

Имя пользователя:
ДН 105 Дмитро Крицкий

ID проверки:
1015693962

Дата проверки:
26.06.2023 18:58:07 EEST

Тип проверки:
Doc vs Library

Дата отчета:
26.06.2023 18:59:28 EEST

ID пользователя:
43712

Название файла: B10

Количество страниц: 82 Количество слов: 12012 Количество символов: 92573 Размер файла: 1.56 MB ID файла: 1015337787

2.05% Совпадения

Наибольшее совпадение: 1.01% с источником из Библиотеки (ID файла: 1015324745)

Поиск совпадений с Интернетом не производился

2.05% Источники из Библиотеки

97

Страница 84

0% Цитат

Не найдено ни одной цитаты

Исключение списка библиографических ссылок выключено

0% Исключений

Нет исключенных источников

ЗМІСТ

ВСТУП	6
1 ПЕРЕДПРОЕКТНІ ДОСЛІДЖЕННЯ	8
1.1 Опис предметної області, що підлягає автоматизації	8
1.2 Аналіз прототипів системи	9
1.3 Обґрунтування вибору інструментального середовища для розробки програмного забезпечення	15
1.4 Обґрунтування вибору технічної платформи, що розробляється	17
1.5 Задачі дипломного проекту	19
2 ДЕТАЛІЗАЦІЯ ЗАДАЧІ АВТОМАТИЗАЦІЇ	24
2.1 Аналіз 1-го рівня деталізації	26
2.2 Аналіз 2-го рівня деталізації	27
2.3 Аналіз 3-го рівня деталізації	30
2.4 Аналіз 4-го рівня деталізації	32
2.5 Висновок до розділу	34
3 АЛГОРИТМИ РОЗВ'ЯЗАННЯ ЗАДАЧ	36
4 КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	41
4.1 Архітектура програмного забезпечення	41
4.2 Інформаційний простір системи	44
4.3 Інтерфейс користувача	49
5 ТЕСТУВАННЯ І ДОСЛІДНА ЕКСПЛУАТАЦІЯ СИСТЕМИ	56
5.1 Функціональне тестування	56
5.2 Тестування взаємодії з користувачем	72
5.3 Тестування безпеки	74
5.4 Тестування продуктивності	76
5.5 Тестування на сумісність	77
5.6 Тестування помилок	78
ВИСНОВКИ	80
ПЕРЕЛІК ПОСИЛАНЬ	85

ВСТУП

В епоху, яка характеризується розповсюдженням цифрового контенту, люди стають завалені невинним потоком новинних статей, публікацій у блогах і оновлень у соціальних мережах. Швидкий розвиток технологій і поширення онлайн-платформ кардинально змінили спосіб доступу до інформації. Однак ця безпрецедентна доступність викликає унікальний набір проблем, які вимагають ретельного розгляду.

Актуальність наукової роботи полягає у вирішенні суттєвої проблеми, пов'язаної з широким впливом пошукових систем та новинних платформ. Цю проблему, відому як «doomscrolling», часто не помічають, але вона має серйозні наслідки для людей. Такі платформи, як Google, Yandex, YouTube, SMITwo, CNN та інші, в основному зосереджені на приверненні уваги користувачів і розміщенні реклами, що сприяє зростанню проблеми doomscrolling. Це явище призводить до перевантаження інформацією, постійної втоми, порушень сну, підвищеного стресу, психологічних розладів. Це негативно впливає на продуктивність, погіршує якість сну та перешкоджає здатності критичного мислення.

Метою нашого проекту є розробка ресурсу, який допоможе користувачам уникнути негативних наслідків думскролінгу та перевантаження інформацією. Ми створюємо систему, яка надає персоналізовані та відповідні рекомендації виключно на основі пошукових запитів користувачів, забезпечуючи конфіденційність даних. Завдяки цьому наше рішення допоможе користувачам уникати цікавих новин і забезпечить конфіденційність їхніх даних.

Ця робота пропонує альтернативу традиційним пошуковим системам і платформам новин, які прагнуть забрати увагу користувачів. Наша система допоможе відфільтрувати інформаційний шум, заощадити час і надати лише актуальну інформацію, яка цікавить користувачів. Важливо підкреслити, що

аспект персоналізації спеціально зосереджений на боротьбі з doomscrolling, і він включає лише необхідні функції для досягнення цієї мети.

Метою цього дослідження є розробка та впровадження бота Telegram, а також веб-сайту, для автоматичного збору новинних статей, визначення ключових слів і надання персоналізованих рекомендацій користувачам для боротьби з doomscrolling.

Предметом цього дослідження є розробка та впровадження бота Telegram, який автоматизує процес збору новин, визначає ключові слова та пропонує персоналізовані рекомендації.

За допомогою методів обробки природньої мови, розроблений Telegram бот буде здатний ефективно збирати новинні статті та екстрагувати релевантну інформацію з них для подальшого використання. Цей підхід дозволить користувачам залишатися в курсі подій, не потрапляючи у пастку нескінченного прокручування стрічки.

У рамках цієї дипломної роботи буде представлено всебічне дослідження методології, використаної для розробки бота Telegram. Буде детально розглянуто архітектурний дизайн, включаючи базові алгоритми та методи обробки даних. Крім того, буде вивчено процес впровадження, а також показники оцінки, які використовуються для оцінки продуктивності бота. Очікувані результати цього дослідження мають потенціал для розвитку персоналізованих систем рекомендацій новин, надаючи практичні ідеї та стратегії для пом'якшення згубних наслідків думскролінгу.

1 ПЕРЕДПРОЕКТНІ ДОСЛІДЖЕННЯ

1.1 Опис предметної області, що підлягає автоматизації

У цьому дослідному проекті ми вирішили зайнятися розробкою програмного забезпечення, яке збиратиме новинні статті та робитиме їх токенизацію. Головна мета полягає в автоматизації процесу збору новин з різних джерел, а також вилучення корисної інформації для подальшого аналізу та використання.

Автоматизація цього завдання потрібна з кількох причин. По-перше, зараз в Інтернеті є величезна кількість новинних статей, а користувачі все більше прагнуть персоналізованої інформації. Це ускладнює ручний пошук та фільтрацію контенту. Автоматизація збору статей дозволить ефективно отримувати матеріали з різних джерел, забезпечуючи широке охоплення тем і економію часу для користувачів.

По-друге, токенизація статей відіграє важливу роль в організації та аналізі контенту. Шляхом отримання ключової інформації, такої як ключові слова, програмне забезпечення полегшить створення персоналізованих рекомендацій. Це дозволить користувачам отримувати новини, що відповідають їхнім конкретним запитам, та підвищить загальний інтерес та задоволення.

Крім того, автоматизація збору та токенизації статей також допоможе вирішити проблему нескінченного прокручування стрічок новин. Хоча основна увага програмного забезпечення не зосереджена на персоналізованій доставці контенту на основі переваг користувачів та історії пошуку, можливість надання відповідних рекомендацій на основі пошукових запитів може допомогти користувачам уникнути цього нескінченного циклу та більш ефективно знаходити цінні статті новин.

Переваги автоматизації в цій галузі сягають далеко за межі персоналізованих рекомендацій та боротьби з нескінченим прокручуванням.

Завдяки використанню методів обробки природної мови та автоматизації, програмне забезпечення може забезпечити точність та послідовність вилучення даних, знизити ймовірність людських помилок та упереджень. Воно також здатне обробляти великий обсяг новинних статей у стислі терміни, надаючи користувачам актуальну та всебічну інформацію.

Отже, автоматизація збору новин та токенизації є життєздатним та корисним підходом у рамках даного дипломного проекту. Завдяки автоматизації цих процесів програмне забезпечення зможе пропонувати персоналізовані рекомендації, покращувати ефективність споживання новин та допомагати вирішувати проблеми, пов'язані з нескінченним прокручуванням.

1.2 Аналіз прототипів системи

У цьому розділі ми розберемо три прототипи системи: пошукову систему Google, агрегатор новин SMITwo та чат-бот Litery для Telegram. Аналіз базуватиметься на наступних критеріях: велика база даних, підтримка різних мов, постійне поповнення бази даних, підтримка та патчі, наявність повідомлень, наявність відкладеного запиту, боротьба з думскролінгом, персоналізація новин, наявність параметра поділу статей на теми та категорії, кросплатформеність, неранжованість, повна конфіденційність даних користувача. За кожним критерієм буде відповідний результат у вигляді «+» чи «-», що означає, чи відповідає це твердження політиці чи функціональності ресурсу. Якщо ви хочете швидко ознайомитися з результатами дивись рис. 1.1 стр. 15.

Отже, давайте проведемо аналіз за кожним критерієм і пояснимо чому ми визначили його саме таким чином:

а) великий об'єм даних:

1) Google: так;

2) SMITwo: так;

3) Litery: ні.

Висновок: Google і SMITwo мають значну базу даних, що дозволяє користувачам отримувати доступ до широкого кола новинних статей. Однак Litery має меншу базу даних порівняно з двома іншими прототипами.

б) багатомовна підтримка:

1) Google: так;

2) SMITwo: так;

3) Litery: ні.

Висновок: Google і SMITwo забезпечують багатомовну підтримку, що дозволяє користувачам отримувати доступ до статей новин різними мовами. Однак Litery має обмежену багатомовну підтримку, що може обмежити його використання для тих, для кого мова не є рідною.

в) постійне оновлення бази даних:

1) Google: так;

2) SMITwo: так;

3) Litery: так.

Висновок: Усі три прототипи забезпечують безперервне оновлення бази даних, надаючи користувачам останні статті новин.

г) підтримка та виправлення:

1) Google: так;

2) SMITwo: так;

3) Litery: так.

Висновок: Google, SMITwo та Litery мають підтримку та регулярні оновлення виправлень, вирішуючи проблеми користувачів і покращуючи продуктивність системи.

г) система сповіщень:

1) Google: так;

2) SMITwo: так;

3) Litery: так.

Висновок: Усі три прототипи містять надійну систему сповіщень, яка інформує користувачів про актуальні новини.

д) відкладений запит:

1) Google: ні;

2) SMITwo: ні;

3) Litery: так.

Висновок: Litery виділяється тим, що забезпечує функцію відкладених запитів, що дозволяє користувачам планувати або відкладати пошук новин на пізніший час. Ця функція забезпечує підвищену гнучкість і зручність для користувачів.

е) прокручування:

1) Google: ні;

2) SMITwo: ні;

3) Litery: так.

Висновок: Хоча Google і SMITwo прямо не згадують перешкоду прокручування як функцію, Litery вирішує цю проблему, впроваджуючи заходи для боротьби з втомою прокручування та покращення взаємодії з користувачем.

є) персоналізовані новини:

1) Google: так;

2) SMITwo: так;

3) Litery: так.

Висновок: Усі три прототипи пропонують персоналізовані новини. Хоча Litery не спирається на переваги користувача чи історія пошуку, проте з урахуванням теми та подальшого пункту про повну конфіденційність має гідну персоналізацію з урахуванням наявних даних.

ж) категоризація статті:

1) Google: так;

2) SMITwo: так;

3) Litery: ні.

Висновок: І Google, і SMITwo забезпечують категоризацію статей, що дозволяє користувачам переглядати новинні статті на основі тем і категорій. Однак Litery не має цієї функції, що може вплинути на організацію та доступність статей.

з) кроссплатформна сумісність:

1) Google: так;

2) SMITwo: так;

3) Litery: так.

Висновок: Усі три прототипи відрізняються кроссплатформною сумісністю, що гарантує безперебійний доступ користувачів до системи на різних пристроях і операційних системах. Але варто зауважити, що Litery спирається на платформу Telegram і повністю залежить від неї в цьому плані.

и) неранжовані результати:

1) Google: ні;

2) SMITwo: ні;

3) Litery: так.

Висновок: Google і SMITwo покладаються на алгоритми ранжирування для представлення статей новин, потенційно впливаючи на різноманітність інформації, що відображається. Litery, з іншого боку, прагне надавати результати без рейтингу, сприяючи більш повному та неупередженому досвіду новин.

к) конфіденційність даних:

1) Google: ні;

2) SMITwo: ні;

3) Litery: так.

Висновок: Конфіденційність даних є проблемою для Google і SMITwo. Однак ніяким чином не збирає дані користувача, наприклад такі як:

платформа, місцезнаходження та інші, а також не обробляє їх без урахування пошуку статей за запитом та реєстрацією на ресурсі для персонального зберігання відкладених промтів.

На основі аналізу 12 критеріїв Litery стає кращим вибором для теми дослідження «Розробка ПЗ для збору та токенизації новинних статей з метою персоналізації рекомендацій та протидії думскролінгу». Litery перевершує Google і SMITwo в кількох ключових сферах, таких як можливість відкладених запитів, протидії думскролінгу та зобов'язання щодо конфіденційності даних. Тому можна точно сказати, що Litery відповідає цілям дослідження.

Також я хотів би зупинитися на ще одному виді аналізу прототипів, який ґрунтується на виборі методу розробки ресурсу та причин з яких був обраний конкретний.

У цьому розділі ми проаналізуємо два потенційних прототипи системи для розробки програмного забезпечення, спрямованого на збір і токенизацію статей новин, з метою персоналізації рекомендаційного матеріалу та боротьби з doomscrolling. Аналіз буде зосереджений на оцінці придатності та ефективності цих прототипів для вирішення конкретних вимог і цілей проекту.

Прототип 1:

Перший прототип досліджує використання інтерфейсів прикладного програмування (API), які надаються платформами новин і пошуковими системами. Використовуючи ці API, програмне забезпечення може безпосередньо отримувати доступ до статей новин і отримувати їх на основі певних критеріїв пошуку. Цей прототип пропонує перевагу доступу до даних у режимі реального часу та включає вдосконалені механізми фільтрації та ранжирування для забезпечення актуальності та точності зібраних статей. Однак це не може бути використане нами, оскільки немає можливості отримати дозвіл на використання API ключа у своїх програмних продуктах,

що може призвести до проблем із законодавством, а також отримати сам ключ досить проблематично, оскільки ресурси не роздають ключі всім підряд.

Прототип 2:

Другий прототип передбачає інтеграцію алгоритмів машинного навчання для підвищення ефективності та точності збору новинних статей і токенизації. Навчаючи систему на великому наборі даних новинних статей, програмне забезпечення може вивчати закономірності та тенденції, дозволяючи йому автоматично визначати відповідні статті на основі вподобань користувачів. Цей прототип пропонує потенціал для створення персоналізованих рекомендацій, оскільки він враховує вподобання користувачів. Однак він вимагає великої бази даних для навчання нейронної мережі, а також певну кваліфікацію у цій сфері діяльності.

Враховуючи конкретні цілі нашого проекту, включаючи генерацію персоналізованих рекомендацій на підставі лише пошукового запиту і боротьбу з doomscrolling, важливо вибрати прототип, який ефективно вирішує ці проблеми. Хоча проаналізовані прототипи пропонують певні переваги, жоден із них повністю не відповідає нашим цілям. Тому ми вирішили розробити новий прототип, який збирає статті новин з веб сторінок і обробляє її самостійно таким чином, яким нам буде завгодно для досягнення мети.

Ретельно аналізуючи та порівнюючи ці прототипи систем, ми прагнемо визначити найефективніше рішення, яке відповідає цілям проекту щодо створення персоналізованих рекомендацій і вирішення проблеми думскролінгу. Обраний прототип стане основою для наступних етапів розробки та впровадження програмного забезпечення, а так рішення, яке я зможу реалізувати самостійно.

