Національний університет «Полтавська політехніка імені Юрія Кондратюка»
(повне найменування вищого навчального закладу)
Навчально науковий інститут інформаційних технологій та робототехніки
(повна назва факультету)
Кафедра комп'ютерних та інформаційних технологій і систем
(повна назва кафедри)

Пояснювальна записка
до дипломного проекту (роботи)
магістра
(освітньо-кваліфікаційний рівень)
на тему

Розробка пошукової системи для підтримки прийняття рішень

Виконав: студент 6 курсу, групи 601 ТН
спеціальності
122 Комп'ютерні науки
(шифр і назва напряму)
Пронь В.О.
(прізвище та ініціали)
Керівник Миронцов М.Л.
(прізвище та ініціали)
Рецензент _____
(прізвище та ініціали)

Полтава – 2024 року

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
«ПОЛТАВСЬКА ПОЛІТЕХНІКА ІМЕНІ ЮРІЯ КОНДРАТЮКА»

НАВЧАЛЬНО НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА РОБОТОТЕХНІКИ

КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

спеціальність 122 «Комп'ютерні науки»

на тему

«Розробка пошукової системи для підтримки прийняття рішень»

Студента групи 601 ТН Проня Владислава Олеговича

Керівник роботи
д.ф.-м.н., проф.
Миронцов М.Л.

Завідувач кафедри
кандидат ф.-м. наук,
Двірна О.А.

Полтава – 2024

РЕФЕРАТ

Кваліфікаційна робота магістра: 70 с., 11 рисунків, 8 таблиць, 2 додатки, 46 джерел.

Об'єкт дослідження: процес розробки інформаційно-пошукової системи для підтримки прийняття рішень.

Мета роботи: розробка автоматизованої інформаційно-пошукової системи, що використовує алгоритми штучного інтелекту та обробку великих даних для прийняття обґрунтованих рішень у різних сферах.

Методи: аналіз сучасних підходів до проектування пошукових систем, застосування алгоритмів машинного навчання, інтеграція чат-ботів для автоматизації пошукових процесів, використання Python для розробки серверної частини та обробки даних, JavaScript та HTML для створення веб-інтерфейсу.

Ключові слова: пошукові системи, прийняття рішень, Python, JavaScript, HTML, чат-боти, обробка даних.

SUMMARY

Master's qualification work: 70 p., 11 pictures, 8 tables, 2 appendices, 46 sources.

Object of research: the process of developing an information-search system for decision support.

Purpose: development of an automated information-search system that uses big data processing algorithms for making informed decisions in various fields.

Methods: analysis of modern approaches to search system design, application of machine learning algorithms, integration of chatbots for automating search processes, use of Python for developing the server side and data processing, JavaScript and HTML for creating the web interface.

Keywords: search systems, decision-making, Python, JavaScript, HTML, chatbots, data processing.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ІС – Інформаційна система

ПЗ – Програмне забезпечення

БД – База даних

СУБД – Система управління базами даних

API – Application Programming Interface

HTTP – HyperText Transfer Protocol

SQL – Structured Query Language

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

JavaScript – Мова програмування для веб-розробки

Python – Мова програмування для розробки інформаційних систем

Flask – Веб-фреймворк для Python

UI – User Interface (Інтерфейс користувача)

UX – User Experience (Досвід користувача)

REST – Representational State Transfer (Архітектурний стиль для веб-сервісів)

MVC – Model-View-Controller (Архітектурний патерн)

GDPR – General Data Protection Regulation (Загальний регламент захисту даних)

ВСТУП

В умовах цифрової трансформації економіки важливим аспектом ефективної діяльності підприємств є автоматизація процесів, що підтримують прийняття рішень. Пошукові системи, здатні забезпечити швидкий доступ до необхідної інформації та даних, грають ключову роль у прийнятті обґрунтованих і обґрунтованих рішень у різних сферах бізнесу та управління. Зокрема, розробка пошукових систем для підтримки прийняття рішень на основі великих даних, штучного інтелекту та машинного навчання стає необхідністю для підвищення ефективності та точності процесів, що відбуваються на підприємствах, в організаціях та у сфері особистого розвитку.

Проектування таких систем передбачає комплексний підхід до інтеграції алгоритмів пошуку, аналізу даних та рекомендацій. Використання сучасних технологій дозволяє створювати платформи, що автоматизують процес пошуку, аналізу альтернатив та вибору оптимальних варіантів для прийняття рішення. Пошукові системи для підтримки прийняття рішень не лише надають доступ до великої кількості інформації, але й інтегрують різноманітні технології для адаптації результатів пошуку відповідно до потреб користувача.

Метою дипломної роботи є розробка інформаційно-пошукової системи для підтримки прийняття рішень, що дозволяє автоматизувати процеси обробки запитів і аналізу даних. Для досягнення цієї мети в роботі буде здійснено розробку системи, яка використовує новітні технології програмування та інтерфейси для забезпечення швидкого і точного прийняття рішень у різних галузях діяльності.

Завданнями, які розв'язуються у роботі, є:

– аналіз сучасних підходів до проектування пошукових систем для підтримки прийняття рішень;

– дослідження існуючих рішень та архітектури таких систем;

– розробка вимог до інформаційно-пошукової системи;

– проектування і реалізація пошукової системи для конкретної області застосування.

Об'єктом дипломної роботи є процес розробки інформаційно-пошукової системи для підтримки прийняття рішень.

Предметом роботи є розроблена інформаційно-пошукова система, яка включає функціональні можливості для обробки даних, пошуку інформації та надання рекомендацій на основі аналізу великих даних.

Дипломна робота складається з трьох розділів. У першому розділі розглядаються основи роботи пошукових систем в Інтернеті, їх еволюція та роль у прийнятті рішень. Також досліджуються сучасні підходи до проектування пошукових систем, що включають використання алгоритмів штучного інтелекту, машинного навчання та обробки великих даних. У другому розділі розглядається загальна постановка проектного завдання, аналізується створення вимог до інформаційно-пошукової системи та описуються технології розробки чат-ботів, які можуть бути частиною системи. У третьому розділі зосереджено увагу на практичній реалізації системи, включаючи створення серверної частини чат-бота, архітектурні рішення та реалізацію API для взаємодії з зовнішніми сервісами.

Для реалізації розробленої інформаційно-пошукової системи використовуються сучасні мови програмування та технології, такі як Python для серверної частини та обробки даних, JavaScript для інтерактивності на веб-інтерфейсі, а також HTML для створення зручних та функціональних веб-сторінок.

РОЗДІЛ 1
АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ДО ПРОЕКТУВАННЯ ПОШУКОВИХ СИСТЕМИ ДЛЯ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ

Основи роботи пошукових систем в Інтернеті

Обсяги інформації, необхідні для прогресу сучасного суспільства, є надзвичайно великими і продовжують зростати в експоненціальній прогресії. Ще однією особливістю сучасної інформації є відсутність сталих джерел інформації. Традиційні джерела інформації, такі як бібліотеки, архіви і бази даних, вже не сприймаються як окремі інформаційні точки, а розглядаються як частина великої мережі джерел. Найбільше явною ця тенденція розпорошення інформації є в нових цифрових середовищах, наприклад, у глобальних комп'ютерних мережах.

Розпорошення джерел інформації дає можливість доступу до необхідних даних, але також створює низку серйозних проблем, зокрема, пошук та класифікація потрібних ресурсів. Інтернет, як глобальне інформаційне середовище, містить мільйони джерел, доступних для загального користування і охоплює всі можливі теми. Труднощі з орієнтацією у цьому величезному масиві інформації не зумовлені лише його обсягом чи різноманіттям форматів, а й динамічністю змін, яка вимагає постійного оновлення даних про доступність і розташування інформаційних ресурсів. Для повноцінної участі в житті сучасного інформаційного суспільства людина звертається до різних джерел або сховищ знань, що доступні в Інтернеті, де можна знайти відповіді на численні питання. Проте ефективне використання цих інформаційних ресурсів неможливе без наявності потужних пошукових систем. Застосування сучасних пошукових механізмів дозволяє швидко і точно знаходити необхідну інформацію. Яскравим прикладом є популярна пошукова система Google, яка вже давно здобула визнання завдяки своїм потужним функціям і високій надійності.

Розподіленість джерел інформації забезпечує доступ до великої кількості даних, але одночасно створює значні труднощі, зокрема у пошуку і

систематизації необхідних матеріалів. Інтернет, як глобальна інформаційна платформа, містить мільйони відкритих джерел, що охоплюють всі можливі теми. Складність орієнтування у цьому безкрайньому інформаційному просторі не лише в його масштабах або різноманітті форматів даних, але й у постійних змінах, що вимагають регулярного оновлення інформації про доступність і розташування необхідних ресурсів.

Для ефективного використання таких середовищ, як глобальна мережа Інтернет, необхідно застосовувати потужні пошукові механізми, зокрема інформаційні пошукові системи (ІПС).

У наукових працях знаходимо такі визначення інформаційно-пошукової системи:

1) Інформаційно-пошукова система (ІПС) – програмна система для збереження, пошуку і видачі інформації, необхідної користувачу [22].

2) Інформаційно-пошукові системи – автоматизовані системи, призначені для зібрання, пошуку, оброблення, збереження та видачі інформації за допомогою технічних засобів [24].

3) Інформаційно-пошукові системи – це різновид автоматизованих інформаційних систем, в яких завершальна обробка даних не передбачається, натомість ці системи призначені для пошуку текстів (документів, їх частин, фактографічних записів) в сховищах (базах даних) за формальними характеристиками [25].

4) Інформаційно-пошукова система (ІПС) – це сукупність довідково-інформаційного фонду (сукупність інформаційних масивів і пов’язаного з ними довідково-пошукового апарату) і технічних засобів інформаційного пошуку в ньому [21]

Є кілька загальних аспектів, які характерні для різних інформаційних систем, а саме: «призначення для збору, обробки та запису інформації, що свідчить про те, що основою функціоналу є база даних; орієнтація на кінцевих користувачів, що спрямоване на налаштування простого, легкого та зручного інтерфейсу [22]».

Окремо зосередимо увагу на понятті «механізм пошуку» в ІПС – «це система, що забезпечує пошук і відбір необхідних даних у спеціальній базі з описами джерел інформації (індексі) на основі інформаційно-пошукової мови і відповідних правил пошуку [26]».

Пошукові системи включають три основні компоненти:

- веб-сторінка з пошуковим механізмом, яка виконує роль інтерфейсу для організації взаємодії з базою даних;

- база даних, де міститься інформація, що зібрана спеціальними програмами пошукової системи. Власне наявністю баз даних пояснюється висока швидкість виведення результатів пошуку на сторінку пошукової системи;

- пошукові роботи (Robots), павуки (Spiders) або хробаки (Worms) - спеціальні програми, які автоматично періодично «відвідують» сайти, збирають відомості про вміст сторінок, тобто індексують їх і наповнюють бази даних пошукової системи.

Архітектура інформаційно-пошукової системи представлена на рисунку 1.1.

Логіко-семантичний апарат ІПС складається з трьох основних блоків:

- інформаційно-пошукової мови (ІПМ);

- системи індексування (перекладу на інформаційно-пошукову мову);

- логіки, що забезпечують пошук, які в свою чергу можуть бути деталізовані та реалізовані різними способами.

Сканування та збір даних. Веб-кроулер – це програма, яка за допомогою однієї або кількох початкових URL-адрес завантажує веб-сторінки, пов’язані з цими URL-адресами, витягує будь-які гіперпосилання, що містяться в них, і рекурсивно продовжує завантажувати веб-сторінки за цими гіперпосиланнями [8]. Веб-сканер виявляє і додає вміст веб-сторінок до сховища даних пошукової системи. Типи зібраного вмісту включають текстові сторінки, зображення, відео та інші типи контенту. Більшість Веб-сканерів знаходять інформацію, починаючи з однієї сторінки та переходячи за вихідними URL-посиланнями на

ції сторінці. Тому, якщо сторінка не має посилання з іншої сторінки, вона може так і не бути знайдена пошуковою системою.

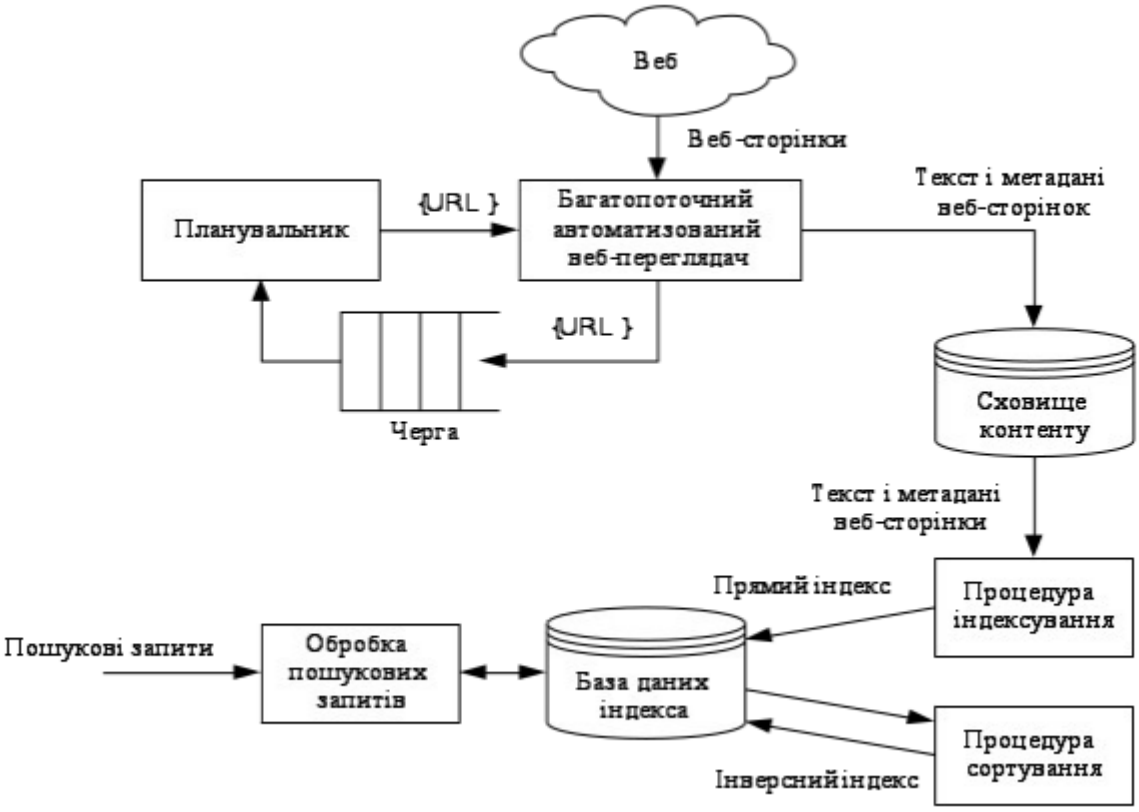

Рисунок 1.1 – Архітектура інформаційно-пошукової системи [38]

У деяких випадках пошукова система може отримувати контент через приватні потоки даних від інших компаній. Уся інформація, яку отримує веб-сканер, зберігається в сховищі даних, яке є основою для створення онлайн-індексу, доступного для пошуку користувачами. Веб-сканер є частиною пошукової системи для збору даних з Інтернету; він може розпізнавати та збирати сторінки HTML та інші типи документів, включаючи PDF, PowerPoint, Word і Excel. Він витягує текстову інформацію з цих документів за допомогою програмного забезпечення для перетворення, щоб відповідну текстову інформацію можна було індексувати та шукати.

Сховище даних та обробка офлайн. Деякі пошукові системи мають надзвичайно великі бази даних з мільярдами сторінок, тоді як інші мають порівняно невеликі. Пошукова система може агрегувати результати як зі своєї власної бази даних, так і з інших пошукових систем, що знаходяться в різних

локаціях, щоб розширити охоплення бази даних. Офлайн-обробка зазвичай полягає в зборі сторінок та побудові відповідної структури даних для пошуку. Офлайн-система також виконує виявлення дублікатів і спаму, а також проводить додатковий аналіз даних для визначення властивостей сторінки як сигналів ранжування.

Онлайн-пошукова система та ранжування. Пошукова система працює як веб-сервіс для відповіді на запити користувачів. Система обробки запитів використовує індекс для знаходження релевантних документів і їхнього ранжування. Вона генерує сторінку результатів, що містить найкраще оцінені документи, і подає її користувачу. Коли користувачі взаємодіють з результатами пошуку, пошукова система реєструє їхню поведінку під час перегляду і використовує ці дані для покращення якості пошуку.

