — Robert C. Berwick, 2019. — 208 с.
8. Song-Chun Zhu. Classic Parsing Algorithms. — University of California, 2021. — 16 с.
9. Harold Chestnut. Systems Engineering Methods — New York : John Wiley & Sons, 1967. — 416 с.
10. Carlee Bishop. Systems Engineering Fundamentals. — Wayback Machine Defense Acquisition University Press, 2001. — 222 с.
11. Python 3.9.5 Documentation [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://www.python.org/doc/
12. Django Documentation [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://docs.djangoproject.com/en/4.2/

13. PostgreSQL Documentation [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://www.postgresql.org/docs/
14. SpaCy Documentation [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://spacy.io/api/doc/
15. NetworkX Documentation [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://networkx.org/documentation/stable/index.html
16. PRAW Documentation [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://praw.readthedocs.io/en/stable/index.html
17. Levy, Steven. "Graphical User Interface (GUI)". Britannica.com. Retrieved 2019-06-12.
18. Bootstrap 5 Documentation [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://getbootstrap.com/docs/
19. Testing in Django Documentation [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://docs.djangoproject.com/en/4.2/topics/testing/
20. Glenford J. Myers. The Art of Software Testing. — Wiley, 2004. — 256 с.