Загалом, аналіз прототипів систем є критичним кроком на етапі попередніх досліджень. Це дозволяє нам оцінити різні підходи до автоматизації збору та токенизації новинних статей, і вибрати найбільш

підходящий не витрачаючи ресурси на переробку прототипу на етапі завершення або в подібних ситуаціях. Це економить гроші, час та сили, даючи перевагу перед конкурентами.

	Google - 8	SMITwo - 8	Litery - 9
Великий об'єм даних	+	+	-
Підтримка різних мов	+	+	-
Постійне поповнення даних	+	+	+
Підтримка та патчі	+	+	+
Повідомлення	+	+	+
Відкладений промт	-	-	+
Боротьба з думскролінгом	-	-	+
Персоналізація новин	+	+	+
Поділ на теми та категорії	+	+	-
Кросплатформеність	+	+	+
Неранжованість	-	-	+
Повна конфіденційність	-	-	+

Рисунок 1.1 – Таблиця аналізу прототипів за 12 критеріями

1.3 Обґрунтування вибору інструментального середовища для розробки програмного забезпечення

У цьому розділі ми надамо детальне обґрунтування вибору засобів розробки, які були використані в процесі розробки програмного забезпечення. Вибрані інструменти були ретельно розглянуті, щоб переконатися, що вони відповідають цілям проекту та сприяють успішному впровадженню системи.

Для реалізації всієї програми ми обрали мову програмування Python. Python надає широкий спектр бібліотек і фреймворків, які добре підходять

для різних завдань і пропонують широку підтримку обробки природної мови, веб-розробки та керування базами даних.

Для розробки інтерфейсу користувача, зокрема бота для платформи Telegram, ми використовували BotFather та бібліотеку aiogram. BotFather дозволив нам створити бота Telegram, а бібліотека aiogram надала зручні абстракції та інструменти для взаємодії з API Telegram і побудови адаптивного та зручного інтерфейсу.

Для розробки баз даних ми використовували MySQL Workbench і використовували мову SQL для ефективного керування даними. MySQL Workbench надав повний набір інструментів для проектування, моделювання та адміністрування бази даних MySQL. Мова SQL дозволила нам ефективно визначати та маніпулювати схемою бази даних, таблицями та запитами.

Для обробки серверної розробки ми використали фреймворк Flask. Flask — це легкий і універсальний фреймворк, який полегшує створення веб-додатків і API. Його простота та гнучкість зробили його ідеальним вибором для розробки серверних компонентів нашого програмного забезпечення.

Для написання та керування кодовою базою ми поклалися на інтегроване середовище розробки (IDE) PyCharm. PyCharm пропонує багатий набір функцій для розробки Python, включаючи доповнення коду, інструменти налагодження та інтеграцію системи контролю версій. Це допомогло нам оптимізувати процес розробки та забезпечило послідовність і якість коду.

Для токенизації тексту та вилучення ключових слів ми використали бібліотеки nltk і rymorphy2. Бібліотека nltk надає широкий спектр інструментів і ресурсів для завдань обробки природної мови, включаючи токенизацію, основну мову та тегування частини мови. З іншого боку, бібліотека rymorphy2 дозволила нам виконати морфологічний аналіз і виділити базові форми слів російської мови.

Для встановлення з'єднання з базою даних MySQL ми використали бібліотеку `pymysql` . Він надавав зручний інтерфейс для встановлення з'єднання, виконання SQL-запитів і отримання даних із бази даних. Бібліотека `pymysql` бездоганно інтегрована в середовище Python і сприяє ефективній взаємодії з базами даних.

Для веб-збирання ми використовували бібліотеку `BeautifulSoup` у поєднанні з бібліотекою запитів. `BeautifulSoup` дозволив нам аналізувати документи HTML і XML, отримувати відповідну інформацію та отримувати новинні статті з різних онлайн-джерел. Бібліотека запитів надала інтуїтивно зрозумілий і простий у використанні інтерфейс для створення HTTP-запитів і отримання веб-вмісту.

Вибір цих конкретних інструментів розробки ґрунтувався на їх сумісності з вимогами проекту, їх популярності та підтримці спільноти, а також за власним досвідом. Разом ці інструменти забезпечили комплексне та ефективне середовище для розробки системи програмного забезпечення.

Підсумовуючи, вибрані інструменти розробки, зокрема Python, бот Telegram (бібліотека `BotFather` і `aiogram`), MySQL Workbench, мова SQL, Flask, PyCharm, `nltk` , `pymorphy2` , `pymysql` , `BeautifulSoup` і бібліотека запитів, відіграли важливу роль у досягненні цілей проекту. Вони дозволили нам розробити функціональну та зручну програмну систему, яка спрямована на збір, токенизацію, персоналізовані рекомендації та боротьбу з думскролінгом у новинних статтях.

1.4 Обґрунтування вибору технічної платформи, що розробляється

У цьому розділі ми детально обґрунтуємо вибір технічної платформи, необхідної для розгортання розробленої системи. Обрана платформа відіграє вирішальну роль у забезпеченні успішної роботи та інтеграції програмного забезпечення.

Серверна сторона системи була розроблена таким чином, щоб бути сумісною з платформою Windows. Windows пропонує надійне та широко використовуване середовище операційної системи, яке забезпечує надійну продуктивність, масштабованість і функції безпеки. Вибравши Windows як серверну платформу, ми можемо скористатися перевагами її широкої підтримки технологій веб-хостингу, керування базами даних і мережевих служб.

З іншого боку, клієнтська сторона системи, яка інтегрована з платформою Telegram, розроблена як кросплатформна. Telegram доступний у різних операційних системах, включаючи Windows, macOS, Linux, iOS та Android. Ця крос-платформна сумісність дозволяє користувачам отримувати доступ до системи з різних пристроїв і операційних систем без будь-яких обмежень. Це забезпечує безперебійну роботу користувачів на різних платформах, забезпечуючи гнучкість і зручність для користувачів.

Вибір Telegram як платформи для інтеграції на стороні клієнта зумовлений кількома факторами. Telegram пропонує добре задокументований і багатофункціональний API, який дозволяє розробникам ефективно взаємодіяти з його платформою обміну повідомленнями. Він забезпечує розширену підтримку для створення ботів, керування взаємодією користувачів і безпечної доставки повідомлень. Використовуючи можливості Telegram, ми можемо створити зручний інтерфейс і забезпечити спілкування між системою та користувачами в режимі реального часу.

Важливо відзначити, що функціональність і продуктивність системи тісно пов'язані з платформою Telegram. Система покладається на інфраструктуру Telegram для доставки повідомлень, автентифікації користувачів і зберігання даних. Таким чином, доступність і стабільність платформи Telegram безпосередньо впливають на загальну продуктивність і надійність системи. Постійний моніторинг стану та оновлень платформи

Telegram має вирішальне значення для забезпечення безперебійної роботи системи.

Таким чином, вибір технічної платформи для розробленої системи базується на сумісності з серверним середовищем Windows і кросплатформенному характері клієнтської інтеграції з платформою Telegram. Використовуючи сильні сторони цих платформ, ми можемо розгорнути надійну та універсальну програмну систему, яка забезпечує безперебійний зв'язок, генерацію персоналізованих рекомендацій та ефективну боротьбу з доомскролінгом на багатьох пристроях і операційних системах.

1.5 Задачі дипломного проекту

У цьому розділі ми окреслимо цілі дипломного проекту, які узгоджуються з попередньо сформульованою метою дослідження. Метою проекту є створення та впровадження Telegram-бота, який автоматизує процес збору новин, визначення ключових слів та генерацію персоналізованих рекомендацій. Цей бот має на меті покращити взаємодію з користувачем, заощадити час і боротися з doomscrolling.

1. Дослідити основні причини та фактори, що сприяють виникненню проблеми:

- 1.1 Визначити основні причини поширеності думскролінгу серед користувачів.
- 1.2 Розглянути соціальні, психологічні та культурні аспекти, що сприяють негативним звичкам споживання новин.
- 1.3 Проаналізувати вплив надмірного споживання новин на добробут і продуктивність користувачів.

- 1.4 Дослідити наслідки і залежності між перебірливим переглядом новин і психоемоційним станом людей, а також їхньою продуктивністю та загальним самопочуттям.
- 1.5 Дослідити психологічні та поведінкові фактори, які сприяють безперервній поведінці прокручування.
- 1.6 Розглянути механізми пристрасті до перегляду новин та виявити основні стимули та умови, що спонукають людей до безконтрольного скролінгу.
- 1.7 Дослідити роль персоналізованих рекомендацій у вирішенні проблеми та покращенні взаємодії з користувачем.
- 1.8 Проаналізувати, як використання персоналізованих рекомендацій може вплинути на зміну споживацького поведінки та зменшення негативного впливу думскролінгу.
2. Провести аналіз існуючих підходів і рішень, які використовуються для боротьби з проблемою:
 - 2.1 Переглянути поточні платформи агрегації новин та їхні системи рекомендацій.
 - 2.2 Дослідити методики та алгоритми, використані для формування рекомендаційного контенту, що допомагає зменшити думскролінг.
 - 2.3 Оцінити ефективність алгоритмів фільтрації вмісту для зменшення поведінки думскролінгу.
 - 2.4 Проаналізувати наявні підходи до фільтрації вмісту та їхній вплив на виключення непотрібних, негативних або надмірних новинних матеріалів.
 - 2.5 Переглянути використання налаштувань користувача для створення персоналізованих рекомендацій.

- 2.6 Дослідити, як налаштування особистих вподобань та інтересів користувачів можуть бути використані для створення більш релевантних та корисних рекомендацій.
- 2.7 Дослідити інтеграцію методів обробки природної мови для виділення ключових слів.
- 2.8 Розглянути різноманітні методи та підходи до обробки природної мови, які можуть допомогти виявити та виділити ключові слова та фрази в новинах для покращення рекомендаційного процесу.
3. Розробити модуль веб-збирання: перша мета полягає в розробці та реалізації модуля веб-збирання, який може збирати новинні статті з різних онлайн-джерел. Модуль повинен мати можливість отримувати релевантну інформацію зі статей, таку як назва, зміст і дата публікації. Для забезпечення ефективного й точного пошуку даних він має використовувати методи веб-збирання.
4. Впровадити алгоритми обробки природної мови (NLP): Друга мета полягає в інтеграції алгоритмів NLP в систему. Ці алгоритми використовуватимуться для токенизації зібраних статей, виділення ключових слів і аналізу настрою вмісту. Використовуючи методи НЛП, система може генерувати значущу інформацію та надавати персоналізовані рекомендації на основі вподобань користувачів.
5. Розробити бота Telegram. Третя мета — створити зручний та інтерактивний бот Telegram. Бот повинен дозволяти користувачам взаємодіяти з системою, вводити свої вподобання та отримувати персоналізовані рекомендації щодо новин. Він має забезпечувати зручну та інтуїтивно зрозумілу взаємодію з користувачем, дозволяючи користувачам легко переміщатися між функціями системи та отримувати відповідні оновлення новин.

6. Впровадити рекомендаційні алгоритми. Четверте завдання полягає в тому, щоб включити рекомендаційні алгоритми в систему. Ці алгоритми використовуватимуть витягнуті ключові слова та налаштування користувача.
7. Оцінити ефективність рішення шляхом порівняльного аналізу:
 - 7.1 Провести тестування користувачів і зібрати відгуки.
 - 7.2 Виміряти час, витрачений на перегляд статей новин без бота, і порівняти його з часом, витраченим з використанням бота.

Досягнувши поставлених цілей, дипломний проект має на меті створення комплексного та ефективного рішення, спрямованого на автоматизацію збору новин, визначення ключових слів та надання персоналізованих рекомендацій. Цей проект відповідає головній меті покращення взаємодії з користувачем, економії часу та боротьби з явищем doomscrolling.

Одним з головних напрямків дипломної роботи є створення системи автоматичного збору новин. Ця система буде сканувати різні джерела новин, аналізувати їх зміст та виокремлювати ключові слова. Завдяки цьому користувачам будуть запропоновані тільки актуальні та цікаві новини, що сприятиме покращенню їхнього досвіду використання.

Не менш важливим аспектом дипломного проекту є боротьба з явищем doomscrolling. Для цього будуть використовуватися спеціальні методи та підходи, що спрямовані на позитивну зміну звичок користувачів у споживанні новин. Розроблена система надаватиме користувачам можливість відстежувати час, проведений за новинами, а також контролювати кількість негативного контенту, що сприятиме створенню більш здорової звички споживання новин.

Отже, дипломний проект охоплює широкий спектр розробок, спрямованих на покращення взаємодії з користувачем, ефективне використання часу та зменшення впливу doomscrolling. Результатом цих робіт

буде створення цінного інструменту, який допоможе користувачам отримувати лише якісну та актуальну інформацію, покращуючи їхній досвід використання новинних ресурсів.

2 ДЕТАЛІЗАЦІЯ ЗАДАЧІ АВТОМАТИЗАЦІЇ

Цей розділ має на меті забезпечити всебічне розуміння необхідності та характеру деталізації завдань на кожному рівні разом із функціональним призначенням визначених підзадач та їхніми відповідними операндами та результатами.

Автоматизація відіграє ключову роль у оптимізації процесів, підвищенні ефективності та досягненні бажаних результатів. Однак для ефективної автоматизації завдання вкрай важливо розбити його на складові підзадачі, що дозволить проводити більш детальний аналіз і впровадження. Поступово деталізуючи завдання на різних рівнях, ми зможемо визначити та вирішити конкретні проблеми та вимоги, пов'язані з кожним підзавданням.

У цьому розділі ми почнемо з формулювання завдання автоматизації в найбільш загальному вигляді. Потім ми приступимо до поетапного процесу деталізації завдання, рівень за рівнем, розбиваючи його на підзавдання. Буде детально пояснено обґрунтування та необхідність кожного рівня деталізації, підкреслюючи переваги розбиття завдання на його компоненти.

Кожна ідентифікована підзадача буде описана з точки зору її функціонального призначення в дисертпроцесу автоматизації. Крім того, ми окреслимо операнди та результати, характерні для кожного підзавдання, забезпечуючи чітке розуміння вхідних даних і очікуваного результату або результату, пов'язаного з кожним компонентом.

Щоб забезпечити цілісність і узгодженість процесу автоматизації, ми також перевіримо правильність взаємозв'язків між підзадачами на кожному рівні деталізації. Цей етап перевірки має вирішальне значення для того, щоб підзавдання працювали злагоджено та спільно сприяли досягненню загальної мети автоматизації.

Відомості, отримані в результаті цього аналізу, не тільки допоможуть ефективно реалізувати завдання автоматизації, але й стануть орієнтиром для

майбутніх досліджень і розробок у сфері автоматизації завдань. Ретельно вивчивши та деталізувавши підзавдання, ми можемо забезпечити надійний і комплексний підхід до автоматизації складних процесів.

Почнемо з формулювання завдання автоматизації в найбільш загальному вигляді. Потім ми приступимо до поетапного процесу деталізації завдання, рівень за рівнем, розбиваючи його на підзавдання.

Отже до цілей входять:

- 1) авторизація користувача у Telegram боті або на веб-сайті;
- 2) створення інтерфейса ресурсу;
- 3) обробка та показ даних.

Для вирішення задачі визначаємо:

- 1) запит на авторизацію;
- 2) отримані дані;
- 3) оброблені дані;
- 4) результат.