Робот-індексатор (“павук”). Це «програма», завдання якої читати вміст сторінки, аналізувати її вміст і HTML-код. Павук переходить за посиланнями, розміщеними на сайті (вони є як внутрішніми, так і зовнішніми посиланнями) і з’являється як на абсолютно нових, так і на старих сторінках. Сторінка, яку ще не відвідав робот, не індексується – тобто її немає в результатах пошуку.

Покажчик — це список заголовків і коротких уривків з посиланнями на сторінки, де є інформація, що відповідає нашому запиту.

База даних — це сукупність усіх вебсайтів, проіндексованих роботами. Під час відвідування вебсайтів роботи індексації створюють величезну базу даних, яка постійно оновлюється, оскільки з’являються нові сторінки, а ці, в свою чергу, оновлюються новими підсторінками.

Опишемо основні характеристики ІПС [38].

Повнота — це одна з основних характеристик пошукової системи, яка відображає співвідношення між кількістю знайдених за запитом документів і загальною кількістю документів в базі даних, що відповідають запиту.

Точність — ще одна ключова характеристика пошукової системи, яка визначається рівнем відповідності знайдених документів конкретному запиту користувача. Чим точніший пошук, тим швидше користувач отримає необхідні

результати, і тим менше він зіткнеться з нерелевантними чи непотрібними документами.

Актуальність — важливий аспект пошукових систем, що визначає час, що минув від публікації документа в Інтернеті до його внесення в індекс пошукової системи. Великі пошукові системи використовують так звані «швидкі бази», які оновлюються кілька разів на день, що дозволяє індексувати найбільш важливі документи та робити їх доступними для пошуку.

Швидкість пошуку безпосередньо залежить від стійкості системи до високих навантажень. Велика кількість одночасних запитів вимагає скорочення часу обробки кожного окремого запиту. Тут інтереси користувача і пошукової системи збігаються: відвідувач хоче отримати результати якомога швидше, а пошукова система повинна ефективно обробляти запити, не знижуючи продуктивність при великих навантаженнях.

Зручність представлення результатів пошуку — ще один важливий фактор для користувача. Оскільки зазвичай пошукові системи надають сотні або навіть тисячі документів по запиту, важливо, щоб інформація була структурована та представлена чітко. Іноді через неточність запитів або обмеження пошукових алгоритмів перші сторінки результатів можуть містити не тільки потрібну інформацію, що змушує користувача проводити додатковий пошук всередині знайдених документів. Ось чому правильне форматування і організація результатів пошуку відіграють важливу роль у зручності користування системою.

В основі сучасних пошукових систем закладено також і алгоритми ранжування, які можуть ранжувати результати пошуку різними способами. Векторне ранжування, BM25 і семантичне ранжування — це всі методи, які використовуються в пошуку інформації та пошукових системах для ранжування та отримання документів або частин вмісту на основі їх відповідності запиту.

Кожен із цих методів представляє окрему парадигму способу визначення релевантності пошуку.

BM25, традиційний і широко використовуваний алгоритм, відмінно підходить у сценаріях, де відповідність ключових слів і простота є першочерговими. BM25 — це ймовірнісна функція ранжирування, яка є частиною сімейства моделей пошуку «сумка слів». Він розраховує релевантність документа для запиту, враховуючи такі фактори, як частота термінів (як часто термін з'являється в документі), зворотна частота документа (наскільки поширений або рідкісний термін у всіх документах) і нормалізація довжини документа.

Vector Ranking (векторне ранжування) , використовуючи геометричні зв'язки між словами у багатовимірному просторі, пропонує більш тонкий підхід до подібності документів. Векторне ранжування стосується використання векторних просторових моделей (VSM) для пошуку інформації. У цьому методі і документи, і запити представлені як вектори в багатовимірному просторі. Відповідність документа запиту визначається косинусною подібністю між їхніми векторами.

Семантичне ранжування — це більш досконалий метод, який враховує значення слів і фраз, а не лише їх дослівне входження в текст. Він часто використовує моделі глибокого навчання, такі як BERT (Bidirectional Encoder Representations from Transformers) або інші моделі NLP, які можуть зрозуміти контекст і семантику Семантичне ранжування ґрунтується на останніх досягненнях у обробці природної мови та прагне зрозуміти глибше значення запитів, що робить його незамінним для складних контекстно-насичених завдань пошуку.

Розуміння цих методів ранжирування має важливе значення для будь-кого, хто займається розробкою або оптимізацією систем пошуку та пошуку.

Підходи до проектування пошукових систем

При проектуванні ІПС небхідно розуміти їх види. Пошукові системи можна поділити на тематичні (класифікатори) та індексні системи.

Тематичні системи пропонують користувачам список категорій, де веб-сторінки організовано за ієрархічним принципом. Переміщуючись по дереву категорій, можна поступово звужувати область пошуку і отримувати перелік посилань на веб-сайти, що відповідають заданій темі. Для ефективного використання таких систем потрібно точно знати, до якої категорії належить необхідна інформація. Важливо також врахувати, що класифікація здійснюється людиною, тому вона може бути суб'єктивною. До тематичних пошукових систем належать такі ресурси, як: www.mckinley.com, www.yahoo.com, www.list.ru.

Індексні системи виконують пошук за заданими ключовими словами. Після пошуку система генерує список сайтів, що відповідають критеріям запиту, які можуть бути представлені одним словом, набором слів або логічним виразом. Для підвищення ефективності своєї роботи індексні системи використовують такі компоненти:

- індексатор, який періодично сканує Інтернет, збираючи інформацію про ресурси;
- індекс — масив даних, який використовується для пошуку відповідних ресурсів. Індекс складається з прямого та інвертованого списків, що встановлюють відповідність між термінами пошуку та документами, які їх містять;
- інтерфейс пошуку — це інформаційно-пошукова мова системи, яка допомагає користувачу формулювати запити, а також методи виконання запитів у базі даних.

Для оптимізації словників та індексів у цих системах застосовується поняття «вага терміна», що визначається під час індексування і залежить від обраного методу індексації.

Методи індексування можна класифікувати на три основні категорії [40]:

1. Статистичні методи — документи розглядаються як точки в інформаційному просторі. Чим ближче розташовані терміни, тим ближче знаходяться й відповідні точки. Для індексації вибираються терміни, які зменшують щільність простору документів.

2. Теоретико-інформаційні методи — базуються на принципі, що найбільшу цінність мають слова, що зустрічаються рідко. Оцінка термінів здійснюється за допомогою методів теорії інформації.

3. Ймовірнісні методи — використовують множину документів для оцінки релевантності результатів. Ці методи розраховують ймовірність присутності терміна в конкретному документі на основі його релевантності до запиту.

Формальна релевантність визначається системою і використовується для ранжування результатів пошуку. Натомість реальна релевантність оцінюється самим користувачем на основі корисності знайдених документів.

До індексних пошукових систем належать: www.excite.com, www.altavista.com, www.infoseek.com, www.lycos.com, www.rambler.ru, www.yandex.ru.

Розробка інформаційно-пошукових системм є нетривіальною задачею та включає ряд складнощів і викликів. Зокрема, веб-кроулерам доводиться стикатися з обме-женнями, накладеними веб-серверами. Деякі сервери можуть обмежувати швидкість доступу до сторінок для одного IP, що ускладнює під-тримку високої швидкості сканування. Також, сканування великих обсягів даних може при-зводити до проблем із зберіганням, обробкою та передачею цих даних мережею. Кроулери повинні ефективно масштабуватися та викори-ствовуввати паралельну обробку даних. Робота із великою кількістю задач одночасно потребує правильної архітектури та керування ресурсами.Крім того, однією з найскладніших задач є обробка динамічного контенту сайтів, згене-рованого з використанням технологій, таких як AJAX або JavaScript.

Існує низка факторів, які ще більше ускладнюють проектування інформаційно-пошукових систем [39]:

- політика сайту. Веб-сайт часто містить велику кількість сторінок. Щоб швидко отримати ці сторінки, сканер може займають значну кількість обчислювальних ресурсів і ресурсів пропускної здатності цього сайту, і це може вплинути на роботу такого сайту. Щоб пом’якшити негативний вплив на ресурси

сайту, сканер повинен стежити за політикою сайту, яка обмежує швидкість сканування. Наприклад, сканер не може робити більше одного запиту до того самого хосту одночасно, і, це може додати часову затримку між двома послідовними запитами до одного сайту;

- протокол виключення роботів. Веб-адміністратор може виключити свої сторінки з індексу пошукової системи, а саме визначаючи обмеження на відвідування для сканера пошукової системи [12]. Під час отримання сторінки із сайту веб-сканер повинен перевірити файл robots.txt із відповідного веб-хосту. Щоб уникнути повторного доступу до файлу robot.txt з того самого хосту, сканер може зберігати копію такого файлу у своїй системі локального кешу з періодичним оновленням;

- деякі сайти повільно відповідають на запит сканера або навіть не відповідають. Деякі сайти можуть повертати сканеру неправильні або несправні результати. Веб-сканер повинен бути стійким до збоїв або повільності веб-сайту та мати можливість повторити спробу пізніше. Інший тип аномалії виникає тоді, коли веб-сайт тримає пошуковий робот зайнятим, динамічно надаючи нескінченну кількість непотрібних веб-сторінок. Таку пастку можна виявити за допомогою орієнтованого на сайт аналізу подібного вмісту;

- прихована мережа та карта сайту. Роботи з основних пошукових систем можуть ефективно збирати веб-сторінки на поверхневому рівні, але існує величезна кількість прихованої глибокої веб-інформації, невидимої для пошукових систем. Будь-яка сторінка, на яку явно не вказують інші веб-сторінки, не може бути знайдена сканером за допомогою аналізу гіперпосилань. Динамічно створені сторінки, які вимагають подання HTML-форми або Javascript, часто невидимі. Автоматичне заповнення форм може досліджувати глибоку мережу [6], і все ще існує проблема досягнення високої точності чи покриття. Альтернативним методом є використання протоколу Sitemaps [13], який дозволяє веб-майстрам інформувати пошукові системи про список сторінок, які доступні для сканування з їхніх сайтів. Карта сайту – це XML-файл, який містить URL-адреси сайту разом із додатковими метаданими про кожну

URL-адресу. Таким чином, веб-майстри можуть рекламувати глибокі URL-адреси, які навіть не мають гіперпосилань із зовнішнього Інтернету.

Визначають наступні вимоги до розробки веб-сканерів [19, 27]:

1. Функціональні вимоги:

- налаштовувати вхідні URL-адреси та визначати глибину сканування;
- створювати список посилань для завантаження;
- фільтрувати URL-адреси за допомогою шаблонів, що визначають імена хостів або шляхи, які потрібно врахувати;
- одночасно завантажувати веб-сторінки за посиланнями;
- аналізувати завантажені сторінки та витягувати з них гіперпосилання;
- нормалізувати отримані посилання згідно з правилами їх перетворення;
- зберігати структуру посилань у вихідному вигляді;
- проводити статистичний аналіз за допомогою відповідних методів;
- здійснювати кластеризацію різних сегментів веб-простору.

2. Нефункціональні вимоги:

- організація процесу завантаження URL-адрес (визначення порядку їх обробки);
- політика відвідування сторінок (регулює навантаження на веб-сервер при скануванні);
- обробка веб-сторінок (знаходження гіперпосилань);
- ідентифікація дублюючих сторінок (запобігання дублюванню під час сканування);
- інтерактивність (підтримка файлів cookie та простих схем аутентифікації);
- стабільність (стійкість до можливих перерв у роботі, ведення журналу для моніторингу та діагностики проблем);
- підтримка нових протоколів завантаження для кроулера;
- висока пропускна здатність.

Роль пошукових систем у прийнятті рішень

Пошукові системи стали невід'ємною частиною нашого повсякденного життя та важливим інструментом для отримання інформації. Їхня роль значно виходить за межі простого пошуку веб-сторінок в Інтернеті. У сучасному світі, де інформація є основним ресурсом, пошукові системи відіграють критичну роль у процесах прийняття рішень як на індивідуальному, так і на організаційному рівні. Розглянемо, як саме пошукові системи впливають на ефективність прийняття рішень у різних сферах діяльності [39].

1. Доступ до релевантної інформації

Перше і найочевидніше значення пошукових систем у прийнятті рішень полягає в тому, що вони забезпечують швидкий доступ до величезних обсягів інформації. У сучасних умовах обсяг доступних даних значно зріс, і пошукові системи дозволяють користувачам знаходити необхідні матеріали з різноманітних джерел — наукових статей, новин, економічних звітів, технічної документації тощо. Таким чином, ефективність прийняття рішень залежить від того, наскільки оперативно і точно можна отримати потрібну інформацію.

2. Підтримка в аналізі даних

Пошукові системи не лише надають доступ до інформації, але й допомагають у її обробці та аналізі. Багато з них вбудовують алгоритми для ранжування результатів, що дозволяє виводити найбільш релевантні документи на перші позиції. Для фахівців, які займаються прийняттям рішень, важливо мати доступ до інформації, яка відповідає їхнім запитам з максимальною точністю. Завдяки цим алгоритмам користувач може уникнути перевантаження інформацією та зосередитися на найбільш важливих аспектах.

3. Розширення можливостей для прогнозування

У деяких пошукових системах, особливо у сфері бізнесу, використовуються додаткові інструменти для аналізу та прогнозування. Наприклад, системи пошуку в комерційних даних можуть допомогти компаніям

оцінити ринкові тенденції або виявити нові можливості. Використання таких систем дозволяє приймати більш обґрунтовані рішення, основані на даних з різних джерел, які можуть передбачити зміни на ринку чи в економіці.

4. Пошукові системи у корпоративному середовищі

Особливо важливу роль пошукові системи відіграють в корпоративному середовищі, де для прийняття рішень часто необхідно працювати з великими обсягами внутрішньої інформації. Власні пошукові системи, інтегровані в корпоративні системи управління знаннями, дозволяють ефективно знаходити потрібні документи, проекти, електронні листи та інші ресурси. Це дозволяє приймати швидкі і обґрунтовані рішення в процесах стратегічного планування, управління проектами, а також у вирішенні поточних операційних завдань.

5. Інтелектуальні пошукові системи та підтримка прийняття рішень

Завдяки розвитку технологій, сучасні пошукові системи стають все більш інтелектуальними. Використання штучного інтелекту та машинного навчання дозволяє системам не лише знаходити релевантну інформацію, а й робити прогнози або рекомендувати можливі шляхи вирішення проблем. Наприклад, рекомендаційні системи в Інтернет-магазинах, в медичних або фінансових застосунках допомагають користувачам приймати обґрунтовані рішення, пропонуючи варіанти на основі аналізу даних.

6. Покращення процесу колективного прийняття рішень

Пошукові системи також сприяють колективному прийняттю рішень, дозволяючи групам людей ефективно співпрацювати, обмінюватися інформацією та ідеями. Наприклад, системи підтримки прийняття рішень (DSS), що інтегрують пошукові технології, допомагають зібрати всі необхідні дані для групової роботи, оцінити різні варіанти та прийняти спільне рішення. Такі системи зазвичай використовуються в урядових установах, великих корпораціях та наукових колективах для вирішення комплексних проблем.

7. Перспективи розвитку пошукових систем у прийнятті рішень

У майбутньому роль пошукових систем у прийнятті рішень буде лише зростати. Впровадження нових технологій, таких як аналіз великих даних,

глибинне навчання та обробка природної мови, дозволить значно покращити точність результатів пошуку, а також допоможе приймати рішення в реальному часі. Зокрема, в фінансовому секторі та медицині пошукові системи будуть використовуватися для прогнозування тенденцій та визначення оптимальних дій на основі даних про поведінку користувачів або пацієнтів.

Висновки до розділу 1

Пошукові системи, зокрема для підтримки прийняття рішень, за останні роки пройшли значний шлях розвитку, від простих текстових пошуків до складних платформ, що використовують штучний інтелект, машинне навчання та великі дані для аналізу та надання рекомендацій. Це відкриває нові можливості для ефективного прийняття рішень у різних сферах, від бізнесу до медицини.

Важливою складовою сучасних пошукових систем є вдосконалення алгоритмів пошуку та інтерфейсів, які здатні враховувати контекст, пріоритети користувача та специфіку його запиту. Інтеграція таких підходів дозволяє значно покращити точність пошуку та адаптацію результатів до конкретних умов використання.