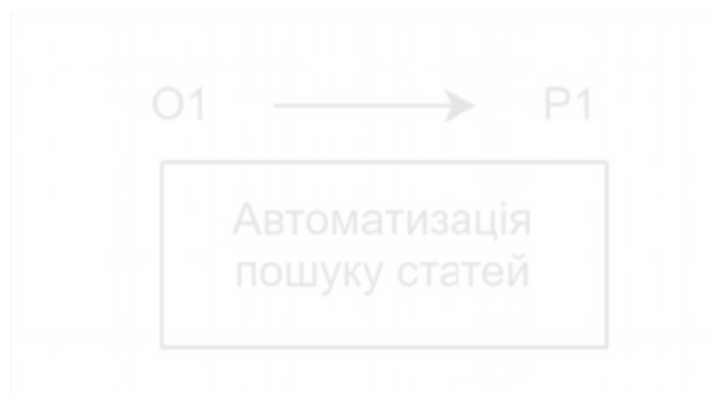


Рисунок 2.1 – Графічне зображення задачі

За рис. 2.1 маємо, що O1 – запит авторизації, а P1 – дані, які програма отримала від користувача.

2.1 Аналіз 1-го рівня деталізації

Далі здійснюється послідовна, порівнева деталізація завдань, починаючи із загальної задачі.

Виконуючи декомпозицію виділяємо три головні підзадачі:

- 1) авторизація;
- 2) взаємодія з розділом користувача;
- 3) виведення результату.

У першій задачі ми отримуємо дані для подальшої обробки.

Авторизацією – є данні користувача, які отримала програма при запуску програми. У ході виконання даної задачі виділяються данні, які необхідно обробити. Ці данні і вхідними.

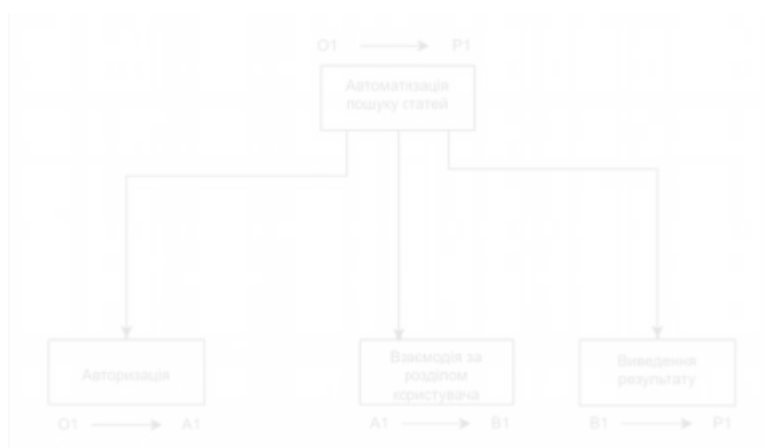


Рисунок 2.1 – Графічне зображення деталізації першого рівня

Розглянемо рис. 2.1, маємо, що A1 – це дані від користувача у ході авторизації. В ході виконання задачі отримуємо оброблені дані.

Тепер перевіримо коректність деталізації. Для цього визначимо операндно-результатну множину на рис. 2.2:



Рисунок 2.2 – Розрахунки операндно-результатної множини

Якщо скоротити з обох сторін спільні операнди (ті, що підкреслені), отримаємо рівняння, яке еквівалентне операндно-результатній множині попереднього рівня деталізації.

2.2 Аналіз 2-го рівня деталізації

На другому рівні опишемо авторизацію користувача на веб-сайті та Telegram боті. Це необхідно для проведення аутентифікації у майбутніх розділах та коректно записувати та відправляти дані користувачеві.

Тож давайте визначимо підзадачі цього рівня деталізації:

- 1) отримання авторизаційних даних від користувача;
- 2) обробка та запис даних до бази даних;
- 3) надання доступу користувачеві.

Так як авторизаційні дані будуть зберігатися програмою до бази даних створимо нову змінну L1, яка буде означати, що дані були успішно відправлені до сервера і готові до слідуєчого етапу.

Для того щоб обробити інформацію на сервері та записати їх, також уведемо нову змінну з ім'ям J1, яка у свою чергу буде означати, що дані від L1 прийшли на обробку та будуть передані на наступний крок під цією назвою.

Наступним же кроком стане надання доступу до сервісу користувачеві, а значить передаємо J1 до A1 як результат роботи цього блоку.

Отримаємо такий результат на рис. 2.3.

Перевіримо коректність декомпозиції другого рівня за допомогою розрахунку операндно-результатної множини на рис. 2.4:



Рисунок 2.4 – Розрахунок операндно-результатної множини

Бачимо, що якщо скоротити з обох сторін спільні операнди, то отримаємо еквівалентне рівняння операндно-результатній множині попереднього рівня. Отже все зроблено правильно.



Рисунок 2.3– Графічне зображення деталізації другого рівня

2.3 Аналіз 3-го рівня деталізації

На третьому рівні опишемо як користувач взаємодіє за веб-сайтом чи Telegram ботом. Після авторизації, користувач отримує доступ до основного функціоналу, який дає змогу знайти бажану новину статтю та переглянути її.

Тож давайте визначимо підзадачі цього рівня деталізації:

- 1) створення пошукового запиту;
- 2) надання фільтрів для пошукового запиту;
- 3) перегляд наданого результату.

Дані користувача після авторизації будуть зберігатися в програмі і кожен раз коли він буде повертатися до ресурсу, дані будуть звірятися з даними цього облікового запису. Це дані які були збережені у браузері, тому це теж саме, що і авторизація, але дані не потрібно вводити власноруч. Тому, якщо дані зберігаються у програмі, було виділено окрему підзадачу, яка відправляє дані до програми. Вона має ім'я A1, а створення пошукового запиту буде мати назву C1.

Для того щоб обробити інформацію на сервері, ми уведемо нову зміну з ім'ям D1, яка буде означати, що дані від C1 відправляться на обробку та будуть передані на наступний крок під цією назвою. Наступним же кроком стане перегляд результату користувачем у ресурсі, а значить передаємо D1 до B1 як результат роботи цього блоку.

Отримаємо такий результат рис. 2.5.

Перевіримо коректність декомпозиції другого рівня за допомогою розрахунку операндно-результатної множини, рис. 2.6:

Бачимо, що якщо скоротити з обох сторін спільні операнди, то отримаємо еквівалентне рівняння операндно-результатній множині попереднього рівня. Отже все зроблено правильно.



Рисунок 2.5 – Графічне зображення деталізації другого рівня



Рисунок 2.6 – Розрахунок операндно-результатної множини

2.4 Аналіз 4-го рівня деталізації

На четвертому рівні опишемо як користувач зможе переглядати результат роботи програми. Після створення пошукового запиту, користувача перекине на іншу сторінку, де він зможе це зробити.

Тож давайте визначимо підзадачі цього рівня деталізації:

- 1) отримання результуючих даних від сервера;
- 2) підготовка інтерфейса для відображення результату;
- 3) перегляд наданого результату.

Для того щоб отримати дані від сервера ми повині завести нову змінну. Давайте назовемо її H1. Вона буде означати, що дані успішно прийшли до користувача та готові до слідуєчого етапу.

Далі необхідно підготувати інтерфейс для відображення результатів. Для цього серверу також потрібно відправити дані, тільки цього разу це будуть HTML, CSS та JS файли. Давайте також заведемо нову змінну для цього – M1.

Отримуючи дані від сервера до комп'ютера користувача ми передаємо B1 до H1, а для підготовки інтерфейсу H1 передаємо до M1. Таким чином для показу самого результату M1 переходить до P1.

Отримаємо такий результат рис. 2.7.

Перевіримо коректність декомпозиції другого рівня за допомогою розрахунку операндно-результатної множини, рис. 2.8:

Бачимо, що якщо скоротити з обох сторін спільні операнди, то отримаємо еквівалентне рівняння операндно-результатній множині попереднього рівня. Отже все зроблено правильно.



Рисунок 2.7 – Графічне зображення деталізації другого рівня



Рисунок 2.8 – Розрахунок операндно-результатної множини

роботи всіх компонентів, включаючи Телеграм бота.

Варто зазначити, що кожна система виконує свої специфічні завдання. Система Logic відповідає за логіку функціонування системи, а також забезпечує безперебійну роботу сайту та обробку нових даних. Телеграм бот віднесений до системи, розташованої над Logic, щоб навіть у випадку помилки в самому боті, робота сайту або збір та обробка даних не були порушені.

Розглянемо цей підхід детальніше в розділі «4.1 Архітектура програмного забезпечення», де детально описано взаємозв'язок та взаємодію різних компонентів системи. Даний розділ надасть більше відомостей про структуру системи, її складові та принципи взаємодії, що сприятиме кращому розумінню функціональності та переваг цієї архітектури.

Таким чином, функціональна система забезпечує ефективну та безперебійну роботу компонентів, мінімізуючи вплив помилок та збоїв на роботу всієї системи. Ця архітектура робить проект стійким та надійним, що в свою чергу сприяє безперервному збору та обробці даних, а також забезпечує плавну та неперервну взаємодію з користувачами.

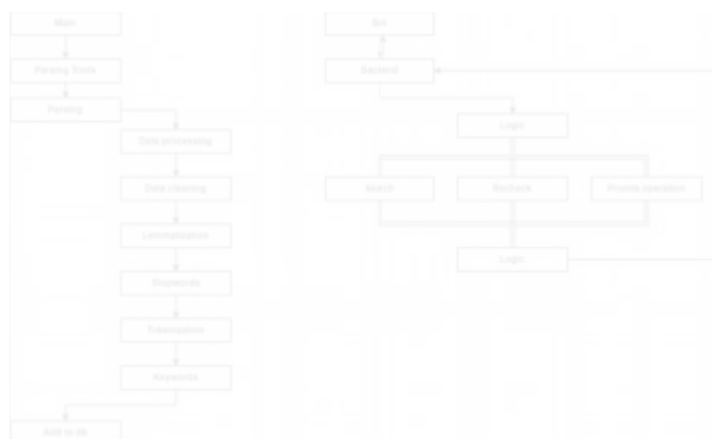


Рисунок 2.10 – Взаємодія модулів python файлів

3 АЛГОРИТМИ РОЗВ'ЯЗАННЯ ЗАДАЧ

Розділ «Алгоритми розв'язування задач» цієї дипломної роботи присвячений алгоритмам, розробленим і реалізованим для вирішення завдань теми дослідження. Цей розділ має на меті надати повний огляд використовуваних алгоритмів, їхніх конкретних методів і операцій, а також їхнього значення в автоматизації різних підзадач для досягнення бажаної функціональності.

Ефективне вирішення проблем лежить в основі будь-якого проекту розробки програмного забезпечення, а вибір і впровадження відповідних алгоритмів відіграють вирішальну роль у досягненні ефективних і точних результатів. У цьому розділі ми розглянемо алгоритми, які використовуються для розробки програмного забезпечення для збирання новинних статей, токенизації, лематизації, фільтрації, очищення, видалення стоп-слів, складання набору ключових слів та аналіз запиту для персоналізованих рекомендацій і боротьби з думскролінгом.

Впровадження цих алгоритмів відіграє життєво важливу роль у розробці ефективної автоматизованої системи пошуку новин. Вони дозволяють системі обробляти, аналізувати та доставляти релевантні новинні статті, тим самим покращуючи взаємодію з користувачем та оптимізуючи процес пошуку. Ось більш детальний опис кожного алгоритму:

1. Розбір новинної статті:

Алгоритм використовує методи синтаксичного аналізу HTML для отримання відповідної інформації з необробленого HTML коду. Цей алгоритм визначає певні теги HTML, які пов'язані з заголовком, вмістом, назвою ресурсу та датою публікації статті. Операндами цього алгоритму є необроблений HTML код, а результатом є видобута інформація, яка включає заголовок, вміст, назву ресурсу та дату публікації. Ця інформація зберігається як поля даних для подальшого використання.

2. Очищення статті (видалення спеціальних символів):

Алгоритм для очищення статті здійснює обробку вмісту статті і застосовує різноманітні операції для видалення спеціальних символів, таких як крапки, коми, знаки питання та інші небуквено-цифрові символи. Операндом цього алгоритму є зміст статті, а результатом є очищений вміст статті без будь-яких спеціальних символів. Це забезпечує, що залишаються лише буквено-цифрові символи, що сприяє подальшому аналізу та обробці тексту.

3. Лематизація статті:

Алгоритм лематизації статті використовує методи обробки природної мови для скорочення кожного слова у вмісті статті до його базової або словникової форми. Цей алгоритм застосовує лінгвістичні правила та алгоритми для стандартизації слів і зменшення надмірності в подальшому аналізі. Операндом цього алгоритму є зміст статті, а результатом є лематизований вміст статті, де слова трансформуються в їхню основну форму. Це поліпшує послідовність слів та спрощує подальше вилучення ключових слів для подальшого аналізу.

4. Видалення стоп-слів:

Алгоритм видалення стоп-слів з лематизованого вмісту статті передбачає вилучення звичайних слів, які не мають істотного значення для загального змісту тексту, наприклад, артиклів, займенників та прийменників. Цей алгоритм зосереджується на словах змісту, які є більш інформативними та значущими для подальшого аналізу. Його операндами є лематизований вміст статті та попередньо визначений список стоп-слів, а результатом є вміст статті з видаленими стоп-словами. Це підвищує релевантність і значимість решти слів у тексті, сприяючи більш точному аналізу та зрозумінню його змісту.

5. Загалом, описані алгоритми Токенізація статті:

Алгоритм розбиває попередньо оброблений вміст статті на окремі токени (слова) шляхом використання методу поділу за пробілами. Операндом цього алгоритму є зміст статті, а результатом є токенизований вміст статті, який представлений у вигляді послідовності окремих слів або tokenів. Це забезпечує зручність подальшого аналізу та обробки тексту на основі окремих слів.

6. Розрахунок часто зустрічаються tokenів і генерація ключових слів:

Алгоритм розраховує частоту кожного токена в токенизованому вмісті статті. Він визначає лексеми, які найчастіше зустрічаються, і обирає їх як ключові слова, що охоплюють основні теми або концепції, присутні в статті. Це досягається шляхом підрахунку кількості для кожного токена і вибору перших N tokenів з урахуванням їхньої частоти. Операндом цього алгоритму є токенизований вміст статті, а результатом є згенеровані ключові слова, які відображають найбільш часто зустрічаються токени. Це надає компактне представлення основних тем, що розглядаються в статті, сприяючи зрозумінню та аналізу її змісту.

7. Зіставлення запиту користувача зі статтею:

Алгоритм порівнює токени, які отримані з пошукового запиту користувача, зі згенерованими ключовими словами статті. Він проводить перевірку збігів між маркерами запиту користувача та ключовими словами статті. Операндами цього алгоритму є токенизований запит користувача та ключові слова статті, а результатом є відповідність, яка вказує на релевантність або подібність між запитом користувача та статтею. Це забезпечує міру того, наскільки добре стаття відповідає наміру пошуку користувача.

8. Відповідна фільтрація:

Алгоритм застосовує критерії фільтрації до відповідних статей. Він перевіряє, чи була стаття раніше надіслана користувачеві, з метою уникнення дублювання вмісту. Крім того, він перевіряє, чи відповідає джерело статті

дозволенним джерелам новин, які визначені в системі. Ця фільтрація гарантує, що користувачу будуть представлені лише унікальні та релевантні статті, які відповідають визначеним критеріям. Операндами цього алгоритму є відповідні статті та критерії фільтрації, а результатом є відфільтрований набір статей, які задовольняють критеріям фільтрації. Це забезпечує оптимальну взаємодію з користувачем завдяки представленню різноманітного та надійного вмісту.

Зв'язавши ці алгоритми між собою, ми отримаємо python модуль з необхідним функціоналом. Для більшої наглядності, на рис. 3.1 можна побачити блок-схему, яка візуально відображає роботу алгоритмів розв'язання задач приведених вище:



Рисунок 3.1 – Блок схема алгоритмів розв’язання задач

4 КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Архітектура програмного забезпечення

Програмна архітектура проекту створена для забезпечення стабільної та надійної системи автоматизації новин. Архітектура складається з кількох програмних компонентів, які взаємодіють один з одним для досягнення бажаної функціональності. Програмну систему складають такі компоненти:

1. Модуль Telegram Bot (aiogrambot.py):

Сервер Telegram Bot відповідає за обробку взаємодії користувачів через інтерфейс бота Telegram. Він використовує бібліотеку aiogram Python, яка надає API високого рівня для взаємодії з API Telegram Bot. Сервер створено за допомогою Flask, легкої веб-платформи для Python, яка забезпечує легку інтеграцію з API Telegram. Він отримує вхідні повідомлення від користувачів, обробляє команди та запити та відповідає відповідним чином. Взаємодіє з базою даних для отримання та зберігання налаштувань користувача, пошукових запитів і відповідної інформації про статті.

2. Сервер веб-сайту (flask_db.py):

Сервер веб-сайту служить інтерфейсом для взаємодії користувачів із системою через веб-інтерфейс. Він реалізований за допомогою Flask, HTML, CSS і JavaScript, забезпечуючи зручний і адаптивний веб-інтерфейс. Сервер забезпечує реєстрацію користувачів та автентифікацію. Він зв'язується з базою даних для отримання та відображення статей новин, обробки запитів користувачів і збереження налаштувань користувачів. Логіка на стороні сервера забезпечує правильну обробку введених користувачем даних і підтримує узгодженість відображеної інформації.

3. Розбір і обробка даних (parsing_tools.py):

Модуль аналізу та обробки даних відповідає за збирання статей новин із зовнішніх джерел і обробку витягнутих даних. Він використовує такі бібліотеки, як BeautifulSoup, і запити на отримання HTML-вмісту з веб-сайтів

і вилучення відповідної інформації. Модуль виконує такі завдання, як очищення статей від непотрібних символів, форматування вмісту та вилучення метаданих. Він також містить функцію збереження проаналізованих статей у файли для зберігання та подальшої обробки. Крім того, модуль реєструє помилки та дії, пов'язані з аналізом, щоб полегшити налагодження та моніторинг.

4. Взаємодія з базою даних (add_article_to_db.py):

Модуль Database Interaction обробляє всі взаємодії з базою даних MySQL. Він надає функції для додавання статей до бази даних, керування інформацією про користувачів і виконання SQL-запитів. Модуль використовує такі бібліотеки, як Flask-MySQLdb і pymysql для встановлення з'єднання з базою даних MySQL. Він забезпечує належне зберігання та пошук статей новин, налаштувань користувача та інформації про пошуковий запит. Модуль також обробляє операції з базою даних, пов'язані з реєстрацією користувачів, автентифікацією та керуванням підписками користувачів.

5. Пошук і фільтрація (check_promts.py):

Модуль «Пошук і фільтрація» реалізує логіку моніторингу пошукових запитів і зіставлення їх із збереженими ключовими словами. Він постійно відстежує вхідні пошукові запити та порівнює їх із ключовими словами із відкладених пошукових запитів. Модуль гарантує, що статті з відповідними ключовими словами будуть отримані та оброблені для подальших дій. Він також містить механізми, які запобігають надсиланню дублікатів статей користувачам і фільтрують результати пошуку на основі вподобань користувачів. Функціональність модуля полегшує персоналізовані рекомендації щодо новин і покращує загальну взаємодію з користувачем.

6. Токенізація та обробка даних (токенізація):

Модуль токенизації та обробки даних відповідає за обробку даних перед їх збереженням у базі даних. Він використовує такі бібліотеки, як morphology2,

nltk, chardet і ruscountr, для виконання різних завдань обробки даних. Модуль містить функції для лемматизації слів, видалення стоп-слів і токенизації вмісту статті. Це забезпечує перетворення даних у відповідний формат для зберігання та вилучення з бази даних.

Крім того, модуль підраховує частоту токенів і генерує ключові слова, які можна використовувати для пошуку та фільтрації. Компоненти програмного забезпечення в архітектурі мають чітко визначені ролі та обов'язки, що забезпечує модульність і поділ завдань. Кожен модуль працює незалежно і отримує лише необхідну інформацію з бази даних для своєї функціональності.

Зв'язок між компонентами програмного забезпечення полегшується через базу даних MySQL, яка діє як центральне сховище для обміну даними. Кожен модуль отримує та зберігає необхідні дані з бази даних, забезпечуючи послідовність і узгодженість у всій системі.

Архітектура програмного забезпечення відповідає моделі клієнт-сервер, де бот Telegram і веб-сайт служать інтерфейсами для взаємодії з користувачем. Ці інтерфейси надають схожі функціональні можливості, дозволяючи користувачам отримувати доступ до системи та взаємодіяти з нею на основі їх бажаного інтерфейсу.

Що стосується шаблонів і принципів архітектури, проект використовує модульний підхід, де кожен відповідальний за свої конкретні завдання. Цей вибір дизайну сприяє модульності, масштабованості та відмовостійкості. Архітектура також дотримується принципів поділу проблем та інкапсуляції, забезпечуючи чіткі обов'язки та мінімізуючи залежності між компонентами.

Важливо відзначити, що архітектуру програмного забезпечення було розроблено та протестовано на платформі Windows, що забезпечує сумісність і стабільність у цьому середовищі.

Використовуючи чітко визначену архітектуру програмного забезпечення та використовуючи відповідні технології, система досягає своїх

цілей щодо автоматизації обробки новин і забезпечення надійного інтерфейсу користувача.

Нагадую, що ви можете ще раз ознайомитися з структурою модулів на рис. 2.10.

4.2 Інформаційний простір системи

Інформаційний простір системи охоплює інформаційні потоки між основними компонентами програмного забезпечення та структурою основної бази даних. У цьому розділі описано потік інформації, структуру бази даних, а також представлено інфологічну, логічну та фізичну моделі бази даних, а також діаграму варіантів використання UML.

1. Бот і сервер Telegram:

Інформаційний потік між Telegram Bot і сервером передбачає обмін такими даними: під час реєстрації користувач вказує своє ім'я користувача, ідентифікатор чату, пароль та облікові дані для входу. Користувач взаємодіє з ботом, вводячи такі команди, як /start, /help, /description, /create і /get. Ці команди служать різним цілям, включаючи ініціювання розмов, пошук допомоги, отримання інформації та створення відкладених пошукових запитів. Інформація, яка передається між Telegram Bot і сервером, є мінімальною і в основному включає реєстраційні дані користувачів і відкладені пошукові запити.

2. Веб-сайт і сервер:

Інформаційний потік між веб-сайтом і сервером містить такі дані: під час реєстрації користувач вказує свою електронну пошту, пароль, псевдонім і за бажанням ім'я користувача Telegram. Згодом користувач може надсилати пошукові запити, а сервер відповідає документами HTML, CSS, файлами JavaScript, зображеннями та новинними статтями. Дані, якими обмінюються веб-сайт і сервер, обмежуються конкретним запитуваним вмістом і

відповідною інформацією користувача. Щоразу, коли користувач отримує доступ до веб-сайту, сервер перевіряє облікові дані для входу (ім'я користувача/пароль або ім'я користувача Telegram), щоб підтвердити особу користувача.

3. Формати даних і безпека:

Дані, якими обмінюються компоненти, переважно мають формат JSON, який широко використовується для зберігання посилань і назв статей під час процесу синтаксичного аналізу. Цей формат дозволяє ефективно організовувати дані, забезпечує зручний доступ до них і дозволяє здійснювати різноманітні маніпуляції. Крім того, в системі використовуються й інші типи даних, такі як списки, словники та рядки, які допомагають структурувати та організовувати інформацію.

Окрім текстових даних, в системі зустрічаються й зображення у форматі JPG. Ці зображення зберігаються на диску і використовуються для візуалізації статей та забезпечення зручного сприйняття контенту користувачами. Крім того, база даних системи зберігає шлях до кожного файлу у вигляді посилання на рядок, що дозволяє ефективно керувати та організовувати зображення.

Важливо відзначити, що наразі система не використовує методи стиснення чи шифрування даних. Однак, це відкриває можливості для подальшого розвитку та покращення системи в аспекті безпеки. Використання методів стиснення даних може допомогти зменшити обсяг пам'яті, необхідний для зберігання інформації, що позитивно позначиться на продуктивності системи. Крім того, шифрування даних є необхідним для забезпечення конфіденційності та захисту від несанкціонованого доступу до чутливої інформації.

Таким чином, для подальшого розвитку системи варто розглянути використання методів стиснення даних та впровадження шифрування, що сприятиме забезпеченню більшої ефективності, безпеки та конфіденційності.

Це дозволить системі стати ще більш потужним та надійним інструментом для обміну та зберігання інформації.2. Структура бази даних

База даних, яка використовується в системі, — це MySQL, і вона має таку структуру: база даних містить таблиці, що представляють різні сутності та їхні атрибути. Основні таблиці містять інформацію про користувачів (реєстраційні дані), пошукові запити, аналізовані статті та вміст веб-сайту. Зв'язки між таблицями існують для підтримки цілісності даних. Нижче наведено опис фізичної моделі бази даних рис. 4.1. На фізичній моделі зупинимося детальніше та на ній зрозуміємо що і як зроблено.

Фізична модель представляє реальну реалізацію бази даних на обраному апаратно-програмному середовищі. Він зосереджений на оптимізації продуктивності, ефективності зберігання та доступу. Фізична модель враховує такі аспекти, як структури зберігання даних, стратегії індексування, розділення та інші деталі, що стосуються реалізації.

Основою нашої бази даних є центральна сутність під назвою `user_account`, що служить невід'ємною складовою всіх інших сутностей, таких як `news_resource`, `history`, `like`, `sessions`, `promts`, `os`, `subscriptions`, `history_search` і `sent_articles`. Хоча деякі з цих сутностей наразі не реалізовані у програмі, вони вже попередньо включені для майбутнього масштабування бази даних, що свідчить про нашу наміру розвивати проект і після завершення дипломного розділу.

Звичайно ж, `user_account` зберігає безліч стовпців, які містять значення таких даних, як електронна пошта, пароль, телеграм-ім'я користувача, дата реєстрації, ім'я користувача, аватар аккаунту, ідентифікатор чату і багато інших стовпців, які в подальшому будуть необхідні для дальшого розширення функціоналу. Але варто зазначити, що деякі з цих стовпців не є обов'язковими для заповнення, такі як телеграм-ім'я користувача, ім'я користувача, ідентифікатор чату і аватар аккаунту. Це рішення прийняте з метою уникнення надмірного завантаження користувачів зайвими даними, які не

впливають на основний функціонал і не є обов'язковими для роботи програми.

Ця головна сутність, `user_account`, встановлює зв'язки з іншими сутностями за допомогою стовпця `id_uacc`, який є унікальним ідентифікатором користувача. Завдяки цьому стовпцю ми зможемо отримувати інформацію, що стосується конкретного користувача, та забезпечувати зв'язок між різними даними.

Крім `user_account`, існує ще одна важлива сутність під назвою `article`, яка не має безпосереднього зв'язку з `user_account`. Ця таблиця містить в собі інформацію про статті, включаючи оброблені дані, наприклад ключові слова.

Варто відзначити, що взаємозв'язки між сутностями відрізняються в різних таблицях. У всіх випадках маємо або зв'язок "багато до одного", або зв'язок "один до багатьох". Такий підхід до зв'язків обумовлений тим, що кожному користувачеві може відповідати безліч запитів, дій або надісланих статей. Такий зв'язок є необхідним для належного зберігання даних та подальшої обробки їх.

Через те, що база даних робилася під завдання швидкого розширення функціоналу сервісу, база даних буде трохи більшою. Однак усе те, що було описано вище, повністю застосовується до неї і робилося на основі цього.



Рисунок 4.1 – Модель базы данных у программе MySQL Workbench

4.3 Інтерфейс користувача

Інтерфейс користувача (UI) відіграє вирішальну роль у забезпеченні бездоганної та зручної взаємодії з програмною системою. У цьому розділі ми обговоримо обрану структуру користувальницького інтерфейсу, надамо детальний опис його елементів і пояснимо технологію, використану для його реалізації.

Для нашої системи ми обрали модульну структуру, яка вміщує як Telegram Bot, так і веб-інтерфейс. Ця структура забезпечує узгоджену роботу користувачів на різних платформах. До основних компонентів інтерфейсу належать:

1. Інтерфейс Telegram Bot: інтерфейс Telegram Bot дозволяє користувачам взаємодіяти з системою за допомогою текстових команд. Він надає такі команди, як /start, /help, /description, /create і /get, які дозволяють користувачам виконувати різні дії. Структура інтерфейсу Telegram Bot відповідає лінійному потоку команд, де користувачі ініціюють певні команди для запуску функціональності системи.
2. Веб-інтерфейс: веб-інтерфейс забезпечує графічний інтерфейс користувача для взаємодії користувачів із системою через веб-браузер. Він містить форми реєстрації та входу, поля введення пошуку, кнопки для створення та отримання відкладених пошукових запитів і відображення результатів пошуку. Веб-інтерфейс має багатосторінкову структуру з окремими аркушами для реєстрації, входу, пошуку та відображення результатів.

4.3.1 Опис елементів інтерфейсу

Опис елементів інтерфейсу бота:

1. Команда запуску: ця команда ініціює процес реєстрації для нових користувачів рис. 4.2.
2. Команда довідки: ця команда надає користувачам інформацію про доступні команди та їхні функції рис. 4.3.
3. Команда опису: ця команда відображає короткий опис системи та її можливостей рис. 4.4.
4. Створити команду: ця команда дозволяє користувачам створювати відкладені пошукові запити, вказуючи критерії пошуку рис. 4.5.
5. Отримати команду: ця команда отримує та відображає результати пошуку для попередньо створених запитів.

Опис елементів веб-інтерфейсу:

1. Реєстраційна форма: користувачі вказують свою електронну адресу, пароль, псевдонім і обов'язкове ім'я користувача Telegram для створення облікового запису.
2. Форма входу: користувачі вводять свої облікові дані (електронну пошту та пароль) або ім'я користувача Telegram для доступу до свого облікового запису.
3. Поле введення пошуку: користувачі вводять пошукові запити на основі своїх уподобань.
4. Кнопка «Створити»: користувачі натискають цю кнопку, щоб створити відкладені пошукові запити за вказаними критеріями.
5. Відображення результатів: система представляє результати пошуку, включаючи документи HTML, CSS, JavaScript, зображення та статті новин, на основі запитів користувачів.

Ще трішки про інтерфейс Telegram Bot та веб-інтерфейс. Інтерфейс бота реалізовано за допомогою бібліотеки aiogram Python, яка забезпечує просту у використанні структуру для взаємодії з API Telegram Bot. Бібліотека дозволяє надсилати та отримувати повідомлення, обробляти команди користувача та безперебійно керувати взаємодією користувачів.