Пошукові системи для підтримки прийняття рішень не лише надають доступ до великої кількості даних, але й здатні допомогти користувачеві визначити найкращий вибір, аналізуючи всі можливі варіанти, порівнюючи їх між собою і враховуючи надану інформацію. Це дає можливість значно підвищити якість прийнятих рішень і знизити ризики помилок.

З огляду на стрімкий розвиток технологій, можна очікувати, що пошукові системи для підтримки прийняття рішень будуть ще більше інтегрувати новітні досягнення в галузі штучного інтелекту, забезпечуючи більш точне і швидке прийняття рішень в умовах постійно змінюваної інформаційної реальності.

Сучасні підходи до проектування пошукових систем для підтримки прийняття рішень є основою для вдосконалення управлінських процесів в

організаціях, а також для поліпшення особистих та професійних рішень користувачів. Актуальність розвитку таких систем зростатиме з розвитком технологій та з необхідністю обробляти дедалі більші обсяги даних для прийняття обґрунтованих рішень в умовах невизначеності та швидко змінюваних обставин.

РОЗДІЛ 2
ЗАГАЛЬНА ПОСТАНОВКА ПРОЕКТНОГО ЗАВДАННЯ

2.1 Розробка вимог до інформаційно-пошукової системи

Одним із багатьох інструментів, які забезпечують швидший обмін інформацією, є чат-бот. Чат-бот — це акаунт на платформі для чатів, який може отримувати та відправляти повідомлення для забезпечення більш ефективної комунікації. Він не потребує додаткової установки з боку користувача на платформі, що робить його зручним у використанні. Хоча він може отримувати і відправляти повідомлення, акаунт чат-бота відрізняється від акаунта людини, оскільки не має статусу онлайн, логів останнього входу та можливості дзвонити іншим акаунтам [11]. Одна з платформ, що зазвичай використовується для розробки чат-ботів, — це Telegram. Telegram — це хмарний чат-додаток, який надає можливості для обміну різними типами файлів. Клієнт Telegram можна запускати як на смартфоні, так і на десктопі. Код клієнта Telegram є відкритим, але з власницькою ліцензією. Використання Telegram безкоштовне і не має обмежень на розмір передаваних медіафайлів [16]. Користувачі можуть надсилати роботам чат-боту будь-які типи повідомлень , включаючи текст, файли, розташування, наклейки, голосові повідомлення. Боти Telegram пропонують багато інших інструментів для створення гнучких інтерфейсів, адаптованих до ваших конкретних потреб [34]:

- команди , які підсвічуються в повідомленнях і можуть бути обрані зі списку після введення /.
- клавіатури , які замінюють клавіатуру користувача з попередньо визначеними варіантами відповідей.
- кнопки , які відображаються біля повідомлень від бота.

Для ще більшої гнучкості веб-програми підтримують 100% користувальницькі інтерфейси з JavaScript.

Боти можуть надати користувачеві зручний та інтуїтивно зрозумілий інтерфейс, який містить список будь-якої кількості груп, каналів або інших користувачів відповідно до спеціального набору критеріїв. Вибір чату надсилає його ідентифікатор боту в службовому повідомленні та плавно закриває інтерфейс. Користувачі можуть взаємодіяти з вашим ботом за допомогою вбудованих запитів прямо з поля повідомлення в будь-якому чаті . Все, що їм потрібно зробити, це почати повідомлення з @username вашого бота та ввести ключове слово.

Боти Telegram мають механізм глибокого посилання, який дозволяє передавати боту додаткові параметри під час запуску. Це може бути команда, яка запускає бота, або токен автентифікації для підключення облікового запису Telegram користувача до його облікового запису на іншій платформі.

Кожен бот має посилання, яке відкриває бесіду з ним у Telegram – https://t.me/<bot_username>. Параметри можна додавати безпосередньо до цього посилання

Отримавши запит, ваш бот може видати деякі результати. Як тільки користувач вибирає один, він надсилається до відповідного чату . Таким чином, люди можуть запитувати та надсилати вміст від вашого бота в будь-якому зі своїх чатів, груп або каналів.

Технічно, чат-бот — це акаунт на платформі для чатів, який може отримувати та відправляти повідомлення, щоб ефективно забезпечити спілкування з користувачами. Чат-бот не потребує додаткової установки з боку користувача на платформі. Він надає функції для отримання та відправлення повідомлень. Однак чат-бот відрізняється від акаунта людини, оскільки не має статусу онлайн, логів останнього входу та не може здійснювати дзвінки іншим акаунтам [5].

Чат-бот імітує розмову з користувачем. Розмови ведуться шляхом відповіді на запити користувача [3]. Чат-бот отримує повідомлення через API. Обробка чат-ботом входящих повідомлень відбувається наступним чином: після обробки чат-бот відправляє відповідь через API користувачу. Telegram не лише

має платформу для чатів, але й надає API-сервіс, до якого розробники можуть звертатися для використання Telegram у різних цілях. API зазвичай використовується для створення чат-ботів. Якщо ми хочемо використовувати Telegram API, ми повинні створити акаунт чат-бота в Telegram. Цей акаунт можна створити через акаунт @botfather (https://telegram.me/BotFather). Ми отримаємо токен, який можна використовувати для доступу до Telegram API (рис. 2.1).

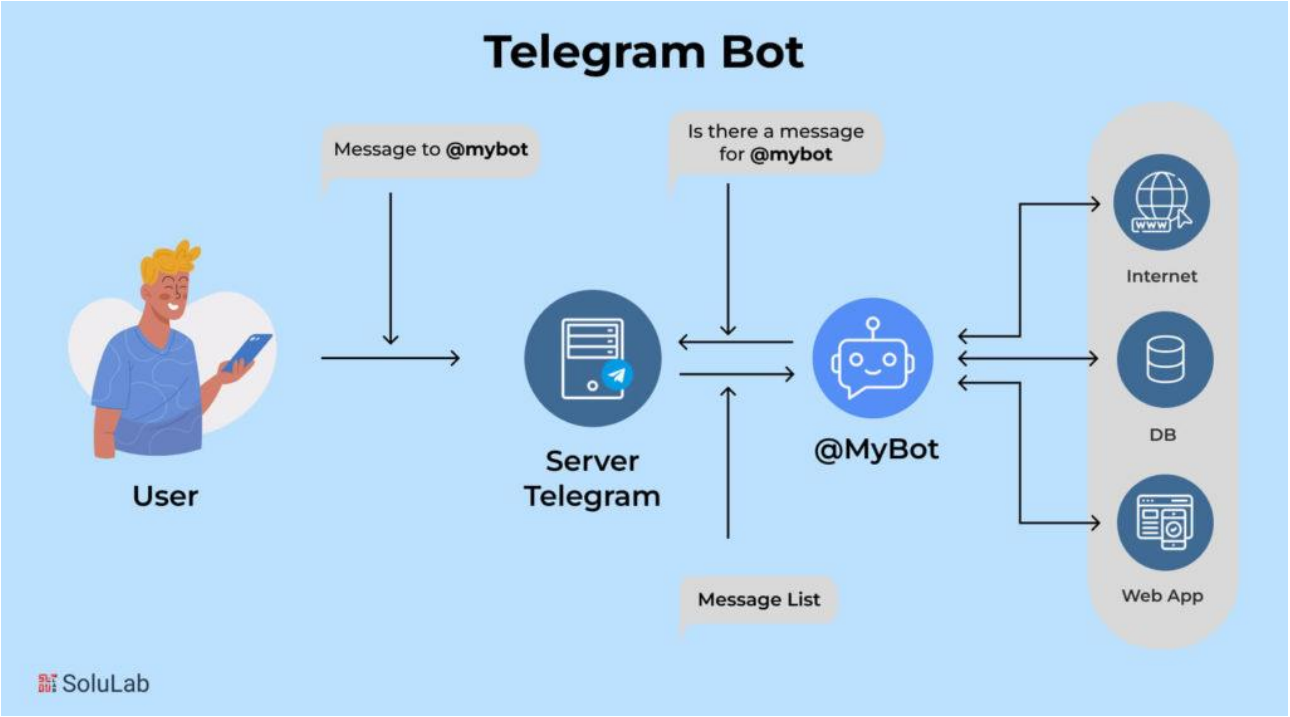

Рисунок 2.1 – Модель функціонування телеграм боту [42]

Telegram пропонує наступні види ботів для користування [42]:

Чат-боти: ці автоматизовані розмовні агенти розроблені для імітації людських розмов за допомогою текстових або голосових взаємодій. Вони широко використовуються в обслуговуванні клієнтів, надаючи допомогу в реальному часі та вирішуючи запити. Чат-боти на базі штучного інтелекту також можна використовувати в розважальних цілях, залучаючи користувачів до інтерактивних розмов або розповідаючи історії. Крім того, вони можуть служити віртуальними помічниками, призначати зустрічі, надавати персоналізовані рекомендації або отримувати інформацію з Інтернету. Якщо вам цікаво, як

створити бота Telegram , ці можливості можна легко інтегрувати в бота Telegram, щоб покращити взаємодію з користувачем.

Новинні боти: ці боти призначені для доставки персоналізованих оновлень новин і статей на основі вподобань та інтересів користувачів. Вони використовують програми обробки природної мови (NLP), щоб розуміти запити користувачів і подавати актуальні новини з різних джерел. Новинних ботів можна інтегрувати в платформи обміну повідомленнями або отримати доступ через автономні програми, щоб користувачі залишалися в курсі поточних подій і тем, які їх цікавлять.

Ігрові боти: ці боти забезпечують захоплюючий ігровий досвід для користувачів, починаючи від простих ігор-головоломок і закінчуючи складними пригодами для кількох гравців. Вони можуть бути розроблені, щоб запропонувати сольну гру або полегшити соціальну взаємодію з іншими гравцями. Навчившись користуватися ботами в Telegram , користувачі зможуть взаємодіяти з ігровими ботами, які часто використовують штучний інтелект (ШІ) , щоб створювати складних і захоплюючих суперників, дозволяючи їм перевіряти свої навички та стратегії.

Службові роботи: Службові роботи призначені для виконання конкретних завдань, які підвищують продуктивність і зручність користувачів. Вони можуть надавати інформацію про погоду в реальному часі, конвертацію валют, переклад на мову або математичні розрахунки. Службових ботів можна інтегрувати в різні платформи або отримати доступ через спеціальні програми, що спрощує щоденні завдання та зменшує потребу в ручному пошуку чи обчисленнях.

В залежності від того, чи чат-бот функціонує на основі правил чи грунтуєтсья на ШІ вирізняють і відповідні види чат-ботів, кожен з яких має свої переваги та недоліки (таблиця 2.1).

Таблиця 2.1 - Характеристика видів чат-ботів [15]

Характеристика	Чат-боти на основі правил	Чат-боти на основі ШІ
Здатність до навчання	Вимагає оновлень	Навчається через взаємодію з користувачами
Гнучкість	Обмежується попередньо визначеними правилами	Здатний адаптуватися до нових сценаріїв
Контекстне розуміння	Неможливо зрозуміти контекст розмови	Здатний розуміти та зберігати контекст
Складність реалізації	Легко розробити та реалізувати	Потребує поглибленого розвитку та навчання
Технічне обслуговування	Нездатний до саморозвитку без ручного втручання	Саморозвивається з мінімальним ручним втручанням
Досвід користувача	Нездатність обробляти складні запити; більше підходить для легких, простих завдань	Здатний обробляти складні та динамічні взаємодії; більш привабливим

Для запуску Telegram чат-бота Telegram API надає два методи: long-polling та webhook. За допомогою long-polling сервер має перевіряти застосунок періодично. Коли з'являються нові повідомлення, сервер обробляє їх. Метод long-polling підходить для тих, хто не має https-сервера або веб-хостингу, оскільки його можна запустити на своєму комп'ютері. В той час як метод webhook не потребує періодичної перевірки, оскільки Telegram API надішле повідомлення на сервер, коли воно надійде. Тому для роботи методу webhook потрібно мати активний https-сервер або веб-хостинг, оскільки Telegram API буде відправляти вхідні повідомлення на його сервер [15] (рис. 2.2).

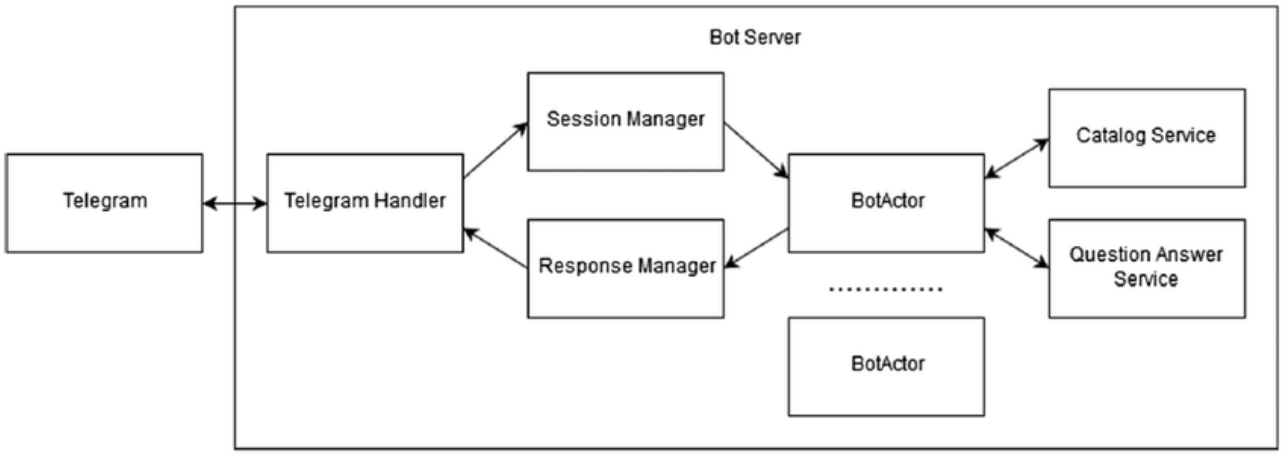

Рисунок 2.2 – Шаблон системи чат-бота з використанням Telegram [15]

Служба обміну повідомленнями Telegram надає підтримку технології чат-ботів і має добре задокументоване API.

Telegram API є інтерфейсом для розробників, який надає набір інструментів та протоколів для взаємодії з платформою Telegram. Цей інтерфейс дозволяє створювати програми, боти та сервіси, що інтегруються з Telegram, використовуючи його функціональні можливості. Він визначає правила обміну даними між різними програмами та сервісами [35]. Зокрема, API Telegram дає можливість розробникам відправляти повідомлення, керувати контактами, створювати групи й канали, а також працювати з медіафайлами [43].

API Telegram відкриває широкі перспективи для створення різноманітних додатків та сервісів. Наприклад:

- розробники можуть створювати боти, які автоматично взаємодіють із користувачами, надсилаючи їм повідомлення і виконуючи різні завдання;
- можна також розробляти програми, що інтегруються з функціями Telegram, такими як соціальні мережі чи платформи для обміну медіа;
- API Telegram дозволяє з’єднувати його з іншими зовнішніми сервісами, наприклад, для автоматичної відправки повідомлень через RSS-стрічки чи CRM-системи.

Щоб почати роботу з API Telegram, необхідно отримати API-ключ, який ідентифікує вашу програму чи бота. Цей ключ забезпечує безпечне з’єднання з Telegram API. Після цього можна відправляти HTTP-запити до відповідних

кінцевих точок для виконання операцій, таких як надсилання повідомлень або створення груп. Для більш складних завдань існують спеціалізовані бібліотеки та SDK для різних мов програмування, які полегшують взаємодію з API Telegram [41].

Спілкування через Інтернет між усіма сторонами зашифроване. Це стосується як обміну інформацією між CamBot та службою обміну повідомленнями Telegram, так і обміну інформацією до смартфонів користувачів. Сервіс питання-відповідь дозволяє вести розмови та реагувати на дії та запити користувача (рис. 2.3).

При цьому в літературі виділяють наступні переваги використання чат ботів для забезпечення взаємодії із користувачами [1]:

1. Миттєва доступність: Для використання чат-бота через службу обміну повідомленнями на смартфоні достатньо лише завантажити додаток. Для цього не потрібні технічні знання.

2. Легка крива навчання: Оскільки більшість користувачів смартфонів знайомі з текстовими повідомленнями, їм легко почати використовувати чат-бота. Зазвичай основні функції можна використовувати інтуїтивно.

3. Сповіщення: Хоча надмірна кількість сповіщень на смартфоні може здатися набридливою, комунікація через службу обміну повідомленнями є звичною практикою для більшості користувачів смартфонів.

4. Соціальна графіка та контакти: Користувачі вже звикли добровільно ділитися контактною інформацією в додатках для обміну повідомленнями. Інформацію можна поширювати в рамках знайомих мереж користувачів (наприклад, родини чи професійних організацій догляду).

5. Незалежність від платформи: Чат-боти, доступні через служби обміну повідомленнями, такі як Telegram, забезпечують незалежність від платформи, оскільки додатки для популярних месенджерів доступні для більшості мобільних операційних систем (наприклад, iOS, Android тощо).

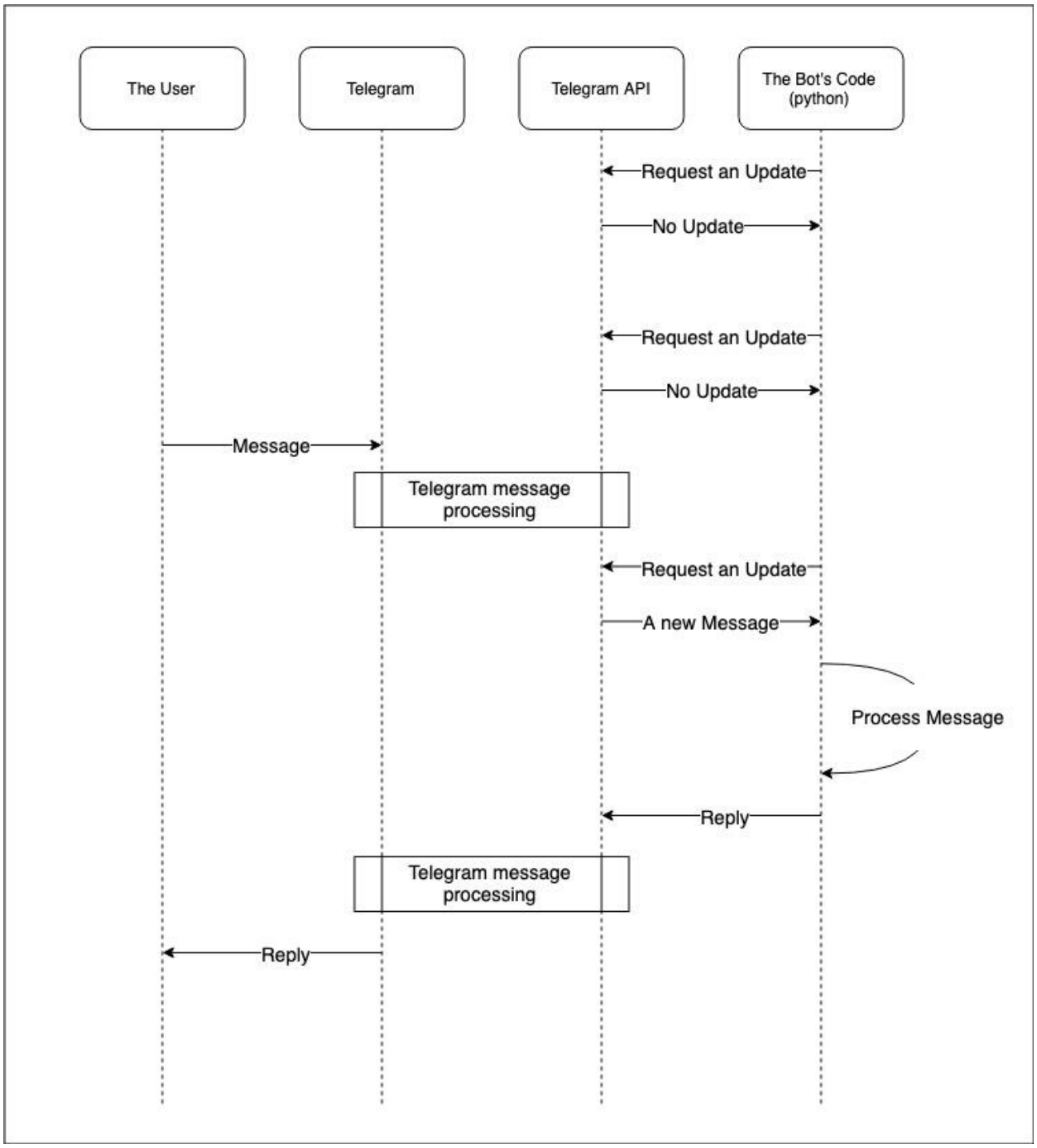

Рисунок 2.3 – Процесна модель роботи телеграм чат-бота

6. Аутентифікація: Оскільки користувацька концепція платформи-хосту може бути використана, реєстрація з іменем користувача та паролем не потрібна. Це відома перешкода для входу користувачів.

7. Асинхронність: Обмін миттєвими повідомленнями є асинхронним завданням: після відправки повідомлення користувачі не повинні чекати

відповіді. Коли отримувач повідомлення тимчасово недоступний, повідомлення буде доставлено пізніше.

Отже, у якості пошукової системи для прийняття управлінських рішень було обрано Telegram чат-бот, що дозволить допоможе користувачам швидко знаходити вакансії в інтернеті, а також створювати та редагувати резюме. Система повинна здійснювати інтелектуальний аналіз вакансій та пропонувати користувачеві найбільш відповідні результати та відповідати наступним вимогам:

1. Функціональність пошуку інформації

Оскільки інфрмаційно-пошукова система — це програмний інструмент, який дозволяє користувачеві знаходити інформацію за певними запитами або критеріями, то основною функцією розробленого чат-боту буде пошук вакансій. Зміст даної функції буде полягати в тому, що користувач вводить запит (ключові слова, посаду, місцезнаходження), і бот здійснює пошук відповідних вакансій на численних зовнішніх платформах (LinkedIn, Work.ua, Jooble тощо). Користувачу буде забезпечена можливість використовувати різні фільтри для звуження результатів пошуку. Тобто користувач може адаптувати свої запити, отримуючи релевантні результати.

2. Агрегація даних з різних джерел

У запропонованому чат-боті пропонується агрегація вакансій з різних джерел, таких як платформи пошуку роботи (Work.ua, LinkedIn, Jooble). Користувач вводить запит, а система агрегує та фільтрує вакансії з усіх доступних джерел, щоб надіслати йому релевантні результати.

3. Алгоритми пошуку та фільтрації

Основною характеристикою пошукової системи є наявність алгоритмів для пошуку та сортування даних. В даному випадку передбачається використання алгоритмів, які дозволяють знаходити роботи за різними критеріями, такими як посада, локація, рівень зарплати, досвід, тип роботи і т.д. Бот також пропонує користувачеві фільтрування і сортування результатів пошуку, що є класичним функціоналом пошукових систем.

4. Інтерфейс пошукової системи

Чат-бот в Telegram виступає як інтерфейс пошукової системи, дозволяючи користувачеві взаємодіяти з системою пошуку за допомогою простих запитів у вигляді повідомлень. Бот автоматично відповідає на запити користувача, обробляє введену інформацію та надає відфільтровані результати — це функціонал, який характерний для будь-якої пошукової системи.

5. Підтримка прийняття рішень

Пошукова система не просто знаходить дані, а й допомагає користувачу у прийнятті рішення, обираючи найкращі варіанти з множини результатів. Крім того додавання функції підтримки складання резюме є додатковою функцією до пошуку вакансій, яка забезпечує підтримку користувача у важливому процесі підготовки до подачі на знайдені вакансії. Пошук і складання резюме можуть працювати синергетично для ефективнішого прийняття рішень на основі отриманих вакансій.

Отже, чат бот буде мати наступні характеристики:

1. Інтерфейс користувача (Telegram чат-бот) буде підтримувати наступні функції:

Пошук вакансій: Користувач вводить бажану посаду або ключові слова для пошуку, і бот надає список відповідних вакансій з різних джерел (наприклад, LinkedIn, Work.ua, Jooble тощо).

Бот може підтримувати фільтрацію за місцем розташування, зарплатою, досвідом роботи.

Розширений пошук: Користувач може уточнити пошук за іншими критеріями, такими як тип роботи (фул-тайм, часткова зайнятість), сфера діяльності та інші параметри.

Складання резюме: Бот допомагає користувачеві створювати резюме, пропонуючи шаблон та поля для введення необхідної інформації, в яких користувач може вводити особисті дані, досвід роботи, освіту, навички тощо. На основі уведеної інформації буде відбуватися генерацію створення резюме для подальшого відправлення роботодавцям.

1. Користувацький процес:

Користувач додає бота в Telegram і запускає його.

Бот пропонує кілька основних опцій:

Пошук вакансій.

Складання резюме.

При виборі пошуку вакансій, бот запитує ключові слова (посаду або навички) і повертає список актуальних вакансій.

Якщо користувач вибирає складання резюме, бот сприяє в заповненні форм і надає рекомендації щодо змісту.

3. Пошукова система вакансій. Для реалізації ефективного пошуку вакансій запропонова інтеграція чат-боту з популярними платформами для пошуку вакансій, такими як:

LinkedIn API — для доступу до вакансій на платформі LinkedIn.

Work.ua API — для отримання вакансій з Work.ua.

Jooble API — агрегатор вакансій з різних сайтів.

Виведення отриманих результатів буде відбуватися із застосуванням алгоритмів фільтрації та сортування за критеріями:

- місцезнаходження;
- зарплата;
- досвід;
- тип роботи.

2.2 Технології розробки чат-ботів

Розробка чат-ботів може бути реалізована за допомогою різноманітних мов програмування та платформ, залежно від потреб і вимог проекту. Вибір мови програмування визначається багатьма факторами, такими як тип чат-бота, його функціональні можливості, інтеграція з різними сервісами та платформами, а також досвід розробника. Вибір правильної мови програмування особливо важливий у контексті розробки чат-ботів . Це впливає як на функціональність

чат-бота, так і на досвід користувача. Вірно обрана мова найкраще відповідатиме цілям проекту та технічним потребам. Різні мови дозволяють додати до чат-боту різноманітних функцій, властивих певному типу чат-бота. Деякі мови в основному зосереджені на продуктивності та масштабованості, тоді як інші підтримують обробку природної мови. Ці параметри необхідно розуміти та брати до уваги, приймаючи відповідне рішення.

Розглянемо основні мови програмування, які часто використовуються для створення чат-ботів [2, 14].

1. Python є однією з найпопулярніших мов програмування для розробки чат-ботів завдяки своїй простоті, потужним бібліотекам та активній спільноті розробників. Python підтримує численні бібліотеки та фреймворки для роботи з чат-ботами, зокрема [32]:

ChatterBot — бібліотека для створення чат-ботів, що використовує машинне навчання для автоматичного вдосконалення відповідей.

NLTK (Natural Language Toolkit) — потужний інструмент для обробки природної мови, що дозволяє чат-ботам розуміти та генерувати людську мову.

Rasa — платформа з відкритим кодом для розробки чат-ботів, яка дозволяє створювати складні діалоги та інтегрувати чат-бота з різними системами.

Python також популярний завдяки своїй універсальності та простоті інтеграції з різними API, що дозволяє створювати чат-боти для платформ, таких як Telegram, Slack, Facebook Messenger

2. JavaScript використовується для розробки чат-ботів на платформі Node.js. Веб-технології, такі як Node.js, дозволяють створювати чат-ботів для інтерактивних веб-сайтів і веб-додатків. Node.js, середовище виконання для JavaScript, ще більше збагачує розробку чат-ботів. Він розроблений для створення масштабованих мережевих програм, дозволяючи чат-ботам обробляти кілька запитів одночасно. Ця можливість має вирішальне значення для чат-ботів, які обслуговують велику базу користувачів.

Велика екосистема JavaScript у поєднанні з Node.js пропонує численні бібліотеки та фреймворки. Такі інструменти, як Botpress і Microsoft Bot

Framework, спрощують створення функцій чат-ботів. Ці фреймворки можуть значно скоротити час розробки.

Неблокуюча, керована подіями архітектура JavaScript робить його ідеальним для додатків чату в реальному часі. Чат-боти можуть миттєво обробляти повідомлення користувачів і відповідати на них. Ця взаємодія в режимі реального часу підвищує залученість і задоволення користувачів.JavaScript дозволяє швидко та ефективно обробляти запити користувачів і взаємодіяти з іншими сервісами.

Деякі популярні інструменти для створення чат-ботів на JavaScript включають [30]:

- Botpress — відкритий фреймворк для розробки чат-ботів, який підтримує інтеграцію з кількома платформами і дозволяє налаштовувати складні діалоги;
- Microsoft Bot Framework — платформа для створення чат-ботів з підтримкою Node.js, яка дозволяє створювати чат-ботів для багатьох каналів (Skype, Facebook, Slack);

3. Java також широко використовується для проектів чат-ботів, оскільки це об’єктно-орієнтована мова загального призначення, яка не залежить від платформи та переноситься. Однією з переваг Java є те, що програми, написані на ній, можуть працювати на будь-якій системі, де встановлено віртуальну машину Java (JVM), що робить її універсальною мовою. Java допускає багатопотоковість, що забезпечує вищу продуктивність, ніж багато інших мов у цьому списку. Він також широко використовується в розробці підприємств — це означає, що чат-бота, написаного на Java, можна легко інтегрувати з корпоративними екосистемами. Java також має великий вибір сторонніх бібліотек для машинного навчання та NLP, включаючи Stanford Library NLP і Apache Open NLP.

Основні інструменти для розробки чат-ботів на Java включають [29]:

- Spring Boot — популярний фреймворк для розробки веб-додатків, який використовується для створення чат-ботів, що потребують інтеграції з іншими веб-сервісами;

- Dialogflow — платформа від Google для розробки чат-ботів, яка підтримує Java та дозволяє легко інтегрувати чат-ботів з різними каналами зв'язку;

Java є одним з найбільш надійних виборів для чат-ботів, що мають складну логіку або повинні працювати в серйозних бізнес-системах, де надійність і безпека мають критичне значення.

4. PHP є мовою програмування, що також широко використовується для створення чат-ботів на веб-платформах, зокрема для інтеграції з чат-системами на веб-сайтах. PHP добре підтримує роботу з базами даних та може бути використаний для інтеграції з різними API чат-платформ.

Основні бібліотеки та інструменти для роботи з чат-ботами на PHP включають [44]:

- PHP-Chatbot — простий фреймворк для створення базових чат-ботів, які можуть працювати на веб-сайтах або інтегруватися з різними сервісами;
- BotMan — популярний PHP-фреймворк для створення чат-ботів, який дозволяє інтегрувати бота з декількома платформами, такими як Telegram, Slack, Facebook Messenger.

PHP є хорошим вибором для створення чат-ботів для веб-сайтів або малих проектів, де основною вимогою є швидкість і простота впровадження.

5. C# використовується для розробки чат-ботів у середовищі .NET. C# є хорошим вибором для інтеграції з продуктами Microsoft і для розробки чат-ботів для корпоративних рішень. Для створення чат-ботів на C# розробники можуть використовувати [45]:

Microsoft Bot Framework — популярний інструмент для розробки чат-ботів на платформі .NET, який дозволяє інтегрувати ботів з різними каналами зв'язку, такими як Skype, Teams, Facebook Messenger.

Botkit — фреймворк для створення чат-ботів, який підтримує C# і дозволяє інтегрувати чат-ботів з популярними месенджерами.

C# є відмінним вибором для створення потужних чат-ботів для великих підприємств, які працюють з Microsoft платформами.

6. Використання мови програмування Ruby вирізняється елегантністю та простотою, що робить цю мову програмування кращим вибором для створення чат-ботів. Його динамічний характер дозволяє швидко створювати прототипи, що є ключовою вимогою для проектів, пов'язаних зі ШІ. Чат-боти, створені за допомогою Ruby, мають інтуїтивно зрозумілий синтаксис та активну спільноти розробників.

НЛП є частиною загального функціоналу чат-бота. Ruby є кращим у цій галузі. Він пропонує надійні бібліотеки та фреймворки, такі як Ruby on Rails. Ці інструменти полегшують виконання більш складних завдань НЛП. Такі завдання, як аналіз настроїв і розуміння мови, виконувати легше.