Веб-інтерфейс побудовано з використанням HTML, CSS і JavaScript. HTML використовується для структурування веб-сторінок, CSS використовується для стилізації та макета, а JavaScript використовується для інтерактивності на стороні клієнта. Крім того, фреймворк Flask Python використовується для обробки серверної логіки та полегшення зв'язку між веб-інтерфейсом і основними компонентами системи.

4.3.2 Обґрунтування структури інтерфейсу

Вибрана модульна структура забезпечує послідовність і легкість використання на різних платформах. Впроваджуючи окремі інтерфейси для Telegram Bot і веб-інтерфейсу, ми надаємо користувачам варіанти взаємодії із системою на основі їхніх уподобань і зручності. Крім того, ця структура забезпечує гнучкість у додаванні нових інтерфейсів у майбутньому, наприклад мобільних додатків.

Також важливо зазначити, що інтерфейс користувача пройшов ретельне тестування, щоб переконатися в його ефективності, зручності використання та відповідності вимогам користувача. Тестування прийнятності користувача (UAT) було проведено для збору відгуків і оцінки продуктивності інтерфейсу в реальних сценаріях. Процес тестування передбачав оцінку інтуїтивності навігації, простоти введення, швидкості реагування та загальної задоволеності користувачів. На основі результатів тестування було внесено кілька уточнень і покращень, щоб підвищити функціональність і зручності використання інтерфейсу користувача.

Враховуючи вищезазначені аспекти, ми розробили користувацький інтерфейс, який забезпечує безперебійну та інтуїтивно зрозумілу роботу користувача. Обрана структура разом із її елементами та технологічною реалізацією забезпечує користувачам ефективні можливості взаємодії та легкий доступ до функціональних можливостей системи.

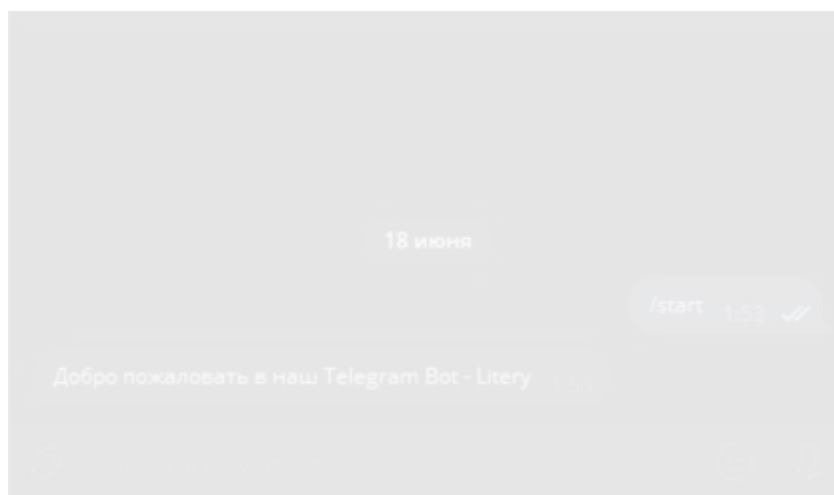


Рисунок 4.2 – Привітання зареєстрованого користувача

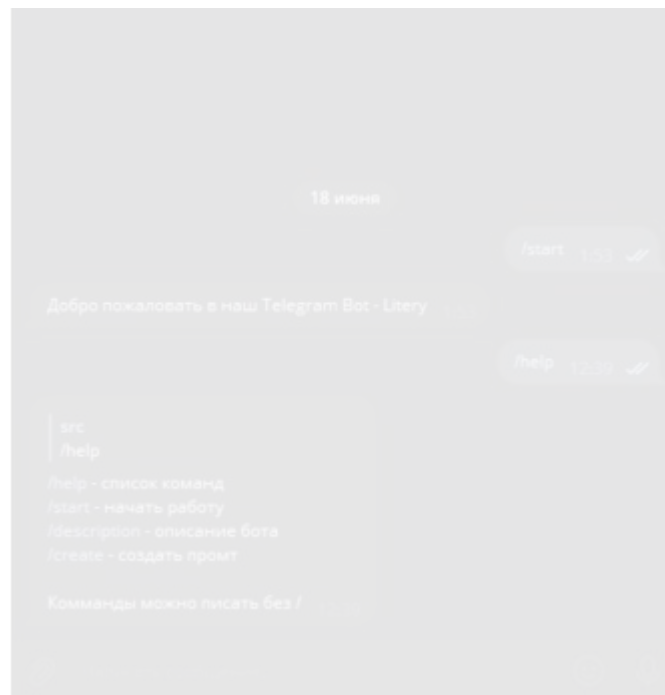


Рисунок 4.3 – Список команд через help-команду

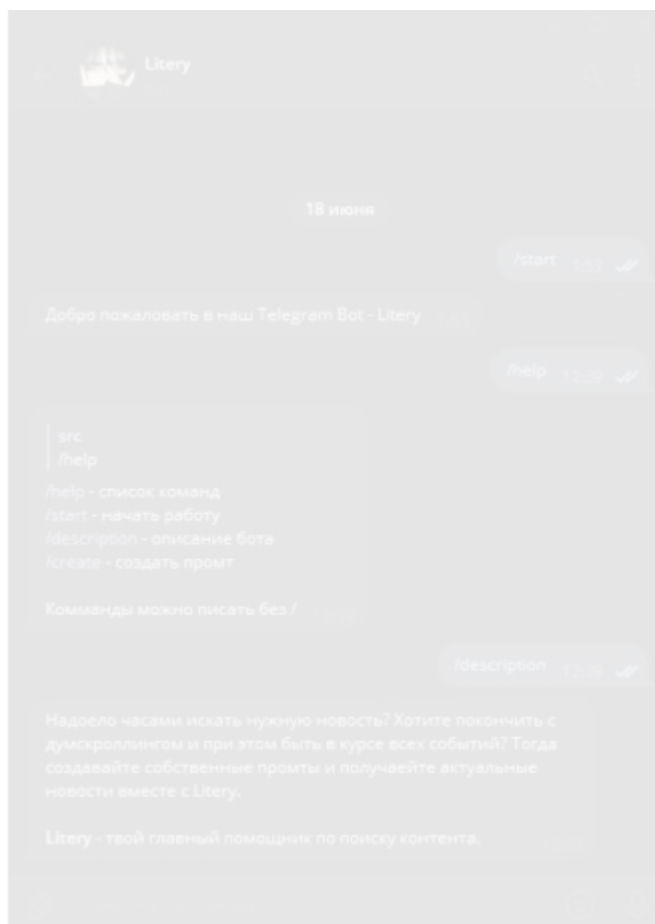


Рисунок 4.4 – Опис бота при написании команды description

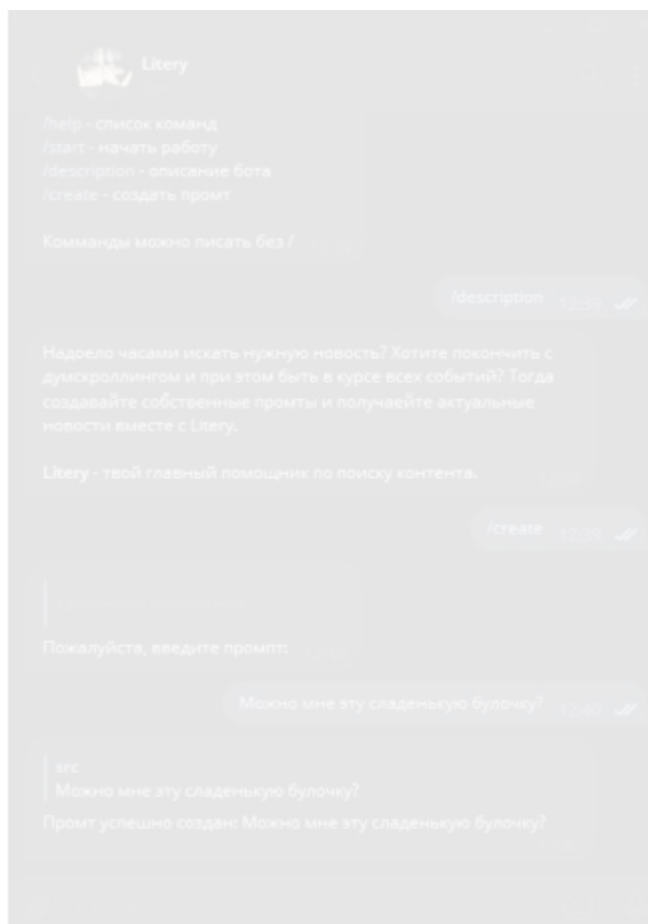


Рисунок 4.5 – Створення відкладеного промту через команду create

5 ТЕСТУВАННЯ І ДОСЛІДНА ЕКСПЛУАТАЦІЯ СИСТЕМИ

5.1 Функціональне тестування

На етапі функціонального тестування функції та можливості порталу перевіряються відповідно до вимог та очікувань. Це включає перевірку правильності роботи функцій, обробки введених користувачем даних, відображення даних і взаємодії з іншими компонентами системи.

Основна мета функціонального тестування — переконатися, що портал функціонує належним чином і відповідає заданим вимогам. Процес тестування передбачає систематичну перевірку кожної функції та функції системи для перевірки її правильності, надійності та зручності використання.

Почнемо з головного для користувача та ресурсу – це реєстрація. Даний крок потрібен системі для коректного ідентифікування користувачів та надання їм усіх функцій ресурсу, які потребують зберегти якісь дані для подальшої обробки.

Процес реєстрації користувача дозволяє окремим особам створити обліковий запис на веб-сайті або зареєструватися через бота Telegram. Для реєстрації потрібна така інформація: електронна пошта, пароль, ім'я користувача та, за бажанням, ім'я користувача Telegram у форматі «@ім'я користувача». Система надає користувачам гнучкість у виборі бажаного методу реєстрації.

5.1.1 Реєстрація на веб-сайті

Опис: цей тест перевіряє процес реєстрації на веб-сайті, гарантуючи, що необхідну інформацію введено правильно, а система точно обробляє потік реєстрації.

Очікуваний результат: система має успішно зареєструвати користувача та зберегти його інформацію в базі даних.

Кроки тесту: перейдіть на сторінку реєстрації, рис. 5.2 та уведіть дійсну електронну адресу, пароль та ім'я користувача рис. 5.3. За бажанням введіть ім'я користувача Telegram. Натисніть кнопку «Зареєструватися».

Переконаємося, що інформація про користувача правильно зберігається в базі даних рис. 5.4.

5.1.2 Реєстрація через бота Telegram

Опис: цей тестовий приклад перевіряє процес реєстрації через бота Telegram, гарантуючи, що необхідну інформацію введено правильно, а система правильно призначає ім'я користувача та ім'я користувача Telegram.

Очікуваний результат: система повинна успішно зареєструвати користувача через бота Telegram і зберегти його інформацію в базі даних.

Кроки тесту: увійдіть до бота Telegram і почніть процес реєстрації рис. 5.5. Введіть дійсну адресу електронної пошти та пароль рис. 5.6.

Переконаємося, що інформація про користувача, зокрема призначене ім'я користувача та ім'я користувача Telegram, правильно зберігається в базі даних рис. 5.7.

Проводячи ретельне тестування процесу реєстрації користувачів, ми можемо переконатися, що користувачі можуть успішно створювати облікові записи та надавати точну інформацію. Це допомагає підтримувати цілісність даних і дозволяє персоналізувати роботу користувачів на основі їхніх уподобань і методу реєстрації.

5.1.3 Стартовий інтерфейс та функціонал веб-додатку та Telegram бота

Переходячи до стартової сторінки веб-інтерфейсу за замовчуванням відображаються дев'ять останніх новинних статей, надаючи користувачам

миттєвий огляд останньої інформації. Користувачі можуть прокручувати ці статті та натискати на них, щоб отримати доступ до повного вмісту. Крім того, інтерфейс містить функцію пошуку, яка дозволяє користувачам знаходити певні цікаві статті.

У веб-інтерфейсі доступні два варіанти пошуку. Перше поле пошуку дозволяє користувачам вводити пошукові запити та отримувати відповідні статті на основі введених ключових слів. Ця функція негайного пошуку надає користувачам результати в реальному часі на основі їхніх запитів.

Другий параметр пошуку дозволяє користувачам створювати відкладені пошукові запити. Ця функція дозволяє користувачам бути в курсі конкретних тем, що цікавлять, без активного проведення повторних пошуків.

У майбутньому веб-інтерфейс надасть користувачам додаткові функції та можливості. Користувачі матимуть доступ до своєї історії пошуку, що дозволить їм переглядати свої минулі пошуки та переглядати раніше переглянуті статті. Інтерфейс також включатиме подібну систему, що дозволить користувачам висловлювати свою вдячність за певні статті. Користувачі матимуть можливість створювати власні публікації, ділитися своїми думками та думками щодо конкретних статей, а також залишати коментарі до статей для участі в обговореннях з іншими користувачами.

У верхньому лівому куті веб-інтерфейсу відображається аватар і псевдонім користувача, створюючи персоналізований досвід. Ця функція дозволяє користувачам ідентифікувати свою особу в системі та створює відчуття спільності та взаємодії. Крім того, оскільки в майбутньому веб-інтерфейс планується інтегрувати в кросплатформну структуру Electron JS, інтерфейс містить мінімалістичні кнопки, розроблені відповідно до загальної естетики інтерфейсу, замінюючи стандартні кнопки керування вікном.

Щоб покращити зручність використання та надати користувачам розширені параметри налаштування, у верхній частині сторінки буде доступна панель завдань або панель. Ця панель містить такі функції, як

доступ до історії пошуку користувача, керування статтями, що сподобалися, і надання доступу до розширених налаштувань. Згодом ці додаткові налаштування та параметри налаштування будуть представлені, щоб надати користувачам більше можливостей контролю та адаптувати веб-інтерфейс до їхніх уподобань.

Telegram-бот має практично ідентичний функціонал, крім деяких адаптацій, враховуючи особливості інтерфейсу Телеграма. При реєстрації в роботі користувачу не буде показано відразу дев'ять останніх новин, оскільки такий обсяг інформації може бути незручний в рамках Telegram-інтерфейсу. Однак, решта функцій залишається доступною.

Користувачі можуть використовувати пошукову функцію бота, щоб знайти конкретні статті новин за ключовими словами. Вони можуть також створювати відкладені пошукові запити та налаштовувати повідомлення для отримання оповіщень про нові статті, які відповідають їхнім критеріям.

Також, у Telegram-боті користувачі зможуть переглядати історію своїх пошукових запитів, переглядати статті, яким вони поставили лайк, та отримувати доступ до розширених налаштувань. Хоча обмеженість інтерфейсу Телеграма можуть обмежувати деякі функціональності, будуть передбачені механізми, що дозволяють користувачам максимально використовувати можливості бота.

У результаті, як веб-інтерфейс, так і телеграм-бот будуть надавати користувачам схожий набір функцій, забезпечуючи їм доступ до новин, можливість пошуку, створення відкладених запитів, взаємодії з контентом і в майбутньому розширених налаштувань.

5.1.4 Взаємодія з компонентами системи

Веб-інтерфейс надає користувачеві різні взаємодії з компонентами системи рис. 5.1. Ось деякі з них:

1. Пошукові плашки: на головній аркуш інтерфейсу є два пошукові плашки. При натисканні на першу плашку користувач може ввести пошуковий запит для негайного пошуку статей, що відповідають його запиту рис. 5.8. При натисканні на другу плашку, користувач може створити відкладений пошуковий запит, щоб отримувати повідомлення про нові статті, які відповідають цим умовам рис. 5.9 та рис. 5.10.
2. Коло зі стрілкою "Новини дня": При натисканні на цей елемент інтерфейсу, користувач буде перенаправлений на сторінку, де відображаються останні доступні новини. Це надає швидкий доступ до найсвіжіших статей без необхідності вводити конкретні запити рис. 5.11.
3. Якщо користувач натискає на окрему новину, його перенаправить на оригінальний ресурс, де була опублікована ця стаття. Таким чином, користувач може отримати повний контекст та подробиці новини, перейшовши на відповідний зовнішній сайт рис. 5.12.
4. Оновлення сторінки: Оскільки система постійно оновлює статті та базу даних новин, при оновленні сторінки користувач може бачити зміни в новинах, що відображаються. Це дозволяє підтримувати актуальність контенту та надавати користувачеві свіжу інформацію під час кожного оновлення інтерфейсу рис. 5.13.

Такі взаємодії з компонентами системи у веб-інтерфейсі забезпечують зручність та гнучкість у використанні, дозволяючи користувачам швидко та легко знаходити новини, виконувати пошукові запити та отримувати доступ до повного контексту кожної статті.

5.1.5 Тестування функцій

Веб-інтерфейс надає широкий спектр функціональності, і загалом усі системи працюють чудово. Однак, є невелика проблема з функцією пошуку та відкладеного пошуку у веб-інтерфейсі, пов'язана з відображенням результатів пошуку.

Проблема полягає в тому, що в HTML-документі закладені блоки для відображення статей, і передбачається, що відобразатиметься до 9 статей. Якщо результати пошуку містять менше або більше статей, ніж передбачається, блоки стають в ряд, що може бути трохи помітно і неестетично. Проте сама функція пошуку працює належним чином і знаходить відповідні статті.

На відміну від веб-інтерфейсу, проблема з відображенням не виникає в телеграм-боті, оскільки там немає обмежень на кількість статей, що відображаються при пошуку. Це з тим, що у телеграм-боті відображаються всі знайдені статті, незалежно від кількості.

Надалі, при здобутті необхідних навичок, цю проблему можна виправити, переглянувши логіку відображення статей під час пошуку та розробивши адаптивний механізм, який коректно відобразить статті, навіть якщо їх кількість менша або більша від очікуваного.

Крім того, важливо зазначити, що якщо користувач не вказав свій нік з Telegram при реєстрації на веб-порталі, при вході в телеграм-бот йому доведеться створити новий обліковий запис для повноцінного використання функцій бота. Це також може бути вирішено та виправлено в майбутньому. Ця проблема не буде стосуватися вас, якщо нік буде вказано при реєстрації на веб-порталі.

Незважаючи на вказані невеликі проблеми, в іншому всі системи працюють відмінно, забезпечуючи користувачу повний функціонал для читання новин, пошуку, відкладеного пошуку, переходу до оригінальних

джерел статей, а також взаємодії з іншими функціями, такими як історія пошуку, лайки, коментарі та розширені налаштування.

Весь функціональний склад системи повністю відповідає встановленим вимогам, крім описаних раніше невеликих проблем. Тим не менш, ці проблеми не суттєво впливають на функціональність та роботу системи в цілому.

При тестуванні системи на навантаження за допомогою п'яти користувачів було виявлено, що один користувач навантажує серверну частину системи на 0.2% процесорного часу на персональному комп'ютері. Це свідчить про хорошу оптимізацію серверної частини ресурсу та ефективне використання ресурсів. На основі цього спостереження можна припустити, що один комп'ютер здатний обслуговувати приблизно п'ять тисяч користувачів одночасно. Однак, слід зазначити, що це припущення виходить із проведеного тестування і може змінюватись в залежності від різних факторів, таких як характеристики серверного обладнання, тип навантаження та інші фактори.

В цілому, на основі наявних ознак та проведеного тестування можна зробити висновок про хорошу оптимізацію та здатність системи обслуговувати значну кількість користувачів одночасно. Однак, для більш точної оцінки та масштабування системи рекомендується проводити додаткові тести та аналізувати результати з урахуванням конкретних вимог та умов експлуатації.

60

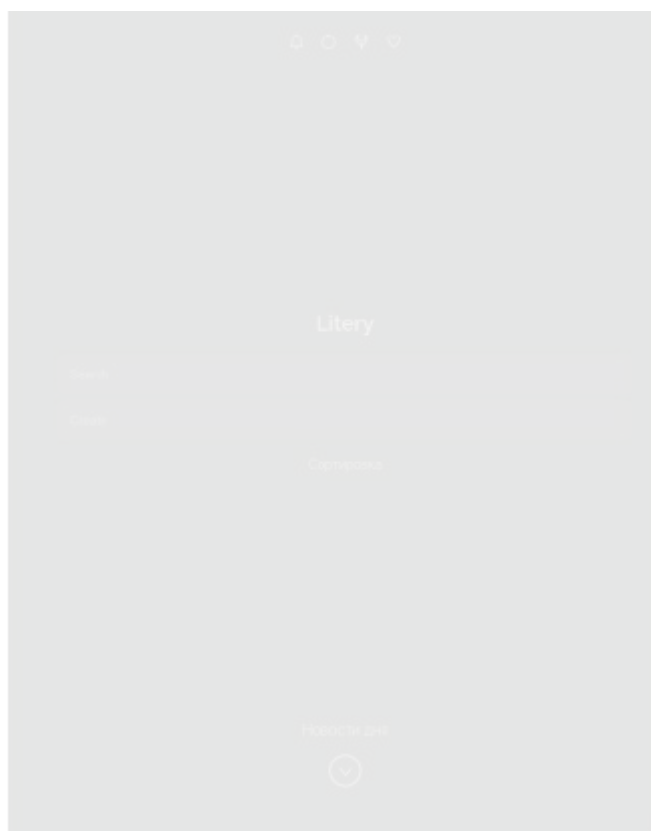


Рисунок 5.1 – Загальний інтерфейс веб-сторінки

61

Create Account

Email *
example@gmail.com

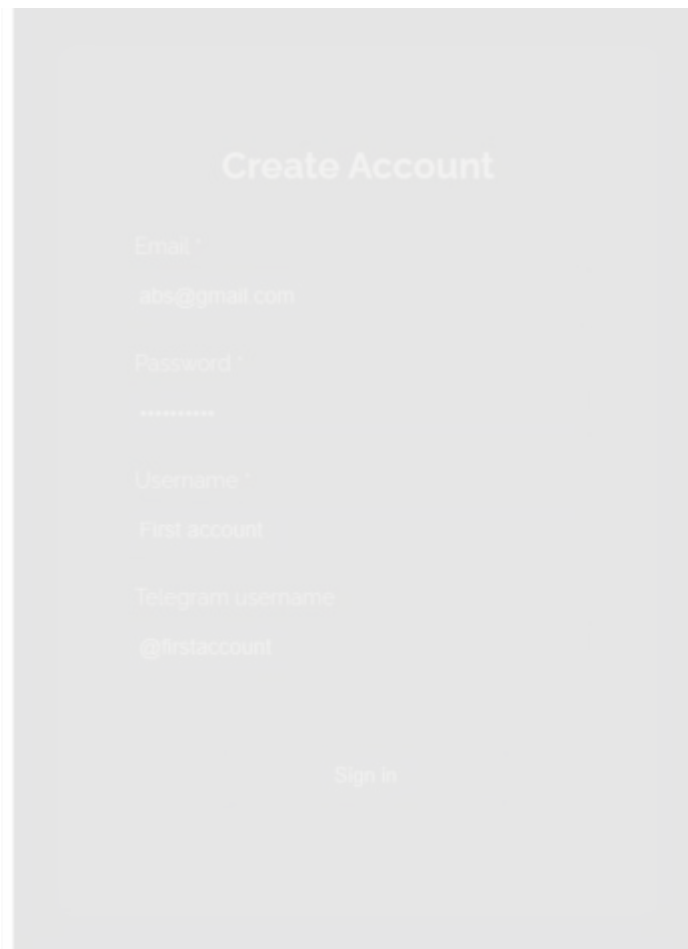
Password *

Username *

Telegram username
@username

Sign in

Рисунок 5.2 – Реєстрація користувача на веб-аркуш



The image shows a 'Create Account' web form. It has a title 'Create Account' at the top. Below it are five input fields: 'Email *' with the value 'abs@gmail.com', 'Password *' with a masked value '*****', 'Username *' with the value 'First account', and 'Telegram username' with the value '@firstaccount'. At the bottom of the form is a 'Sign in' button.

Рисунок 5.3 – Вхідні дані для реєстрації користувача на веб-аркуш

abs@gmail.com	1234qwerly	@firstaccount	First account
---------------	------------	---------------	---------------

Рисунок 5.4 – Дані було внесено до бази даних

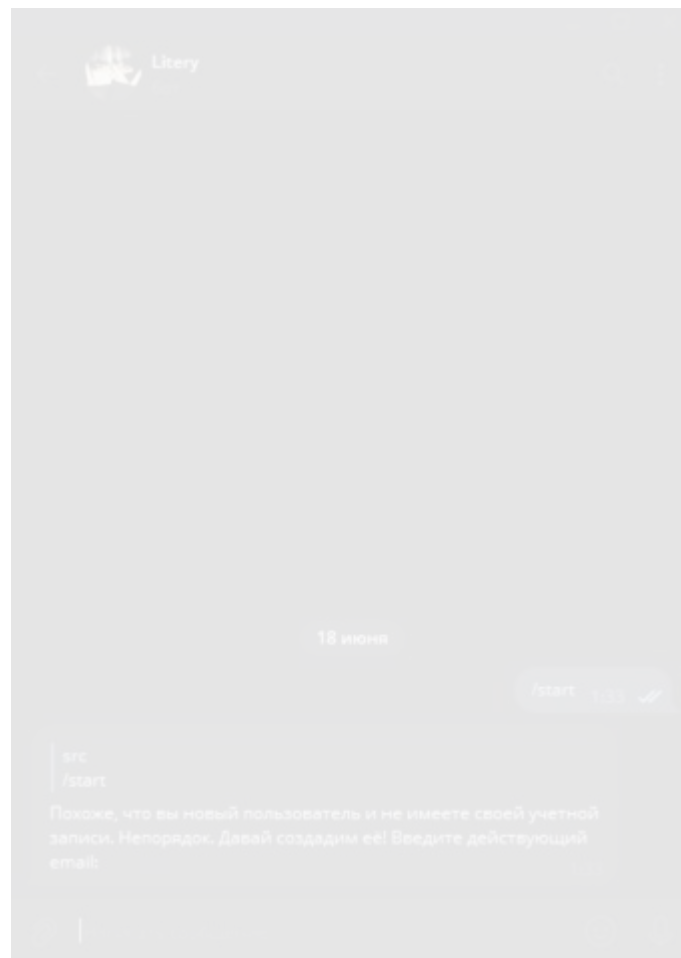


Рисунок 5.5 – Реєстрація у Telegram боті

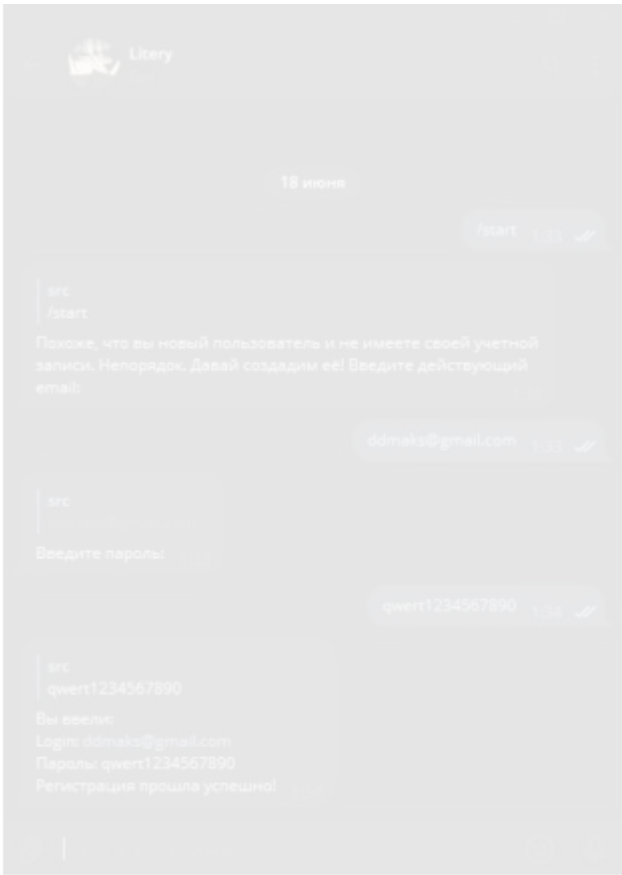


Рисунок 5.6 – Внесения необходимых данных про реестрації у боті



Рисунок 5.7 – Дані було внесено до бази даних

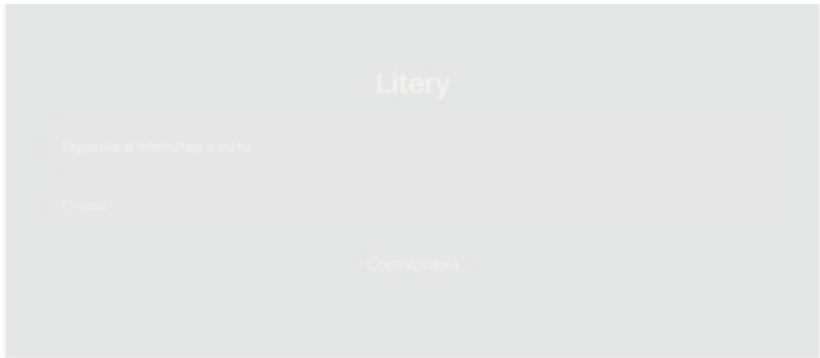


Рисунок 5.8 – Пошуковий запит

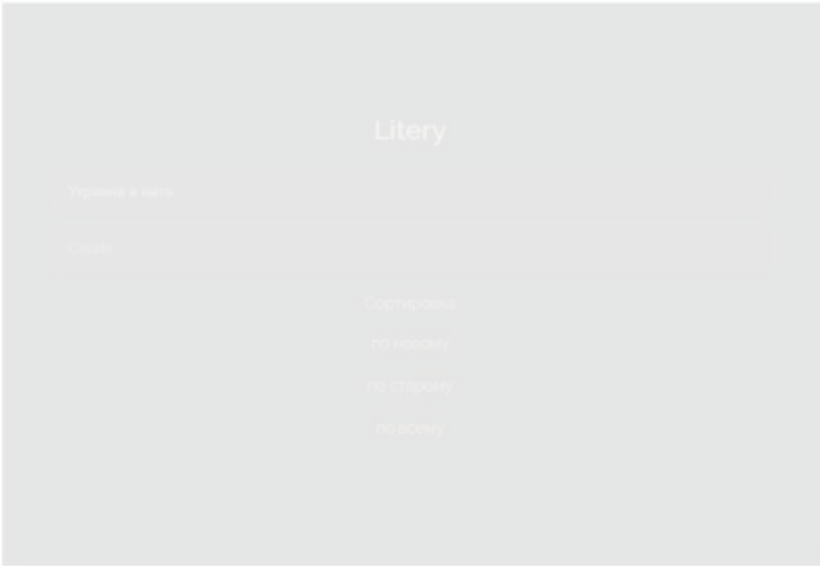


Рисунок 5.8 – Розгорнутий пошуковий запит (для наглядності)