Однією з сильних сторін Ruby є його здатність зосереджуватися на досвіді розробника, просуваючи чистіший код із меншими зусиллями. Такий фокус зменшує потенційні помилки та прискорює процес розробки [33].

7. Go, також відомий як Golang, набирає обертів у середовищі розробки чат-ботів. Go, відомий своєю швидкістю та простотою, ідеально підходить для розробників, які надають перевагу ефективності. Його можливості паралелізму роблять його ідеальним для чат-ботів, які обробляють численні одночасні взаємодії.

Однією з переваг використання Go для розробки штучних чат-ботів є його масштабованість. Зі зростанням попиту проста мова дозволяє системам справлятися з великими навантаженнями без зниження продуктивності. Це вкрай важливо для додатків великого обсягу та чат-ботів, пов'язаних із обслуговуванням клієнтів [28].

Go надає інструменти для швидкої розробки та розгортання. Його чіткий синтаксис і низькі витрати на обслуговування вимагають менше навчання. Це підвищує продуктивність і допомагає розробникам.

Для розробки Телеграм чат-боту було обрано мову програмування Python, що має ряд переваг для проектування чат-ботів:

1. Простота: Python має чистий і читабельний синтаксис, що робить його легким для вивчення та розуміння навіть для початківців. Ця простота дозволяє підприємцям швидко осягнути основи програмування та почати створювати своїх ботів Telegram, не перевантажуючись.

2. Велика спільнота та ресурси: Python має велику й активну спільноту розробників, які сприяють її розвитку. Це означає, що доступні численні ресурси, навчальні посібники та бібліотеки, які допоможуть розробити ваш бот Telegram. Процвітаюча спільнота також гарантує, що ви можете легко знайти допомогу та керівництво, коли це необхідно.

3. Широкий вибір бібліотек: Python пропонує широкий вибір бібліотек і фреймворків, спеціально розроблених для веб-розробки та створення ботів. Ці бібліотеки, такі як python-telegram-bot, спрощують взаємодію з API Telegram Bot і спрощують процес розробки, надаючи готові функції та абстракції.

4. Гнучкість: Python забезпечує гнучку розробку, дозволяючи підприємцям створювати ботів Telegram різного рівня складності. Незалежно від того, чи створюєте ви простого бота для автоматизації завдань, чи більш просунутого бота з можливостями обробки природної мови, Python забезпечує гнучкість, щоб задовольнити ваші конкретні вимоги.

Для створення телеграм чат-боту буде використана бібліотека python-telegram-bot, яка надає чистий асинхронний інтерфейс Python для Telegram Bot API та є сумісною із версіями Python 3.9+ [19].

Основні особливості бібліотеки python-telegram-bot:

1. Простота у використанні: python-telegram-bot має чистий та інтуїтивний API, що робить її відмінним вибором для початківців та досвідчених розробників.

2. Підтримка асинхронності: Бібліотека дозволяє використовувати асинхронне програмування за допомогою async/await, що робить обробку безлічі одночасних запитів більш ефективною.

3.Обробка повідомлень: python-telegram-bot надає зручні засоби для обробки вхідних повідомлень, команд та подій. Ви можете легко реагувати на текстові повідомлення, зображення, відео та інші медіа-файли.

4. Підтримка клавіатури: Ви можете створювати інтерактивні клавіатури для взаємодії з користувачами вашого робота.

5. Інтеграція зі сторонніми сервісами: Бібліотека дозволяє інтегрувати бота із зовнішніми сервісами, такими як бази даних та веб-сервіси.

Крім чистої реалізації API, ця бібліотека містить кілька зручних методів і ярликів, а також низку класів високого рівня, щоб зробити розробку ботів легкою та простою. Ці класи містяться в підмодулі telegram.ext.Application.

Клас Application відповідає за отримання оновлень із update_queue, де Updaterклас постійно отримує нові оновлення з Telegram і додає їх до цієї черги. Application об’єкт за допомогою ApplicationBuilder автоматично створить Updater і зв’язує його із asyncio.Queue. Додатково можуть бути зареєстровані обробники різних типів у Application, які сортуватимуть оновлення, отримані відповідно Updater до зареєстрованих обробників, і доставлять їх до функції зворотного виклику.

Кожен обробник є екземпляром будь-якого підкласу класу telegram.ext.BaseHandler. Бібліотека надає класи обробників майже для всіх випадків використання.

2.3. Середовище програмування

Для розробки та реалізації чат-бота для підтримки прийняття рішень, зокрема пошуку вакансій та складання резюме, було обрано середовище програмування PyCharm.

PyCharm — це інтегроване середовище розробки (IDE), яке використовується для програмування на Python. Він забезпечує аналіз коду, графічний налагоджувач, інтегрований модульний тестер, інтеграцію з системами контролю версій і підтримує веб-розробку за допомогою Django. PyCharm розроблено чеською компанією JetBrains і побудовано на їх платформі IntelliJ [31].

Він кросплатформний, працює на Microsoft Windows, macOS і Linux. PyCharm має професійну версію, випущену за пропрієтарною ліцензією, і видання спільноти, випущену за ліцензією Apache **[10]** Версія PyCharm Community Edition менш обширна, ніж професійна версія.

PyCharm є потужним інтегрованим середовищем розробки, яке пропонує низку переваг для програмістів, зокрема для розробників Python. Серед основних переваг PyCharm можна виділити [31]:

1. Глибоке розуміння структури проєкту та підтримка написання коду: PyCharm автоматично доповнює код, виявляючи помилки та зайві елементи, пропонуючи відповідні виправлення, що сприяє підвищенню продуктивності розробника. Крім того, середовище забезпечує безпечний процес рефакторингу коду.

2. Швидка навігація та ефективний пошук: Незалежно від розміру проєкту, розробники можуть швидко переходити до визначення будь-якої функції, методу, змінної чи класу. Також доступна можливість пошуку файлів, класів, змінних або методів з переглядом всіх співпадінь в одному місці, що значно спрощує роботу з великими кодовими базами.

3. Інтеграція необхідних інструментів: PyCharm надає розробникам доступ до важливих інструментів, таких як можливості налагодження та тестування Python-коду, інтеграція з Git і GitHub, а також підтримка Docker. Це дозволяє працювати в одному середовищі без необхідності постійного перемикання між різними інструментами.

4. Повне доповнення коду: Завдяки функціоналу доповнення коду в реальному часі, PyCharm пропонує AI-підказки для ефективного завершення

коду, спрощуючи роботу розробника і знижуючи ймовірність помилок. Цей процес відбувається локально, без передачі коду через інтернет, що забезпечує конфіденційність.

5. Інтегрований AI-асистент: PyCharm надає вбудовані можливості штучного інтелекту для багаторядкового доповнення коду, контекстно-залежного чату та інших функцій, які підвищують швидкість розробки. Інтеграція цих функцій у середовище дозволяє значно зменшити час на виконання рутинних завдань, таких як написання документації або генерація повідомлень для комітів.

6. Розвиток віддаленої розробки: PyCharm дозволяє налаштувати віддалений Python-інтерпретатор або підключитися до віддаленої машини через SSH, що робить можливим розробку в середовищі, що імітує виробничі умови. Це є важливим для проєктів, які працюють в хмарних середовищах, таких як Google Cloud Workstations або GitHub Codespaces.

7. Інтегроване управління базами даних: Середовище підтримує роботу з численними базами даних, такими як PostgreSQL, SQLite, MySQL, Redis, MongoDB, що дозволяє розробникам безперешкодно досліджувати дані, змінювати схеми, виконувати запити та працювати з UML-діаграмами для аналізу баз даних.

Ці переваги роблять PyCharm потужним інструментом для професійних розробників, які потребують зручного, ефективного та інтегрованого середовища для створення програмних рішень.

Висновки до розділ 2.

Розробка чат-ботів для автоматизації процесів, таких як пошук вакансій та складання резюме, є ефективним інструментом для покращення комунікації з користувачами. Чат-боти, які взаємодіють із користувачами через прості повідомлення, не потребують додаткової установки і є зручними у використанні. Вони отримують і надсилають повідомлення через API, що забезпечує швидку

та зручну взаємодію. Крім того, чат-бот може бути оснащений такими функціями, як фільтрація вакансій за місцезнаходженням, зарплатою чи досвідом, а також автоматичне створення резюме на основі введених користувачем даних.

Для розробки таких чат-ботів вибір мови програмування є ключовим. Python є оптимальним вибором завдяки своїй популярності, простоті освоєння та великій кількості бібліотек, що дозволяють реалізувати необхідні функціональні можливості чат-бота в найкоротші терміни.

Середовище програмування PyCharm надає всі необхідні інструменти для ефективної розробки, тестування та інтеграції з різними API. Завдяки своїм функціям для автоматичного доповнення коду, налагодження та інтеграції з зовнішніми сервісами, PyCharm є ідеальним вибором для створення чат-ботів.

Загалом, вибір Python як мови програмування та PyCharm як середовища розробки дозволяє створити надійного та функціонального чат-бота, що забезпечує ефективне виконання завдань пошуку вакансій та складання резюме.

РОЗДІЛ 3
ПРАКТИЧНА РЕАЛІЗАЦІЯ РОЗРОБЛЕНОЇ ІНФОРМАЦІЙНО-ПОШУКОВОЇ СИСТЕМИ

3.1 Створення серверної частини чат-боту

Одним із основних етапів створення системи керування чат-ботом є розробка її серверної частини. Вона є ключовим елементом, що забезпечує взаємодію чат-бота з користувачами, обробку запитів та їх відповідей, а також інтеграцію з різними зовнішніми сервісами. Створення серверної частини чат-бота для Телеграм передбачає кілька етапів: вибір технологій, налаштування серверного середовища, розробка API для взаємодії з платформою, а також забезпечення безпеки та масштабованості. При створенні інформаційно-пошукової системи пошуку вакансій у вигляді чат-бота на Python, конкретні пункти можна реалізувати наступним чином.

1. Обмін даними з серверами Telegram, надсилання/отримання запитів до/від Telegram Bot API.

2. Функціональні компоненти системи у вигляді мікросервісів.

3. Веб-API.

4. Захист даних

При цьому реалізація буде відбуватися з використанням принципів Solid - це набір інструкцій, які допомагають розробникам програмного забезпечення писати зручний, масштабований і гнучкий код. Акронім SOLID розшифровується як п’ять фундаментальних принципів об’єктно-орієнтованого програмування [37]:

S — Принцип єдиного обов'язку (Single Responsibility Principle, SRP). Введений Робертом Мартіном у книзі «Agile Software Development: Principles, Patterns, and Practices» 2002 року. Мартін підкреслив важливість створення класів, які відповідають лише за один напрям дій.

O — Принцип відкритості для розширення, закритості для змін (Open/Closed Principle, OCP). Вперше описаний Бертраном Меєром у книзі «Object-Oriented Software Construction» 1988 року. Він визначив, що класи повинні бути відкритими для розширення, але закритими для модифікації, щоби сприяти гнучкості та легкості розширення коду.

L — Принцип підставлення Лісков (Liskov Substitution Principle, LSP). Сформульований Барбарою Лісков 1988 року. Її робота «A Behavioral Notion of Subtyping» визначила умови, за якими один об'єкт може замінити інший без змін поведінки програми.

I — Принцип розділення інтерфейсу (Interface Segregation Principle, ISP). Введений Робертом Мартіном, він розширив ідеї використання малих та специфічних інтерфейсів для зменшення залежностей.

D — Принцип інверсії залежностей (Dependency Inversion Principle, DIP). Сформульований Робертом Мартіном, визначає важливість того, щоби модулі низького рівня не залежали від верхніх модулів, а обидва типи залежали від абстракцій.

Для розробки власних чат-ботів існує бот у Telegram, який має назву BotFather. За допомогою BotFather користувачі можуть створювати й налаштовувати свої власні боти, а також отримувати додаткові функції та можливості для них (

BotFather дає змогу створювати нові боти зі своїм унікальним іменем та ідентифікатором. Користувачі можуть вибирати тип бота, який вони хочуть створити, зокрема бот для розсилки повідомлень, бот для створення опитувань або бот для обробки замовлень. Після створення бота BotFather дає змогу налаштувати його функції та параметри. Користувачі можуть встановлювати автоматичні відповіді, генерувати ключі API, визначати права доступу користувачів та багато іншого [38]

Для створення власного бота існує низка команд, які подані в таблиці 3.1.

Таблиця 3.1 - Перелік основних команд для бота [23]

Назва команди	Для чого потрібна команда
/start	початок роботи з BotFather
/newbot	команда для створення нового чат-бота
/setname	команда для зміни назви бота
/setdescription	команда для зміни опису бота
/setcommands	команда для додавання або видалення команд бота
/setprivacy	команда для налаштування приватності бота
/revokebot	команда для відкликання доступу до бота
/help	команда для отримання допомоги та інформації про доступні команди

Для розробки чат бота проведемо декомпозицію мікросервісів на субдоменні моделі. Отримаємо наступні (рис. 3.1):

- взаємодія з Telegram Bot API;
- керування файлами;
- робота з повідомленнями (чат);
- взаємодія з Web-API (requests) та парсінг сайтів;
- аналіз результатів та обробка даних відповідно до фільтрів.

Рисунок 3.1 – Взаємодія роботи з мікросервісами

API (Application Programming Interface) – це набір правил і протоколів, які визначають, як одна програма може взаємодіяти з іншою програмою або сервісом. У контексті розробки ботів API використовується для встановлення зв’язку між ботом і зовнішніми сервісами або додатками. Він визначає, як бот може отримати доступ до функцій, даних та можливостей, що надаються іншими сервісами, як-от соціальні мережі, поштові сервіси, бази даних тощо [7].

Для чат-бота на Python, який здійснює пошук вакансій, обмін даними з серверами Telegram реалізується через Telegram Bot API. Основною бібліотекою для взаємодії з API є python-telegram-bot. Для чат-ботиу з пошуку вакансій це буде реалізовано наступним чином.

Надсилання запитів: Для отримання оновлень від користувачів бот використовує механізм webhook або polling. У випадку webhook, сервер чекає запитів на певний URL-адрес, яку Телеграм використовує для надсилання повідомлень. В разі polling бот періодично опитує сервер Телеграм, щоб отримувати нові повідомлення.

Бот може отримувати запити від користувачів, які передаються через API на сервер, який здійснює пошук відповідних вакансій через API зовнішніх джерел. При цьому логіка обробки запитів буде виглядати наступним чином:

- користувач обирає команду в меню "Пошук вакансій";
- бот запитує додаткові параметри: "У якому місті ви шукаєте роботу?", "Яка професія вас цікавить?";
- бот відправляє запит на сервер або зовнішній API (наприклад, на сайт з вакансіями, як robota.ua або Work.ua) для пошуку відповідних вакансій;
- після отримання результатів, бот надсилає список вакансій користувачу у вигляді текстових повідомлень.

Визначимо API мікросервісу взаємодії з Telegram Bot API (Таблиця 3.2).

Таблиця 3.2 - API мікросервісу взаємодії з Telegram Bot API

Операції	Взаємодія з сервісами	Опис
connectToTelegramAPI(params) *createBotGateServiceListener(params)* *createBotLogicServiceConnection(params)* *createFileServiceConnection(params)*		Ініціалізація сервісу.
onTelegramAnyAction(action)	Сервіс логіки чат-бота	Обробка подій отриманих від *Telegram Bot API.*
onTelegramTextMessage(msg) *onTelegramPhotoMessage(msg)* *onTelegramDocumentMessage(msg)*	Сервіс повідомлень	Обробка текстових або медіа повідомлення.
sendTelegramTextMessage(msg) *sendTelegramPhotoMessage(msg)* *sendTelegramDocumentMessage(msg)*	Сервіс файлів	Відправка повідомлень.

Для чат-бота пошуку вакансій можна розглянути використання мікросервісної архітектури, щоб розділити відповідальність за різні аспекти роботи бота:

Мікросервіс пошуку вакансій: Цей мікросервіс відповідає за запит до бази даних або зовнішніх ресурсів (API), таких як LinkedIn, Work.ua або інші сервіси вакансій. Він отримує параметри пошуку від бота (наприклад, місто, спеціалізація, рівень досвіду) і виконує пошук вакансій.

Мікросервіс обробки запитів від користувачів: Цей сервіс відповідає за обробку команд користувача (наприклад, запит на пошук вакансій) та інтеракцію з ботом. Він може працювати з системою черг (наприклад, RabbitMQ), щоб асинхронно надсилати запити до мікросервісів пошуку та відповідати користувачу.