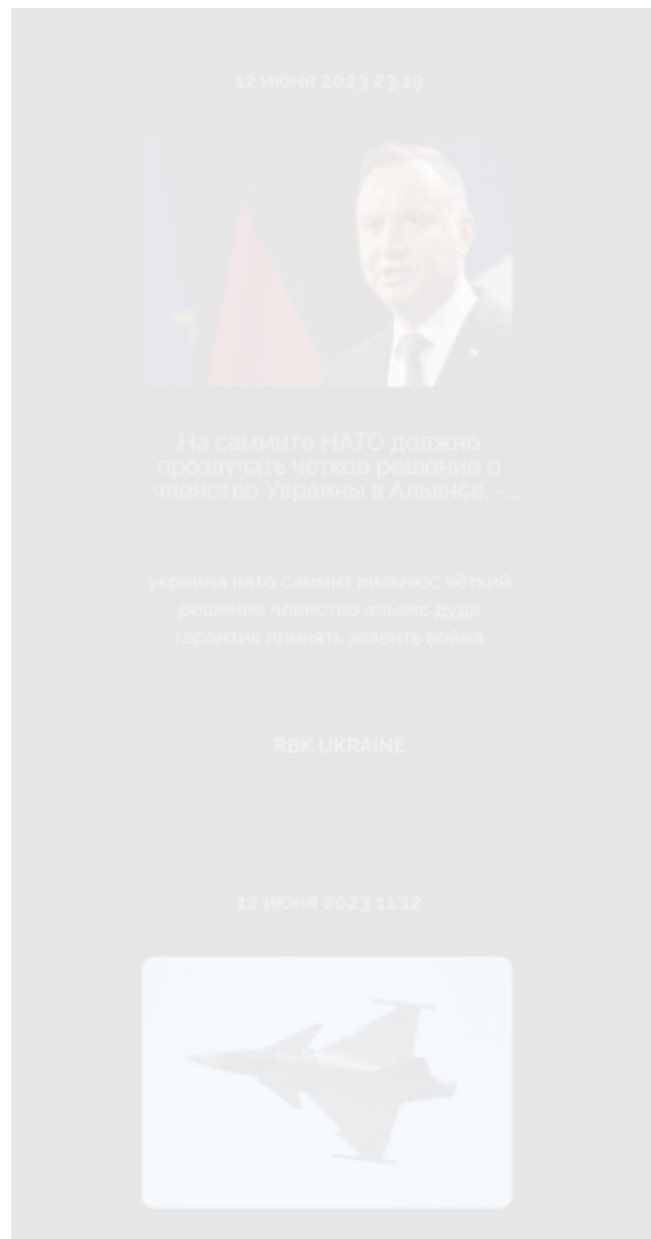


Рисунок 5.9 – Результат поискового запиту

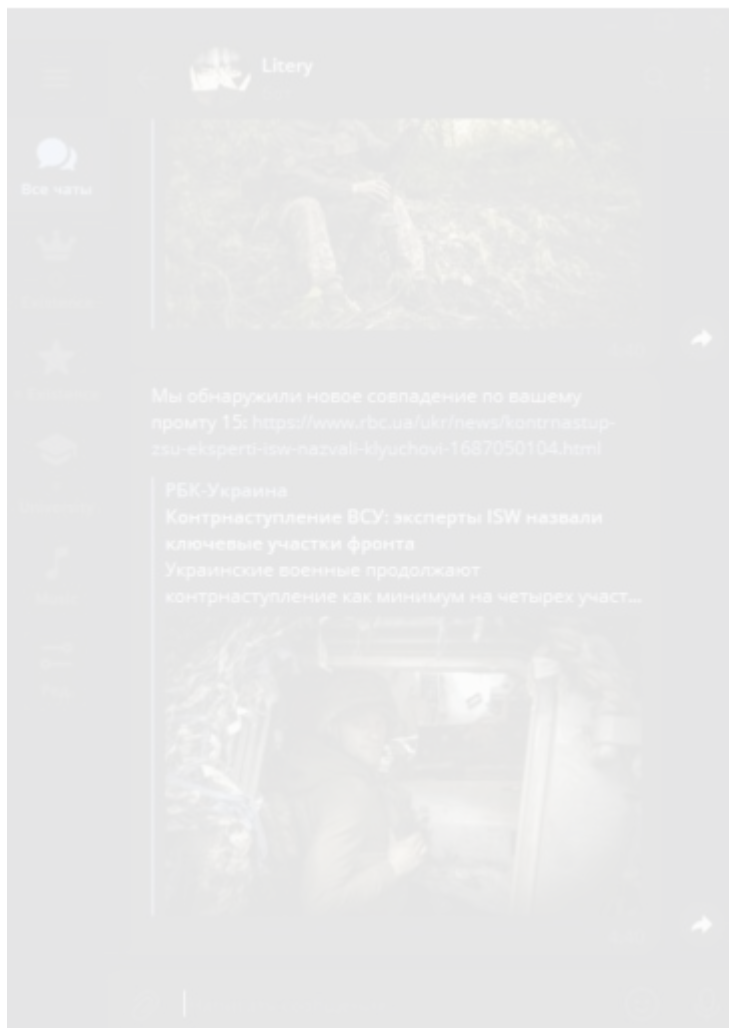


Рисунок 5.10 – Результат повідомлення у відкладеного пошуку

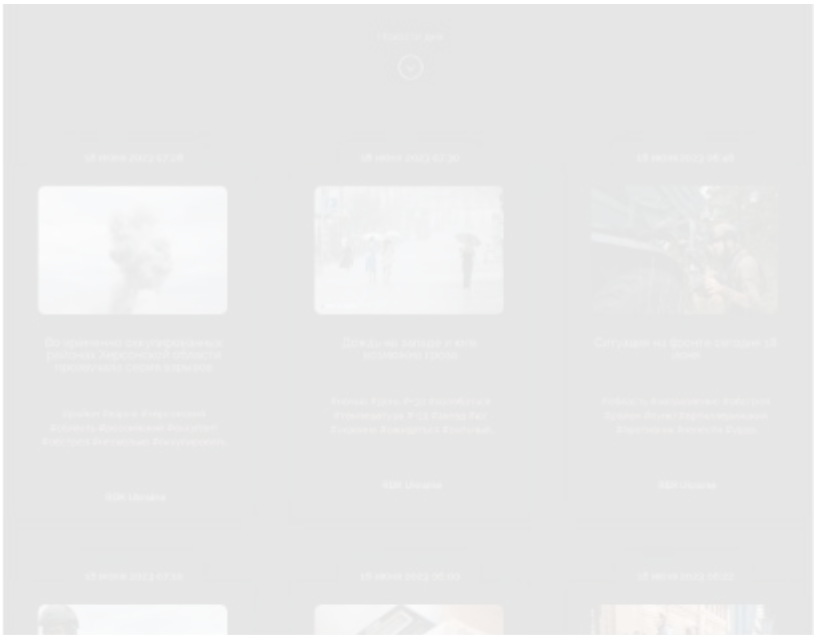


Рисунок 5.11 – Кнопка перехода до останніх новин

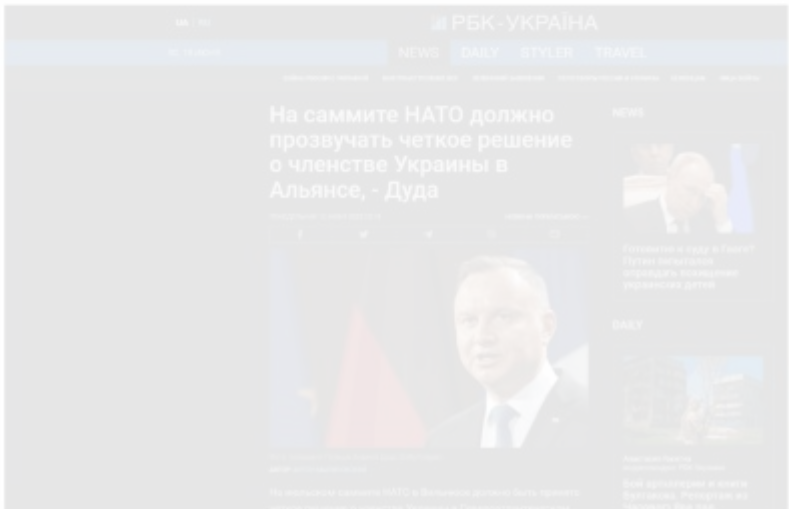


Рисунок 5.12 – Перехід до новинної статті

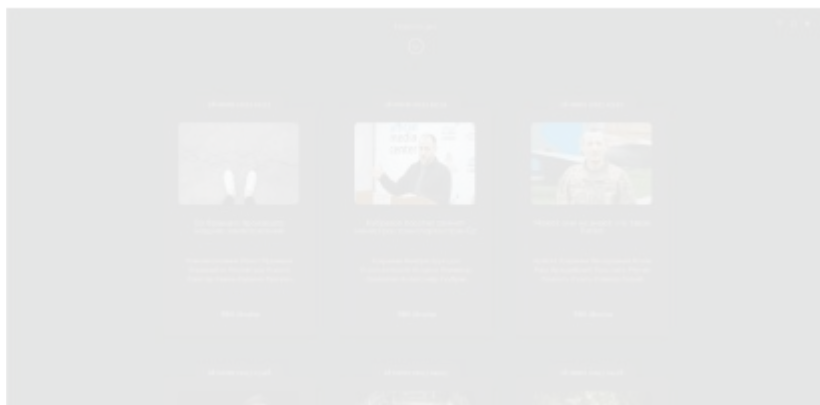


Рисунок 5.13 – Оновлення сторінки

5.2 Тестування взаємодії з користувачем

Тестування взаємодії з користувачем спрямоване на оцінку зручності та ефективності взаємодії користувача з порталом. Це включає оцінку навігації, компонування елементів інтерфейсу, чіткості повідомлень і взаємодії з різними функціями. Давайте окремо обговоримо тестування для вашого сайту та Telegram-бота.

Під час тестування взаємодії з користувачем веб-сайту оцінювалися наступні аспекти:

1. Навігація: навігацію веб-сайту було оцінено, щоб переконатися, що вона інтуїтивно зрозуміла та проста у використанні. Користувачі матимуть можливість переміщатися між різними розділами та отримувати доступ до функцій без плутанини.
2. Макет інтерфейсу: було переглянуто розташування елементів інтерфейсу, таких як панелі пошуку, кнопки та статті новин. Макет є візуально привабливим, організованим і сприяти плавній взаємодії.

3. Функціональна взаємодія: різноманітні функції, такі як пошук, та перегляд статей, перевірялися на їх чуйність і простоту використання. Користувачі повинні мають можливість безперешкодно взаємодіяти з цими функціями та досягати поставлених цілей.

На основі тестування взаємодії з користувачем було встановлено, що веб-сайт забезпечує задовільну взаємодію з користувачем, з інтуїтивно зрозумілою навігацією, добре розробленими елементами інтерфейсу та плавною взаємодією з доступними функціями. Однак згадані раніше проблеми, пов'язані з функціями пошуку слід вирішити, щоб ще більше покращити роботу користувача.

Під час тестування взаємодії Telegram-бота з користувачем оцінювалися наступні аспекти:

1. Виконання команд: була оцінена здатність бота виконувати команди та відповідати на запити користувачів. Користувачі повинні мають можливість взаємодіяти з ботом, надсилаючи повідомлення та миттєво отримуючи відповідні відповіді.
2. Функціональна взаємодія: функціональні можливості, які пропонує бот, такі як пошук новин та создание відкладених пошукових запитів, були перевірені. Користувачі повинні мати можливість ефективно взаємодіяти з цими функціями за допомогою інтуїтивно зрозумілого використання команд.

На основі тестування взаємодії з користувачем було визначено, що бот Telegram забезпечує безперебійну роботу користувача з ефективним виконанням команд, простою навігацією по меню та плавною взаємодією з доступними функціями. Відсутність згаданих раніше обмежень, пов'язаних із відображенням результатів пошуку та затримкою пошукових запитів, сприяє більш оптимізованій взаємодії з користувачем.

На основі тестування взаємодії з користувачем було встановлено, що бот Telegram забезпечує безперебійну взаємодію з користувачем, використовуючи переваги добре налагодженого та зручного інтерфейсу Telegram. Ефективне виконання команд ботом, проста навігація по меню та плавна взаємодія з доступними функціями відповідають очікуванім стандартам, встановленим інтерфейсом Telegram. Відсутність згаданих раніше обмежень, пов'язаних із відображенням результатів пошуку та затримкою пошукових запитів, ще більше сприяє зручності та зручності використання бота.

Підсумовуючи, тестування взаємодії бота Telegram з користувачем підтвердило, що він пропонує зручний досвід завдяки знайомому та інтуїтивно зрозумілому інтерфейсу Telegram. Користувачі можуть легко взаємодіяти з ботом, використовуючи стандартні функції та команди Telegram. Визначені області для вдосконалення, як згадувалося раніше, повинні бути розглянуті, щоб ще більше покращити загальну взаємодію з користувачем бота.

5.3 Тестування безпеки

У процесі розробки порталу важливо провести комплексне тестування безпеки, щоб виявити потенційні вразливості та встановити захист від них. Однак слід зазначити, що програма наразі не має шифрування даних і не пройшла формальної оцінки вразливості, за винятком перевірки введених користувачем даних для забезпечення належної обробки запитів. Отже, існує потреба в суттєвих покращеннях з точки зору безпеки.

Механізми автентифікації та авторизації реалізовано як у веб-інтерфейсі, так і в боті Telegram. Ці механізми вимагають від користувачів надання облікових даних для доступу до своїх облікових записів і виконання

авторизованих дій у системі. Вони відіграють вирішальну роль у захисті даних користувачів і обмеженні доступу до конфіденційної інформації.

Однак у контексті забезпечення безпеки важливо вирішити деякі проблеми.

1. Шифрування даних: на даний момент програма не використовує шифрування даних для безпечної передачі та зберігання. Це створює ризик для конфіденційності та цілісності даних користувача. Застосування надійних засобів шифрування допоможе захистити конфіденційну інформацію від несанкціонованого доступу та забезпечити її збереження.
2. Оцінка вразливості: програма не пройшла офіційну оцінку вразливості, яка є життєво важливою для виявлення та усунення потенційних вразливих місць безпеки. Проведення комплексної оцінки допоможе виявити вразливі місця та зміцнити загальну безпеку системи.

Для забезпечення безпеки порталу дуже важливо визначити пріоритети впровадження шифрування, провести оцінку вразливості, а також постійно контролювати та оновлювати заходи безпеки. Завдяки цьому програма зможе створити більш безпечне та захищене середовище для взаємодії користувачів та захистити конфіденційну інформацію від несанкціонованого доступу.

З метою забезпечення високого рівня безпеки необхідно також акцентувати увагу на наступних аспектах:

1. Багатофакторна аутентифікація: реалізація системи, яка базується на багатофакторній аутентифікації, надасть додатковий рівень захисту від несанкціонованого доступу до облікових записів користувачів. Це дозволить усунути можливість несанкціонованого використання та захистити їх конфіденційні дані.
2. Актуалізація безпеки: регулярне оновлення системи безпеки є важливим фактором для запобігання новим загрозам і атакам.