Мікросервіс збереження даних: Цей сервіс відповідає за збереження історії пошуків, даних про вакансії або користувачів, які взаємодіють з ботом. Він може використовувати реляційну базу даних (наприклад, PostgreSQL) або NoSQL базу (наприклад, MongoDB).

Обмін даними між мікросервісами: Для взаємодії між мікросервісами можна використовувати REST API або протоколи для асинхронної обробки

запитів (наприклад, через RabbitMQ або Kafka). Збір інформації та відповіді від мікросервісів мають оброблятися централізовано через основний API бота.

Веб-API буде використовуватися для взаємодії між чат-ботом та зовнішніми ресурсами або сервісами пошуку вакансій. Для цього в чат-боті на Python можна налаштувати інтерфейс для виклику зовнішніх API, що надають вакансії.

Пошук вакансій через API: Бот може використовувати сторонні API для пошуку вакансій, наприклад, API для роботи з платформами з вакансіями (LinkedIn, Work.ua або інші). Бот може надсилати запити до цих API, обробляти відповіді та передавати їх користувачу.

Приклад запиту до API зовнішнього сервісу для отримання вакансій (рис. 3.2):

```python
import requests

def get_job_vacancies_from_api():
    url = "https://api.work.ua/vacancies"
    params = {'city': 'Kyiv', 'specialization': 'IT'}
    response = requests.get(url, params=params)
    return response.json()  # Обробляємо отриману відповідь
```

Рисунок 3.2 – Запиту до API зовнішнього сервісу

Опишемо API зовнішнього сервісу на основі взаємодії з веб-додатком (табл. 3.3). Даний сервіс буде використаний усіма сервісами, які підтримують передачу даних через протоколи HTTP та WebSockets.

Враховуючи реалізовану функцію завантаження резюме опишемо і API мікросервісу керування файлами

Мікросервіс для керування файлами буде мати кілька основних компонентів:

- API для обробки завантаження файлів: Це API, яке дозволяє отримати та зберігати файли (наприклад, резюме) на сервері;

- API для створення та оновлення резюме: Це API, яке інтегрується з веб-формою для збору даних про користувача, а також для подальшої обробки;
- інтерфейс для чат-бота: Це частина системи, яка взаємодіє з Telegram Bot API для обробки запитів на завантаження файлів і передачу їх до мікросервісу.

Таблиця 3.3 – API зовнішнього сервісу взаємодії з веб-додатком

Операції	**Взаємодія з сервісами**	**Опис**
createAuthServiceListener(params) *createWebServerListener(params)*		Ініціалізація сервісу.
login(login, password) *validateUserSession(userSessionId)* *validateToken(token)*		Керування сесіями користувачів.
service()		Перевірка протермінування сесій.

Для інтеграції мікросервісу з Телеграм чат-ботом використовуватиметься бібліотека python-telegram-bot, яка дозволяє чат-боту взаємодіяти з Telegram Bot API (таблиця 3.4). У цьому випадку чат-бот виконуватиме наступні функції:

Надсилання веб-посилання на форму: Користувач отримує посилання на веб-форму для заповнення резюме.

Завантаження резюме: Після заповнення форми користувач завантажує свій файл (резюме) через чат-бот.

Обробка файлів: Бот перевіряє отриманий файл (формат, розмір) і передає його на сервер для зберігання або подальшої обробки.

Мікросервіс оброблятиме запити на завантаження файлів та взаємодію з чат-ботом. Коли користувач завантажує резюме через чат-бот, файл передається на сервер, де зберігається в базі даних або файловій системі для подальшого використання. Веб-форма дозволяє користувачеві ввести свої дані, включаючи особисті дані, досвід роботи, навички та іншу інформацію, необхідну для складання резюме.

Таблиця 3.4 – API мікросервісу файлів

Операції	Взаємодія з сервісами	Опис
createWebServerListener(params) *createWebSocketsListener(params)* *createFileServiceListener(params)* *createAuthServiceConnection(params)*		Ініціалізація сервісу.
createPhoto(photo) *getPhotoById(photoId)* *updatePhoto(photo)* *createDocument(document)* *getDocumentById(documentId)* *updateDocument(document)*		Основні між сервісні операції.
createPhoto(photo) *getPhotoInline(photoId)* *getPhotoAttachment(photoId)* *updatePhoto(photo)*	Сервіс авторизації.	*HTTP* та *WebSockets API.*

Після заповнення та надсилання форми дані користувача зберігаються в базі даних, а резюме автоматично надсилається через API для подальшої обробки в чат-боті. Після того як файл завантажений, він зберігається на сервері або в хмарному сховищі. Мікросервіс також може бути налаштований для обробки метаданих файлів, таких як розмір файлу, тип файлу та дата завантаження. Це дозволяє адмініструвати файли та надавати доступ до них в подальшому.

Однією з ключових складових системи чат-бота для пошуку вакансій є належне управління повідомленнями між чат-ботом та системою керування. Обмін повідомленнями – це основна функція будь-якого месенджера, тому важливо розробити API, яке забезпечить ефективну та безперебійну комунікацію між користувачем та ботом. У випадку з чат-ботом вакансій, цей процес буде включати обробку текстових повідомлень, а також можливість надсилання різноманітних медіафайлів, таких як фотографії та документи, що відповідають за передачу резюме або іншої важливої інформації.

API сервісу повідомлень буде відповідати за обробку різних типів даних, які надсилаються від користувача до чат-бота та навпаки (табл. 3.5). Основними типами повідомлень, що підтримуються в чат-боті вакансій, є:

- **текстові повідомлення**: Бот має здатність отримувати текстові запити від користувача (наприклад, "Пошук вакансій в IT") та надсилати відповіді у вигляді тексту з інформацією про відповідні вакансії;
- **файли**: У разі, коли користувач хоче завантажити своє резюме, бот повинен підтримувати функціональність завантаження та збереження файлів. Це будуть документи у форматі PDF;

Для того щоб чат-бот коректно працював з різними типами повідомлень, API має підтримувати такі функції:

- **текстові повідомлення**: API повинно мати можливість надсилати і отримувати прості текстові повідомлення. Це може бути як інформація про доступні вакансії, так і запити до користувача (наприклад, запит про вибір категорії вакансій чи місця роботи);
- розробка API сервісу повідомлень для чат-бота пошуку вакансій є важливим етапом у створенні функціоналу для ефективної взаємодії з користувачами. Здійснивши правильне налаштування підтримки різних типів повідомлень – текстових, медіа, голосових та файлових – можна значно підвищити зручність користувачів, а також забезпечити гнучкість для подальшого розвитку чат-бота. Важливо також пам'ятати про безпеку даних і підтримку високих стандартів захисту інформації, щоб забезпечити конфіденційність та цілісність переданих даних.

Таблиця 3.5 – API мікросервісу повідомлень

Операції	Взаємодія з сервісами	Опис
createWebServerListener(params) *createWebSocketsListener(params)* *createMessageServiceListener(params)* *createAuthServiceConnection(params)* *createFileServiceConnection(params)*		
createTgTextMessage(message) *createTgPhotoMessage(message)* *createTgDocumentMessage(message)*		Основні між сервісні операції.
createTgTextMessage(message) *createTgPhotoMessage(message)* *createTgDocumentMessage(message)* *updateTgTextMessage(message)* *updateTgPhotoMessage(message)* *updateTgDocumentMessage(message)* *getTgMessages()* *getTgMessagesCount()* *getTgMessagesByParams(params)* *syncTgMessagesPages()* *syncTgMessages(params)*	Сервіс авторизації. Сервіс *Telegram Bot API*. Сервіс файлів.	*HTTP* та *WebSockets API*.
tgTextMessageCreated(message) *tgPhotoMessageCreated(message)* *tgDocumentMessageCreated(message)* *tgTextMessageUpdated(message)* *tgPhotoMessageUpdated(message)* *tgDocumentMessageUpdated(message)*		Події *WebSockets*.

3.2 Архітектурні рішення управління логістичною системою підприємства

В умовах стрімкого розвитку технологій пошуку вакансій, автоматизація процесу збору інформації про пропозиції роботи стала важливою складовою ефективного кар’єрного консультування. Одним із найпопулярніших ресурсів для пошуку роботи є LinkedIn та Work.ua. Спростити пошук інформації на сайтах з вакансіями може масштабований спосіб збору інформації, її організації та аналізу, а саме web-scraping.

Web Scraper (від англ. Scraping – «вишкрібання», веб-збирання або витягнення вебданих) являє собою перетворення у структуровані дані інформації з веб-сторінок, які призначені для перегляду людиною за допомогою браузера. Як правило, виконується за допомогою комп'ютерних програм, що імітують поведінку людини в інтернеті, або з'єднуючись з веб-сервером напряму по протоколу HTTP, або управляючи повноцінним веб-браузером. Але буває і скрапінг за допомогою копіювання даних людиною. Це форма копіювання, в якій конкретні дані збираються та копіюються з інтернету [46].

Суть методу полягає в тому, щоб надіслати запит до потрібного ресурсу та отримати у відповідь веб-сторінку. Ресурсом може бути як простий лендінг, так і повноцінна, наприклад, соціальна мережа. Загалом, усе те, що веб-сервер може «віддавати» у відповідь на HTTP-запити (рис. 3.3).

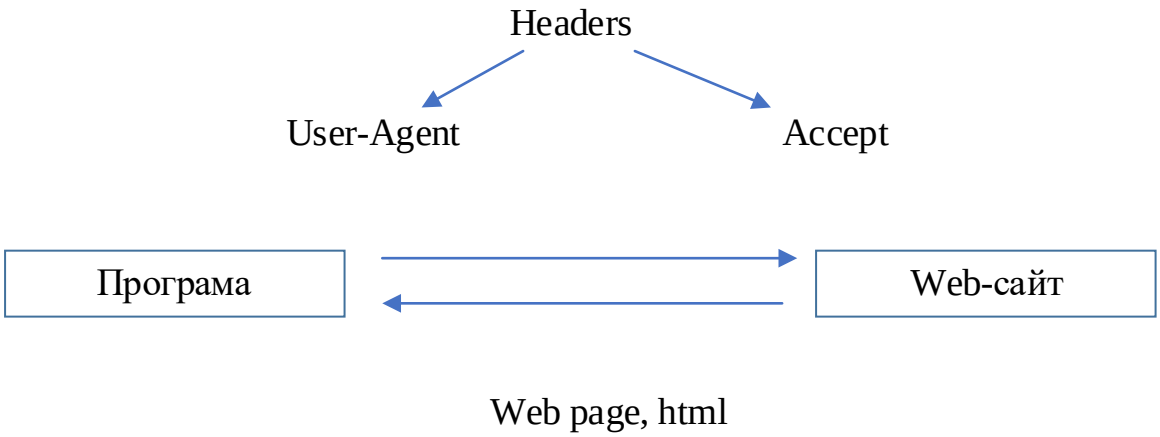

Рисунок 3.3 – Процедура парсінгу сайту

Даний метод має свої переваги та недоліки (таблиця 3.6):

Таблиця 3.6 - Переваги та недоліки Web Scraper [20]

Переваги	Недоліки
Автоматизація роботи сайту	Вимагає подальшого редагування інформації
Працює автономно	Знайдена інформація не унікальна, деколи її публікація без посилання на першоджерело або згоди правовласника порушує авторські і корпоративні права
Миттєве отримання інформації	Налаштування програми проводяться перед запуском і для змін потрібно втручання програміста
Висока точність групування інформації	При зміні структури сайтів-донорів іноді спостерігаються помилки в роботі пошукової машини, що вимагає переналаштування під нові реалії
Скорочує деякий час для пошуку потрібної інформації	Не може повністю замінити роботу людини

Розглянемо особливості реалізації процесу парсингу сайту LinkedIn за допомогою мови програмування Python для збору даних про вакансії з урахуванням заданих критеріїв.

Метою парсингу є автоматичне отримання інформації про вакансії з LinkedIn, використовуючи фільтри, такі як:

- посада (наприклад, "Python Developer");
- локація (місто, країна);
- досвід роботи;
- тип зайнятості (повна, часткова, дистанційна тощо).

Для реалізації парсингу використовуються наступні технології та бібліотеки Python:

- Selenium — для автоматизації веб-додатків і симуляції дій користувача, що дозволяє обійти захист від ботів;

- BeautifulSoup (з бібліотеки bs4) — для обробки та аналізу HTML-коду сторінки;

- Requests — для отримання HTML-контенту веб-сторінки (використовується в разі, якщо доступ до даних не потребує авторизації);

- Pandas — для зберігання та обробки отриманих даних у зручному табличному форматі.

Через високий рівень захисту LinkedIn обов'язковим етапом є авторизація. За допомогою Selenium виконується автоматичне введення логіна та пароля, що імітує дії користувача. Це дозволяє отримати доступ до захищеного контенту.

LinkedIn активно використовує захист від автоматизованого збору даних (CAPTCHA, блокування IP). Для обхідного шляху можуть використовуватися:

- використання ротації проксі-серверів;
- симуляція поведінки реального користувача (наприклад, затримка між запитами).

Щоб обійти виявення антиботів LinkedIn використаємо ScraperAPI, який дозволить:

- керувати ротацією IP;
- керуйте пулом проксі-серверів;
- обробляти CAPTCHA;
- надішліть належні заголовки.

Для використання ScraperAPI створимо безкоштовний обліковий запис ScraperAPI, щоб мати доступ до ключа API.

Таким чином, HTTP-запит буде оброблено сервером ScraperAPI. Він змінюватиме IP-адреси після кожного запиту (або за потреби) і вибиратиме правильні заголовки на основі років статистичного аналізу та машинного навчання.

Для проведення парсінгу даних використаємо низку бібліотек :

import csv

import requests

from bs4 import BeautifulSoup

from selenium import webdriver

Бібліотека requests Python використовується для надсилання HTTP-запитів до вказаної URL-адреси. Запити Python забезпечують вбудовані функції для керування як запитом, так і відповіддю.

Модуль requests Python має кілька вбудованих методів для виконання HTTP-запитів до вказаної URL-адреси за допомогою запитів GET, POST, PUT, PATCH або HEAD. Запит HTTP призначений або для отримання даних із зазначеного URI, або для надсилання даних на сервер. Він працює як протокол запит-відповідь між клієнтом і сервером.

Beautiful Soup — це бібліотека, яка дозволяє легко отримувати інформацію з веб-сторінок. Бібліотека знаходиться на вершині синтаксичного аналізатора HTML або XML, надаючи ідіоми Pythonic для ітерації, пошуку та модифікації дерева аналізу [17].

Модуль csv реалізує класи для читання та запису табличних даних у форматі CSV [18].

Для надсилання запитів створимо змінну, що містить нашу початкову URL-адресу, і передамо її методу requests.get(). Потім збережемо повернутий HTML у змінній під назвою “response”, щоб створити наш об’єкт Python.

url = 'https://www.linkedin.com/jobs-guest/jobs/api/seeMoreJobPostings/search?keywords=Product%20Management&location=San%20Francisco%20Bay%20Area&geoId=90000084&trk=public_jobs_jobs-search-bar_search-submit&position=1&pageNum=0&start=0'

response = requests.get(url)

Щооб полегшити навігацію за допомогою селекторів CSS створимо новий об’єкт Beautiful Soup, передавши response.content як перший аргумент, а метод аналізатора – як другий аргумент:

soup = BeautifulSoup(response.content,'html.parser')

Щоб отримані конкретні дані використаємо той факт, що усередині кожного елемента <li> є div із класом, на який ми можемо орієнтуватися. Щоб отримати доступ до даних усередині, давайте створимо нову змінну, щоб вибрати всі ці та створимо цикл for за допомогою селекторів CSS:

```
for job in jobs:
job_title = job.find('h3', class_='base-search-card__title').text.strip()
job_company = job.find('h4', class_='base-search-card__subtitle').text.strip()
job_location = job.find('span', class_='job-search-card__location').text.strip()
job_link = job.find('a', class_='base-card__full-link')['href']
```

За межами сновної функції відкриємо новий файл на запис за допомогою методу writerow() та створимо ряд заголовків.

```
file = open('linkedin-jobs.csv', 'a')
writer = csv.writer(file)
writer.writerow(['Title', 'Company', 'Location', 'Apply'])
```

Отримаємо наступний результат запису у файл (рис. 3.4).

Розглянемо процес парсінгу даних з сайту Work.ua. Для цього використаємо запит

```
URL_page = "https://www.work.ua/jobs-poltava/?setlp=ua"

r = requests.get(URL_page)
```

Аналіз html кода дозволяє зробити висновок про місце розміщення в тегах необхідної інформації (рис. 3.5):

Title	Company	Location	Apply
Title	Company	Location	Apply
Title	Company	Location	Apply
Python Developer	Tata Elxsi	Bengaluru, Karnataka, India	https://in.linkedin.com/jobs/view/python-developer-at-tata-elxsi-4117383959?position=1&pageNum=0&refId=rROFmxe3c85K52X%2FzJnQGQ%3D%3D&tracking
Python Developer	LTIMindtree	Bengaluru, Karnataka, India	https://in.linkedin.com/jobs/view/python-developer-at-ltimindtree-4117932863?position=2&pageNum=0&refId=rROFmxe3c85K52X%2FzJnQGQ%3D%3D&tracking
Python developer	Pakistan Hiring	Lahore, Punjab, Pakistan	https://pk.linkedin.com/jobs/view/python-developer-at-pakistan-hiring-4118514504?position=3&pageNum=0&refId=rROFmxe3c85K52X%2FzJnQGQ%3D%3D&tracking
****** *********	******** ****	****, **********, *****	https://in.linkedin.com/jobs/view/python-developer-at-deutsche-bank-4118585973?position=4&pageNum=0&refId=rROFmxe3c85K52X%2FzJnQGQ%3D%3D&tracking
*******_ **** ********	***** *********	*********, *********, *****	https://in.linkedin.com/jobs/view/analyst-data-analysis-at-tesco-bengaluru-411852163position=5&pageNum=0&refId=rROFmxe3c85K52X%2FzJnQGQ%3D%3D&tracking
****** *********	*****	*********, *********, *****	https://in.linkedin.com/jobs/view/python-developer-at-wipro-4117936043?position=6&pageNum=0&refId=rROFmxe3c85K52X%2FzJnQGQ%3D%3D&tracking
****** ********* *******	**********	****, **********, *****	https://in.linkedin.com/jobs/view/python-developer-trainee-at-primefactorx-411735407position=7&pageNum=0&refId=rROFmxe3c85K52X%2FzJnQGQ%3D%3D&tracking
Python Developer	Tata Elxsi	Bengaluru, Karnataka, India	https://in.linkedin.com/jobs/view/python-developer-at-tata-elxsi-4117383959?position=1&pageNum=0&refId=8Vlop5CdDm4la6BbFpcZfQ%3D%3D&trackingId=FG

Рисунок 3.4 – Результат парсінгу вакансій

Рисунок 3.5 – Код веб сторінки з інформацією про вакансію

Для запису знайдених вакансій за запитом використаємо наступну функцію:

```
def parse(url = URL_page):
    result_list = {'href': [], 'title': [], 'about': []}
    r = requests.get(url)
```

```
    soup = bs(r.text, "html.parser")
    vacancies_names = soup.find_all('h2', class_='add-bottom-sm')
    vacancies_info = soup.find_all('p', class_='overflow')
    for name in vacancies_names:
        result_list['href'].append('https://www.work.ua'+name.a['href'])
        result_list['title'].append(name.a['title'])
    for info in vacancies_info:
        result_list['about'].append(info.text)
    return result_list
df = pd.DataFrame(data=parse())
df.to_csv('vacancy.csv')
```

3.3 Реалізація API зовнішнього сервісу взаємодії з веб-додатком

У сучасному світі автоматизація процесів пошуку роботи стає все більш актуальною, особливо з огляду на розвиток технологій штучного інтелекту та інтеграцію інструментів комунікації, таких як чат-боти. Telegram, як одна з найбільш популярних платформ обміну повідомленнями, надає широкі можливості для створення інтерактивних рішень, зокрема для пошуку вакансій.

Одним із ключових елементів роботи чат-бота з пошуку роботи є веб-анкета, яка дозволяє ефективно збирати дані користувачів, необхідні для підбору відповідних вакансій. Веб-анкета повинна бути зручною, інтуїтивно зрозумілою та відповідати сучасним вимогам до захисту і обробки даних.

У цьому підпункті розглядаються основні вимоги до створення веб-анкети, включаючи функціональні та нефункціональні характеристики, визначення ключових полів, необхідних для пошуку вакансій, а також забезпечення зручності користувацького інтерфейсу. Це дозволяє сформувати чітке бачення структури веб-анкети, її призначення та способів взаємодії з Telegram-чат-ботом.

Концепція інтеграції Telegram-чат-ботів із веб-сторінками полягає у забезпеченні двосторонньої взаємодії між ботом і веб-додатками, які розгорнуті у зовнішньому середовищі. Це дозволяє значно розширити можливості чат-бота завдяки використанню веб-технологій, таких як HTML, CSS, JavaScript, і забезпечити доступ до веб-ресурсів у межах функціоналу бота.

У Telegram інтеграція з веб-сторінками використовується для вирішення різних завдань:

1. Відображення веб-контенту, коли чат-бот може передавати користувачам посилання на веб-сторінки з додатковою інформацією, наприклад, статтями, каталогами товарів або інструкціями.

2. Взаємодія з веб-додатками - Telegram API дозволяє чат-боту передавати дані до веб-додатків і отримувати результати у відповідь. Це може бути реалізовано для збору форм, авторизації користувачів або виконання транзакцій.

3. Інтеграція зі сторонніми сервісами. Бот може виступати посередником між користувачем і зовнішніми сервісами, такими як CRM-системи, платіжні платформи чи соціальні мережі, для обробки запитів або обміну даними.

4. Динамічний контент. Чат-бот може автоматично генерувати динамічні веб-сторінки на основі запитів користувачів, наприклад, форму зворотного зв’язку або сторінку для підтвердження замовлення.

При цьому інтеграція Telegram-чат-ботів із веб-сторінками дозволяє реалізувати наступні функції:

1. Авторизація користувачів: Використання веб-сторінок для аутентифікації або підтвердження ідентифікації користувача через сторонні сервіси. Наприклад, бот надсилає посилання на сторінку входу в обліковий запис із подальшим поверненням результатів у бот.

2. Форми зворотного зв’язку: Веб-сторінки можуть бути створені для збору заявок, коментарів чи інших форм даних, які передаються назад у бот для обробки.

3. Обробка транзакцій: За допомогою інтеграції із платіжними системами через веб-сторінки користувачі можуть здійснювати покупки, сплачувати рахунки або підписки.

4. Відображення динамічного контенту: Наприклад, бот може надавати доступ до каталогу товарів, сторінок із результатами пошуку вакансій або інтерактивних інструкцій у вигляді веб-сторінок.

Створення веб-анкети вимагає вибору відповідних технологій, які забезпечать зручність використання, функціональність та естетичний вигляд інтерфейсу. Основними інструментами для розробки є HTML, CSS та JavaScript, кожен із яких відіграє важливу роль у побудові веб-форм. Крім базових технологій, до розгляду беруться популярні фреймворки та бібліотеки, які значно спрощують процес розробки й дозволяють реалізовувати складні інтерактивні елементи. У наведеній порівняльній таблиці представлені ключові характеристики цих технологій із метою виявлення їх переваг та недоліків, що допоможе обрати оптимальний стек для реалізації веб-анкети (табл. 3.7).

Таблиця 3.7 – Порівняльна таблиця технологій для створення веб-анкети

Технологія		Переваги	Недоліки
HTML (HyperText Markup Language)	Створення структури веб-сторінки, визначення елементів форми (поля, кнопки, випадаючі списки).	- Простота у використанні. - Основний будівельний блок веб-сторінок.	- Відсутність інтерактивності. - Потребує CSS і JavaScript для динамічної поведінки та стилізації.
CSS (Cascading Style Sheets)	Додання стилів до HTML-елементів, включаючи кольори, шрифти, відступи, макет.	- Гнучкість у дизайні. - Відокремлення структури (HTML) від стилізації.	- Не забезпечує функціональності або інтерактивності.
JavaScript	Забезпечення інтерактивності та динамічної поведінки веб-анкети.	- Динамічна взаємодія з користувачем (валідація даних, підказки, автозаповнення). - Підтримка AJAX для асинхронної передачі даних.	- Може створювати проблеми з безпекою, якщо використовується неправильно. - Потребує знання мови програмування.

		- Можливість роботи з DOM для маніпуляції елементами сторінки. - Великий набір бібліотек і фреймворків для спрощення роботи (React, Vue, Angular).	
React (JavaScript-фреймворк)	Розробка динамічних компонентів для веб-анкети, таких як багатосторінкові форми.	- Простота створення багаторазових компонентів. - Висока продуктивність завдяки віртуальному DOM.	- Високий поріг входу для новачків. - Потребує налаштування середовища (Webpack, Babel).
Vue.js (JavaScript-фреймворк)	Розробка легких та інтерактивних веб-додатків.	- Простий у використанні для невеликих проєктів. - Підтримує двосторонню прив'язку даних.	- Менша популярність і екосистема порівняно з React.
Angular (JavaScript-фреймворк)	Побудова повноцінних SPA (односторінкових додатків) для складних веб-анкет.	- Можливість роботи з великими проєктами. - Потужна екосистема і підтримка від Google.	- Велика складність для невеликих проєктів. - Високий поріг входу.

JavaScript є оптимальним вибором для розробки веб-анкет завдяки його можливостям забезпечувати інтерактивність, асинхронну передачу даних і адаптивність інтерфейсу. Він дозволяє реалізовувати валідацію даних, підказки користувачам, автозаповнення полів і динамічну зміну елементів без перезавантаження сторінки. Інтеграція з Telegram API можлива через використання AJAX або Fetch API, що сприяє обміну даними в реальному часі. Різноманіття бібліотек і фреймворків, таких як React, Vue.js і Angular, спрощує розробку навіть складних анкет із високою продуктивністю. Крім того, JavaScript забезпечує масштабованість через інтеграцію із серверними технологіями, такими як Node.js, створюючи надійну основу для сучасних інтерактивних веб-рішень.Веб-анкета для збору інформації від кандидатів повинна відповідати низці функціональних та нефункціональних вимог, що забезпечать її ефективність, зручність і безпеку. Розглянемо кожні з них:

1. Функціональні вимоги:

1. Форма збору інформації. веб-анкета повинна забезпечувати введення таких даних:
    - особисті дані кандидата (ім’я, прізвище, контактна інформація);
    - освіта та професійні навички;
    - рівень досвіду роботи та попередні місця працевлаштування;
    - знання мови.
    - Хоббі.
2. Валідація даних - перевірка правильності введених значень (наприклад, формат номеру телефону, обов’язковість заповнення ключових полів).
3. Збереження даних - передача введеної інформації на сервер для обробки або в Telegram-бот через API.
4. Мультиплатформеність – підтримка коректного відображення анкети як на ПК, так і на мобільних пристроях.

2. Нефункціональні вимоги:

1. Продуктивність – швидка робота анкети, включаючи завантаження сторінки та валідацію даних.
2. Безпека – захист персональних даних користувачів під час передачі та збереження.
3. Юзабіліті – інтуїтивний дизайн для швидкого і зручного заповнення форми.
4. Сумісність – підтримка сучасних браузерів (Chrome, Firefox, Safari, Edge).

Розглянемо алгоритм роботи веб-анкети. Веб-форма розроблена для динамічного збирання ключових даних, таких як (рис. 3.6, 3.7):

- ім’я та прізвище користувача;
- дата народження;
- освітній бекграунд (навчальний заклад, спеціальність, періоди навчання);
- попередній професійний досвід;
- знання мови;
- хоббі.

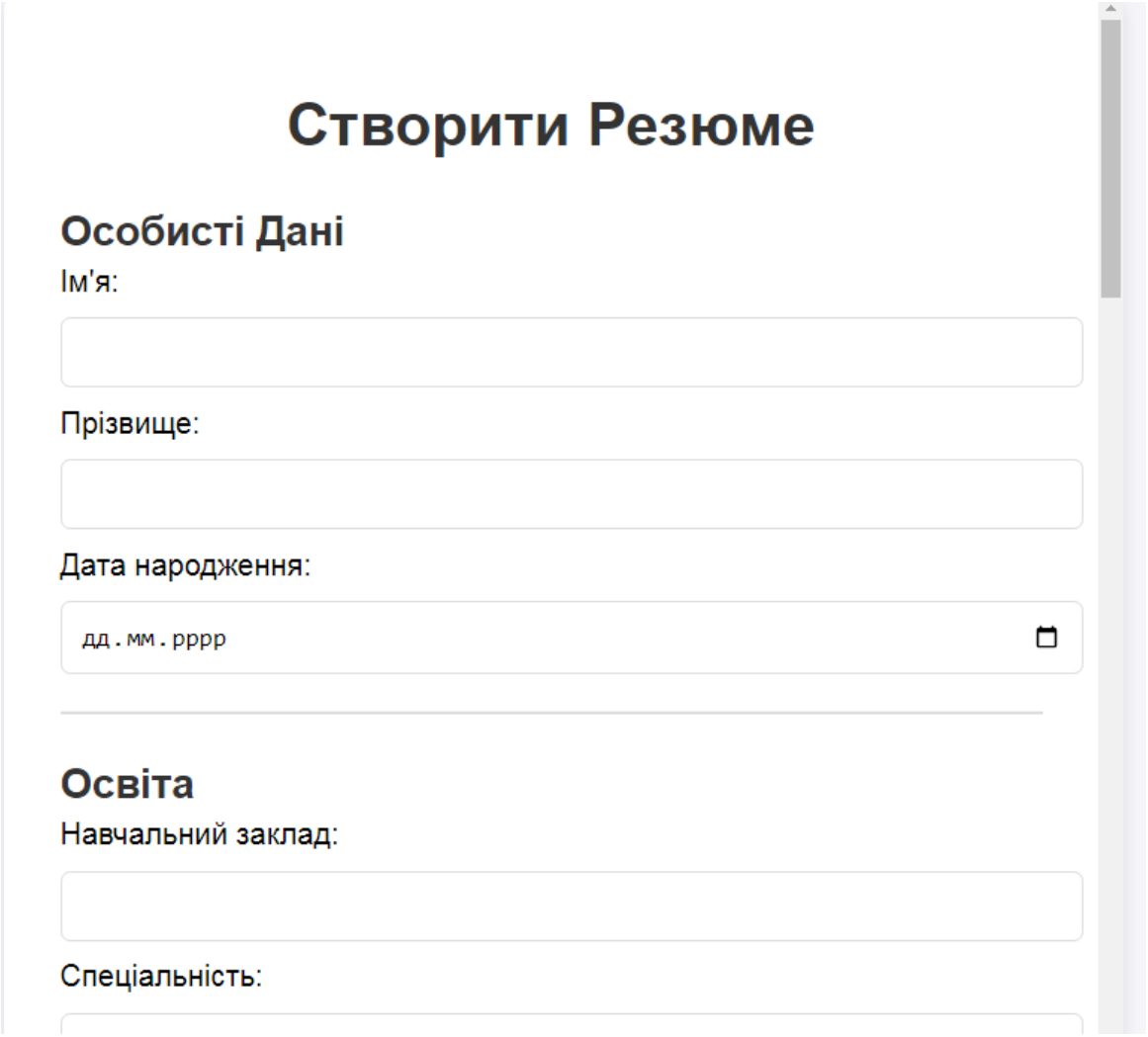

Рисунок 3.6 – Реалізація веб форми анкети для створення CV (част.1)

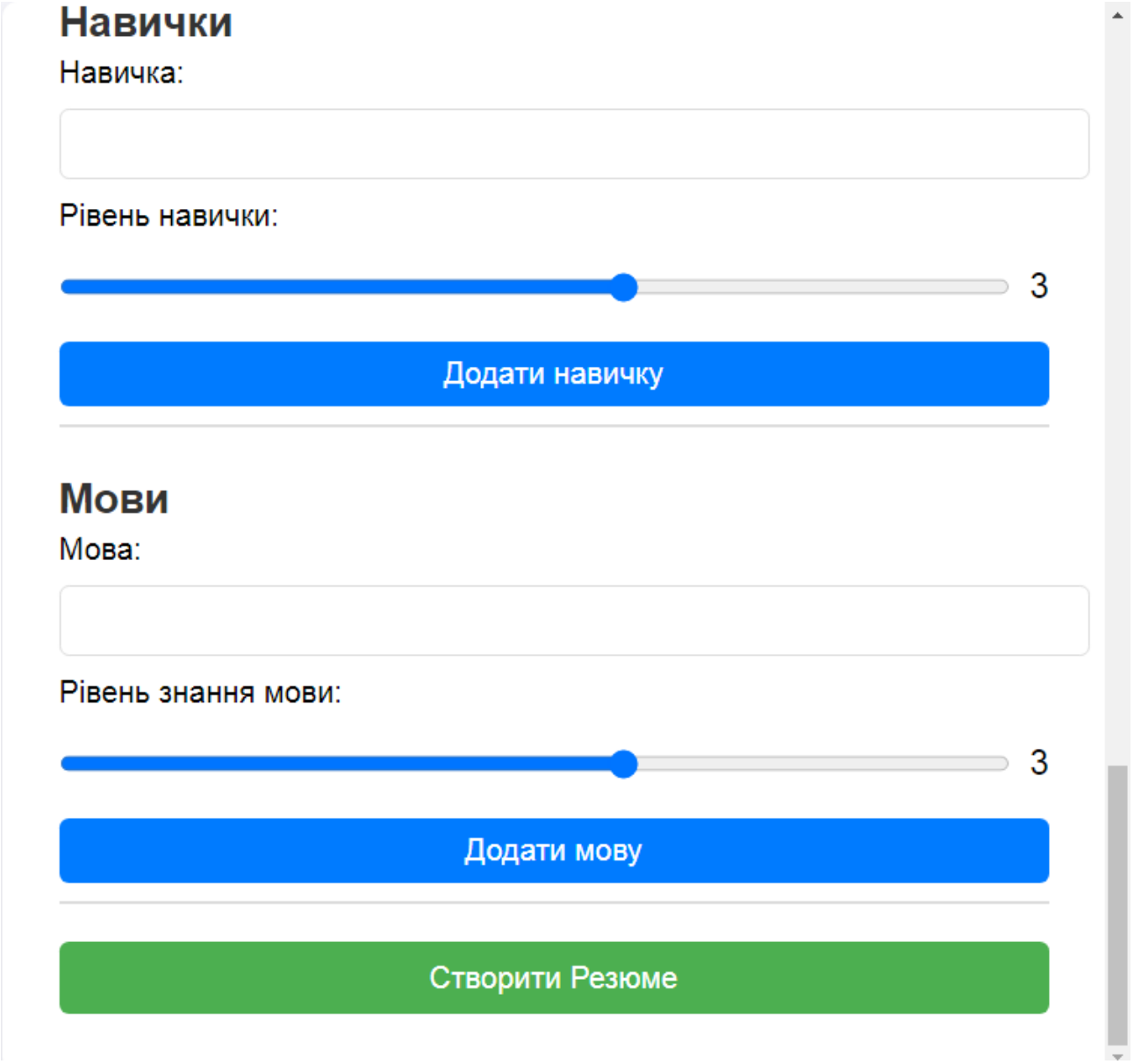

Рисунок 3.7 – Реалізація веб форми анкети для створення CV (част. 2)

Дані, отримані через форму, обробляються за допомогою JavaScript. У формі передбачена можливість динамічного додавання кількох блоків освіти, що робить її зручною для користувачів із різноманітним освітнім досвідом.

Основні етапи обробки форми:

1. Дані з форми отримуються за допомогою DOM-методів (document.getElementById, querySelectorAll).
2. Зібрана інформація форматується у вигляді тексту, придатного для подальшого використання.

3. Після натискання кнопки «Завантажити резюме» сформований текст передається через Telegram API.

Сформований текст резюме надсилається до Telegram за допомогою API запиту. Використовуються такі параметри:

- telegramToken – токен, отриманий під час реєстрації бота;
- chatId – унікальний ідентифікатор користувача, який взаємодіє з ботом;
- Message – текст повідомлення, що містить інформацію з резюме.

Висновки до розділу 3

Розробка серверної частини є одним із ключових етапів створення системи керування чат-ботом, оскільки саме вона забезпечує взаємодію бота з користувачами, обробку запитів та інтеграцію із зовнішніми сервісами. Для створення серверної частини чат-бота в Telegram передбачено реалізацію кількох важливих етапів, серед яких вибір технологій, налаштування середовища, розробка API та забезпечення безпеки й масштабованості. Зокрема, обмін даними між ботом і серверами Telegram відбувається через Telegram Bot API, що дозволяє надсилати й отримувати запити. Функціональні компоненти системи можуть бути представлені у вигляді мікросервісів, що забезпечує гнучкість та зручність у розробці. Важливою складовою є створення веб-API, яке використовується для взаємодії з зовнішніми системами, включно з пошуковими платформами вакансій, такими як Work.ua чи LinkedIn. Забезпечення захисту даних користувачів також є одним із пріоритетів під час розробки системи.

Реалізація таких систем можлива завдяки дотриманню принципів SOLID, які сприяють створенню масштабованого та гнучкого коду. Наприклад, принцип єдиного обов’язку дозволяє кожному компоненту виконувати лише одну задачу, тоді як принцип відкритості для розширення й закритості для змін спрощує додавання нового функціоналу без порушення існуючої архітектури.

Окремо слід відзначити роль автоматизації процесів збору даних про вакансії. Завдяки використанню технологій web-scraping і бібліотек Python, таких як Selenium, BeautifulSoup і Requests, можливо ефективно збирати та

аналізувати дані з веб-ресурсів. Наприклад, автоматичне отримання інформації про вакансії з LinkedIn або Work.ua з урахуванням фільтрів (посада, місце розташування, досвід роботи) дає змогу значно спростити процес пошуку та надання результатів користувачеві.

У контексті взаємодії чат-бота з веб-технологіями, інтеграція з веб-сторінками дозволяє реалізувати динамічні рішення, включаючи відображення контенту, збір даних через веб-форми, обробку транзакцій і передачу результатів користувачу. Вибір технологій, таких як HTML, CSS, JavaScript та сучасні фреймворки, забезпечує створення зручних, функціональних і захищених інтерфейсів для взаємодії з ботом.

Таким чином, розвиток технологій і вдосконалення архітектури чат-ботів відкривають нові можливості для автоматизації процесів, зокрема у сфері пошуку вакансій, що сприяє підвищенню ефективності роботи користувачів і якості їхніх рішень.

ВИСНОВКИ

Отже, результатом даної роботи є розроблена інформаційно-пошукова система пошуку вакасій реалізована у вигляді Телеграм чат-боту. Бекенд частина додатка розроблена з використанням мови програмування Python та Java Script, що робить розроблений додаток ефективним, оптимізованим, зрозумілим.

В ході виконання даної роботи було виконано наступні завдання:

1. Проведено дослідження основних підходів до створення пошукових систем, що застосовуються для підтримки прийняття рішень. Визначено ключові технології та методи, зокрема алгоритми машинного навчання та великі дані, що використовуються для вдосконалення результатів пошуку та персоналізації рекомендацій.

2. Аналіз сучасних пошукових систем показав широкий спектр застосувань, включаючи системи, що інтегрують штучний інтелект для розпізнавання контексту запиту та персоналізації результатів. Визначено архітектурні особливості таких систем, включаючи мікросервісні підходи для масштабованості та інтеграції з різними зовнішніми джерелами інформації.

3. Визначено вимоги до проектованої пошукової системи, що включають необхідність інтеграції з іншими веб-сервісами, забезпечення швидкої обробки запитів та забезпечення високої точності пошукових результатів. Розроблено технічне завдання, яке описує функціональні та нефункціональні вимоги до системи.

4. На основі розроблених вимог проведено проектування пошукової системи для конкретної області застосування, включаючи реалізацію інтерфейсу користувача, серверної частини, а також інтеграцію з API для збору та аналізу зовнішніх даних. Реалізовано функціонал пошуку з використанням бази даних, що дозволяє ефективно обробляти запити користувачів та надавати релевантні рекомендації.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Baechle M., Daurer S., Judt A., Mettler T. Chatbots as a User Interface for Assistive Technology in the Workplace. 2018.

2. Best Coding Languages for Effective AI Chatbot Development. Режим доступу: https://dev.to/hitesh_umaletiya/best-coding-languages-for-effective-ai-chatbot-development-el4. Дата звернення: 22.11.2024.

3. Dahiya M. A Tool of Conversation: Chatbot. IJCSE, 2017. 5:158-161.

4. Idhom M, Alit R, Wahanani H E, and Fauzi A. Implementation System Telegram Bot for Monitoring Linux Server. Intl. Conf. on Sci. and Tech. (Bali) vol 1 (Amsterdam: Netherland/Atlantis Highlights in Engineering) p. 1089-1093.

5. Idhom M, Alit R, Wahanani H E, and Fauzi A. Implementation System Telegram Bot for Monitoring Linux Server. Intl. Conf. on Sci. and Tech. (Bali) vol 1 (Amsterdam: Netherland/Atlantis Highlights in Engineering) p. 1089-1093.

6. Jayant Madhavan, David Ko, Lucja Kot, Vignesh Ganapathy, Alex Rasmussen, and Alon Y. Halevy. Google’s deep web crawl. PVLDB, 1(2):1241–1252, 2008.

7. Johnson S. Digital Communication Skills: Navigating the Modern Landscape. Routledge, 2021. P. 60–73.

8. Najork M. Web Crawler Architecture. Encyclopedia of Database Systems, 2017. P. 1–4.

9. Paech, Barbara & Dutoit, Allen & Kerkow, Daniel & Knethen, Antje. Functional requirements, non-functional requirements, and architecture should not be separated. A position paper. 2002.

10. PyCharm 3.0 Community Edition source code now available. Режим доступу: https://blog.jetbrains.com/pycharm/2013/10/pycharm-3-0-community-edition-source-code-now-available/. Дата звернення: 24.11.2024.

11. Rahman A M, Mamun A A, and Islam A. Programming challenges of Chatbot: Current and Future Prospective. IEEE R10HTC(Dhaka), vol 2018-January (R10-HTC:Dhaka) p. 76.

12. robots.txt. Режим доступу: http://www.robotstxt.org/. Дата звернення: 12.11.2024.

13. Sitemaps. Режим доступу: http://www.sitemaps.org. Дата звернення: 15.11.2024.

14. Top 6 Programming Languages for Chatbot Development. Режим доступу: https://www.codecademy.com. Дата звернення: 22.11.2024.

15. What is AI Chatbot Development and How Is It Done? Режим доступу: https://www.eliftech.com/insights/chatbot-development-guide. Дата звернення: 20.11.2024.

16. Zennaro M, Rainone M, and Pietrosemoli E. Radio Link Planning made easy with a Telegram Bot. Intl. Conf. on Smart Objects and Tech. for Social Good (Venice), vol. 2016 (Springer) p. 1-6.

17. Бібліотека beautifulsoup4. Режим доступу: https://pypi.org/project/beautifulsoup4/. Дата звернення: 26.11.2024.

18. Бібліотека csv. Режим доступу: https://docs.python.org/3/library/csv.html. Дата звернення: 26.11.2024.

19. Бібліотека python-telegram-bot. Режим доступу: https://pypi.org/project/python-telegram-bot/. Дата звернення: 24.11.2024.

20. Вакуленко Ю. Застосування методу парсингу для ефективного пошуку інформації у дослідницькій діяльності / Ю. Вакуленко, О. Щербина. Вісник студентського наукового товариства ДонНУ імені Василя Стуса, 2019, том 2. № 11, с. 153–156.

21. Вовк Н. Архівні інформаційно-пошукові системи: шляхи оптимізації пошуку текстової інформації. Бібліотекознавство. Документознавство. Інформологія. 2018. № 3. С. 37-42.

22. Говорущенко Т. О., Гнатчук Є. Г., Савчук О. М. Концепція інформаційно-пошукової системи (на основі онтологій) для галузі медичного права. Комп’ютерні системи та інформаційні технології. 2020. № 2. С. 5-8.

23. Діброва І. С., Фриз І. В. Розробка телеграм-бота та його застосування. Вісник студентського наукового товариства ДонНУ імені Василя Стуса. Том 1 № 16. 2024. С. 123-127.

24. Енциклопедія Сучасної України. Т. 1: «А» / Гол. редкол.: І.М. Дзюба, А.І. Жуковський, М.Г. Железняк та ін.; НАН України, НТШ. Київ: Інститут енциклопедичних досліджень НАН України, 2001. 823 с.

25. Збанацька О. М. Розвиток інформаційно-пошукових мов у пошукових системах бібліотек і архівів: монографія. Київ: НАКККіМ, 2017. 353 с.

26. Інформаційно-пошукові системи // Соціальна інформатика та технології. Режим доступу: http://abramchuk-inf.blogspot.com/p/blog-page_6023.html. Дата звернення: 10.11.2024.

27. Кириченко О. ОСОБЛИВОСТІ АРХІТЕКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ЗБОРУ ТА АНАЛІЗУ СТАТИСТИЧНОЇ ІНФОРМАЦІЇ В ГЛОБАЛЬНІЙ МЕРЕЖІ. Information Technology: Computer Science, Software Engineering and Cyber Security. 2023. С. 107-112. DOI: 10.32782/IT/2023-2-13.

28. Офіційний сайт Go. Режим доступу: https://go.dev/. Дата звернення: 23.11.2024.

29. Офіційний сайт Java. Режим доступу: https://www.java.com/en/. Дата звернення: 23.11.2024.

30. Офіційний сайт JavaScript. Режим доступу: https://developer.mozilla.org/ru/docs/Web/JavaScript. Дата звернення: 23.11.2024.

31. Офіційний сайт PyCharm. Режим доступу: https://www.jetbrains.com/pycharm/. Дата звернення: 24.11.2024.

32. Офіційний сайт Python. Режим доступу: https://www.python.org/. Дата звернення: 23.11.2024.

33. Офіційний сайт Ruby. Режим доступу: https://www.ruby-lang.org/en. Дата звернення: 23.11.2024.

34. Офіційний сайт Telegram. Режим доступу: https://core.telegram.org/bots/features. Дата звернення: 18.11.2024.

35. Паламар А., Дьомін В., Волоський В. Програмне забезпечення комп’ютерної системи для моніторингу стану пристроїв безперебійного живлення. Матеріали X науково-технічної конференції «Інформаційні моделі, системи та технології», 7–8 грудня 2022 року. Т.: ТНТУ, 2022. С. 83. (Комп’ютерні системи та мережі).

36. Принципи SOLID — що це? Кейси та поради, як їх застосовувати. Режим доступу: https://journal.gen.tech/post/principi-solid-sho-ce-ta-yak-yih-zastosovuvati-kejsi-ta-porady. Дата звернення: 25.11.2024.

37. Принципи SOLID. Навіщо вони потрібні програмістам та як з ними працювати. Режим доступу: https://campus.epam.ua/ua/blog/602. Дата звернення: 25.11.2024.

38. Сухий О. Л., Міленін В. М., Тарадайнік В. М. Алгоритми пошуку в інформаційних системах: методичні рекомендації. К., 2015. 2,0 д.а.

39. Томашевський О. М., Цегелик Г. Г., Вітер М. Б., Дудук В. І. Інформаційні технології та моделювання бізнес-процесів. Навч. посіб. К.: «Видавництво «Центр учбової літератури», 2012. 296 с.

40. Цифрова економіка: підручник / Т. І. Олешко, Н. В. Касьянова, С. Ф. Смерічевський та ін. К.: НАУ, 2022. 200 с.

41. Чи можна використовувати API Telegram? Режим доступу: https://telegramm.com.ua/pytannya-ta-vidpovidi/chimozhna-vykorystovuvaty-api-telegram/. Дата звернення: 20.11.2024.

42. Як створити чат-бот. Режим доступу: https://www.solulab.com/how-to-create-a-telegram-bot/. Дата звернення: 19.11.2024.

43. Яка в Telegram мова програмування? Режим доступу: https://lemon.school/blog/yaka-u-telegram-movaprogramuvanya. Дата звернення: 20.11.2024.

44. Офіційний сайт PHP. Режим доступу: https://www.php.net/. Дата звернення: 20.11.2024.

45. Офіційний сайт C#. Режим доступу: https://dotnet.microsoft.com/ru-ru/languages/csharp. Дата звернення: 21.11.2024.

46. Парсинг html-сайтів для споживача. Парсинг даних для використання підприємством. Режим доступу: parsing.valemak.com/ru/what-why-how/stages-of-parsing. Дата звернення: 01.12.2024.

ДОДАТОК А

ВИХІДНІ КОДИ ПРОГРАМНОГО ЗАСОБУ