Постійний моніторинг уразливостей та своєчасне впровадження виправлень допоможуть підтримувати високий рівень захисту даних та зберігати безпеку системи на актуальному рівні.

Таким чином, акцентування уваги на шифруванні даних, проведення оцінки вразливості, реалізація багатофакторної автентифікації та регулярне оновлення безпеки є ключовими факторами у створенні безпечного та надійного середовища для взаємодії користувачів та захисту їхньої конфіденційної інформації від несанкціонованого доступу.

5.4 Тестування продуктивності

Тестування продуктивності оцінює, наскільки добре система працює за різних навантажень і в реальних умовах. Він передбачає тестування продуктивності системи, швидкості завантаження сторінок, масштабованості та стабільності за різних рівнів навантаження.

1. Продуктивність системи: розроблена система демонструє відмінну продуктивність як під навантаженням, так і в нормальних умовах. Як згадувалося раніше в розділі 5.1, система здатна ефективно обробляти запити від приблизно п'яти тисяч користувачів одночасно. Це вказує на те, що архітектура та інфраструктура системи достатньо надійні, щоб обробляти значний трафік без шкоди для продуктивності.

2. Швидкість завантаження сторінки: сторінки веб-інтерфейсу завантажуються швидко, забезпечуючи зручну взаємодію з користувачем. Користувачі можуть отримати доступ до потрібної інформації без помітних затримок, забезпечуючи ефективну навігацію по порталі. Ця оперативність підвищує задоволеність користувачів і взаємодію з платформою.

3. Масштабованість: хоча конкретні деталі масштабованості прямо не згадувалися, система була розроблена та розроблена з урахуванням масштабованості. Масштабованість означає здатність системи обробляти

зростаючий обсяг робочого навантаження або користувачів без значного зниження продуктивності. Прийнявши масштабовану архітектуру, ефективний дизайн бази даних і впровадивши відповідні механізми кешування, система може задовольнити майбутнє зростання та підвищений попит користувачів.

4. Стабільність під навантаженням: система демонструє стабільність навіть при високих навантаженнях. Він може підтримувати свою продуктивність і швидкодію, забезпечуючи доступ користувачів до порталу та взаємодію з ним без значних збоїв або простоїв. Ця стабільність під навантаженням сприяє позитивному досвіду користувача та дозволяє системі ефективно справлятися з періодами пікового використання.

Хоча система демонструє задовільну продуктивність і стабільність, важливо відзначити, що тестування та оптимізація проводилися з урахуванням наявних знань і ресурсів. Подальші вдосконалення та тонке налаштування можуть знадобитися в міру розвитку бази користувачів і вимог. Регулярний моніторинг, профілювання продуктивності та відгуки від користувачів можуть допомогти визначити сфери, які потрібно вдосконалити, і гарантувати, що система продовжує забезпечувати оптимальну продуктивність з часом.

5.5 Тестування на сумісність

Під час тестування на сумісність оцінюється функціональність і узгодженість відображення порталу на різних пристроях, операційних системах і веб-браузерах. Це забезпечує сумісність і послідовне відтворення на різних платформах.

1. Сумісність із веб-переглядачем. Версію порталу для веб-переглядача перевірено на одному пристрої під керуванням Windows 10 за допомогою популярних інструментів налагодження веб-розробки. Веб-ресурс

продемонстрував адаптивність і сумісність з різними браузерами, забезпечуючи узгоджену взаємодію з користувачем. Однак важливо зазначити, що комплексне тестування на більш широкому діапазоні браузерів і версій, включаючи мобільні браузери, не проводилося.

2. Сумісність операційної системи: Тестування сумісності між різними операційними системами, такими як iOS, Android та іншими, не проводилося спеціально для веб-інтерфейсу. Важливо визнати, що сумісність і продуктивність порталу на цих платформах можуть відрізнятися через відмінності в механізмах візуалізації браузера та особливості платформи.

3. Сумісність Telegram Bot: у випадку Telegram bot, сумісність повинна залишатися незмінною в різних операційних системах. Telegram — це програма для обміну повідомленнями на різних платформах, а це означає, що функціональність бота та взаємодія з користувачем мають бути однаковими незалежно від базової операційної системи. Сумісність бота не прив'язана до конкретної операційної системи, що забезпечує однакову взаємодію для користувачів на різних платформах.

Хоча браузерна версія порталу продемонструвала сумісність і адаптивність на тестованому пристрої, рекомендується провести комплексне тестування на сумісність на кількох пристроях, операційних системах і комбінаціях браузерів, щоб забезпечити безперебійну роботу користувача для ширшої аудиторії. Регулярний моніторинг і відгуки від користувачів на різних платформах можуть допомогти виявити будь-які проблеми сумісності, які можуть виникнути, і дозволити внести необхідні коригування для покращення загальної сумісності та узгодженості між платформами.

5.6 Тестування помилок

Тестування помилок допомагає виявити та виправити дефекти, помилки програмного забезпечення та інші проблеми, які можуть вплинути

на функціональність порталу. Це включає тестування вхідних даних, обробку помилок, обробку винятків і відновлення системи.

1. Тестування вхідних даних: програма обробляє вхідні дані як рядки та використовує інструменти Python для обробки рядків, що значно зменшує ймовірність помилок введення. Розглядаючи всі вхідні дані як рядки, програма гарантує належну обробку неочікуваних або недійсних форматів вхідних даних.

2. Обробка помилок: у разі помилок під час аналізу чи обробки даних система включає механізм реєстрації помилок. Ця система реєстрації фіксує коди помилок і повідомлення, полегшуючи діагностику та вирішення помилок. Зареєстровані помилки можна проаналізувати для виявлення шаблонів або повторюваних проблем, що дозволить розробникам виправити їх і підвищити загальну надійність системи.

3. Обробка винятків: Програма реалізує методи обробки винятків для ефективної обробки несподіваних ситуацій і запобігання збоєм системи. Перехоплюючи та обробляючи винятки, програма може відновлюватися після помилок і продовжувати виконання без збоїв. Це гарантує стабільність і безперебійну роботу порталу.

4. Відновлення системи: для забезпечення стійкості та безперервності було реалізовано автоматизовану систему аналізу. Ця система записує свої дії та прогрес, дозволяючи їй відновити синтаксичний аналіз із того місця, де воно було зупинено, у разі перебоїв у системі або збоєм живлення. Ця функція гарантує цілісність даних і усуває необхідність перезапуску процесу аналізу з самого початку, підвищуючи ефективність і мінімізуючи втрату даних.

Використовуючи ці стратегії перевірки помилок, програма має на меті забезпечити надійну роботу порталу. Постійний моніторинг, регулярний аналіз помилок і впровадження відповідних виправлень допоможуть підвищити стабільність системи та можливості обробки помилок з часом.

ВИСНОВКИ

На основі проведеного аналізу та оцінки проекту зробимо вичерпні висновки на основі досягнених результатів та сфери вдосконалення.

Почнемо з досягнутих результатів:

1. Проект успішно досяг своїх цілей, які включали розробку програмного забезпечення для збору статей новин і токенизації, персоналізованих рекомендацій і боротьби з doomscrolling.
2. Ефективність системи в боротьбі з думскролінгом оцінюється в 40%, що вказує на значний вплив на пом'якшення надмірного споживання негативних новин.
3. Виявлені недоліки включають відсутність шифрування даних та певні функціональні обмеження. Проект потребує подальших доопрацювань, таких як реалізація можливостей редагування облікових записів користувачів та надання функцій перегляду статистики.
4. Основним досягненням проекту є досягнення 40% ефективності та успішна обробка значного обсягу роботи та інформації за короткий проміжок часу.

Щоб оцінити вплив проекту на скорочення часу, витраченого на гортання стрічок новин, був проведений експеримент за участю шести осіб. Експеримент вимірював час, витрачений на перегляд новин із використанням ресурсів проекту та без нього. Результати різнилися між учасниками: двоє показали скорочення витраченого часу на 32%, тоді як решта чотирьох зазнали значного скорочення на 48%. Ці скорочення витраченого часу вказують на суттєве покращення в управлінні звичками споживання новин.

Крім того, скорочення часу, витраченого на прокручування стрічок новин, позитивно вплинуло на самопочуття учасників. Вони повідомили про такі покращення, як зниження рівня стресу, покращення якості сну, зниження

загальної тривоги та піднесений настрій. Отримані дані свідчать про те, що, скоротивши час, витрачений на поведінку doomscrolling, люди відчули відчутні переваги, що призвело до покращення загального самосприйняття та психічного стану.

Ефективність проекту можна вважати винятковою, адже середнє скорочення часу на перегляд новин склало 40%. Цей результат означає значне досягнення в боротьбі з думскролінгом і просуванні здорових звичок споживання новин. Здатність проекту значно скоротити час, який люди присвячують перегляду стрічок новин, демонструє його ефективність у боротьбі з негативними наслідками надмірного споживання новин.

Важливо відзначити, що позитивний вплив проекту виходить за межі кількісної метрики скорочення часу. Розширюючи можливості людей звільнити свій час і встановити більш здорові моделі споживання медіа, проект сприяє їх загальному добробуту, психічному здоров'ю та якості життя.

Підсумовуючи, досягнення проекту щодо скорочення часу, витраченого на прокручування стрічок новин, у середньому на 40% підкреслюють його успіх у боротьбі з поведінкою doomscrolling. Супутнє покращення самопочуття учасників, зокрема зменшення стресу, покращення сну, зменшення тривоги та покращення настрою, підкреслює позитивний вплив проекту на життя людей. Ці результати підтверджують важливість проекту для сприяння більш здоровому та збалансованому підходу до споживання **НОВИН**.

Настав час поговорити про сфери вдосконалення:

1. Рекомендовано вдосконалення в таких сферах: впровадження функцій статистичного аналізу, включення механізмів шифрування даних для забезпечення безпеки даних користувача, вирішення проблем інтерфейсу користувача, розширення діапазону джерел інформації та вдосконалення алгоритмів рекомендацій для

- досягнення максимальної ефективності в боротьбі з доомскроллінгом.
2. Посилення навичок у розробці на стороні сервера та зовнішніх технологіях є важливим для покращення загальної продуктивності системи та взаємодії з користувачем.
 3. Пріоритет для майбутнього розвитку має бути наданий розширенню пулу джерел інформації, вдосконаленню обробки даних і алгоритмів рекомендацій, а також подальшому підвищенню ефективності запобігання думскроллінгу.
 4. Пропозиція інтеграції методів штучного інтелекту (ШІ) для ідентифікації відповідної інформації та підвищення ефективності системи у протидії тенденціям доомскроллінгу.
 5. Слід розглянути можливість інтеграції системи з фреймворком ElectronJS для покращення можливостей сповіщень і забезпечення повної крос-платформної функціональності.

Варто зазначити, що представлені висновки ґрунтуються на глибокому аналізі поточного стану проекту та визначених напрямках для покращення. Рекомендації спрямовані на усунення недоліків і максимізацію потенціалу системи для ефективної боротьби з doomscrolling.

Головна мета проекту — розробка програмного забезпечення для збору новинних статей і токенизації, персоналізованих рекомендацій і боротьби з думскроллінгом — була частково досягнута. Однак для повної реалізації потенціалу проекту вкрай важливо впровадити запропоновані вдосконалення та вдосконалити наявні функціональні можливості.

Подальший розвиток і вдосконалення проекту вимагатиме комплексного підходу, що поєднує технічний досвід, інноваційні технології та постійний моніторинг галузевих тенденцій. Усуваючи виявлені недоліки та дотримуючись викладених рекомендацій, проект має потенціал для значного

впливу на протидію doomscrolling і надання користувачам персоналізованого досвіду споживання новин.

Проект, провівши демонстрацію своєї здатності вирішувати запропоновані виклики та досягати значних результатів у наміченому масштабі, підтверджує свою ефективність. З урахуванням подальшого розвитку та вдосконалення, він може стати цінним інструментом у просуванні більш здорових звичок споживання новин та пом'якшенні негативного впливу, що супроводжує надмірний вплив несприятливого контенту.

Результати, отримані в ході проекту, говорять про його здатність долати певні виклики та досягати суттєвих цілей у наперед визначеному масштабі. Однак, щоб повністю розкрити його потенціал, потрібен подальший розвиток та вдосконалення. За можливості оптимізації та розширення функціональності цей проект може стати важливим інструментом у стимулюванні здорових звичок споживання новин та пом'якшенні негативного впливу, пов'язаного з надмірною дією негативного контенту.

Проект успішно показав свою здатність вирішувати поставлені завдання та досягати значних результатів у межах свого обмеженого масштабу. Однак, з урахуванням подальшого розвитку та удосконалення, його потенціал може бути повністю розкритим. Будучи ефективним інструментом, цей проект здатний стимулювати здорові звички у споживанні новин та зменшити негативний вплив, пов'язаний із надмірним споживанням несприятливого контенту.

Підбивши підсумки, можна дійти невтішного висновку, що проект успішно впорався з поставленими викликами і досяг значних результатів у межах свого призначення. Однак з урахуванням подальшого розвитку та вдосконалення його потенціал може бути повністю розкритий. Він є цінним інструментом, здатним змінити звички споживання новин у більш здоровому

81

напрямку і пом'якшити негативний вплив, що супроводжує надмірний вплив негативного контенту.

ПЕРЕЛІК ПОСИЛАНЬ

1. Эрик Метиз – Изучаем Python 3-е издание. URL: <https://do.sch24.ru>
2. Марк Лутц – Изучаем Python 5-е издание. URL: <http://flibusta.site>
3. Flask: веб-разработка капля за каплей. URL: <http://flibusta.site>
4. Сергей Дмитриевич Кузнецов – Базы данных. URL: <http://flibusta.site>
5. Книга, Зайцев Пётр, Ткаченко Вадим, и Шварц Бэррон – MySQL по максимуму. 3-е издание. URL: <https://www.pdfdrive.com>
6. Владимир Александрович Дронов – HTML и CSS: 25 уроков для начинающих. URL: <https://www.pdfdrive.com>
7. Хомоненко А.Д. – Базы данных: Учеб. для вузов. URL: <https://www.pdfdrive.com>
8. Дейта К. Дж. – Базы данных. URL: <https://www.pdfdrive.com>
9. Элмасри Р. – Системы управления базами данных. URL: <https://www.pdfdrive.com>
10. Коронель К. – Базы данных: проектирование, реализация и сопровождение. URL: <http://flibusta.site>
11. Маркл Л. – Основы проектирования баз данных. URL: <http://flibusta.site>
12. Майя И. – Building Web Applications with Flask. URL: <http://flibusta.site>
13. Гринберг М. – Flask Web Development. URL: <http://flibusta.site>
14. Аггарвала Ш. – Flask Framework Cookbook. URL: <http://flibusta.site>
15. Flask: веб-разработка капля за каплей. URL: <http://flibusta.site>
16. Дуайер Г. – Flask By Example. URL: <http://flibusta.site>
17. Рамальо Л. – Fluent Python. URL: <http://flibusta.site>
18. Бизли Д. – Python Cookbook. URL: <http://flibusta.site>
19. Слаткин Б. – Effective Python. URL: <http://flibusta.site>
20. Мэттиз Э. – Python Crash Course. URL: <http://flibusta.site>



Совпадения

Источники из Библиотеки

97

1	B17	ID файла: 1015324745	Учебное заведение: National Aerospace University Kharkiv Aviation Institute	4 Источник	1.01%
2	B21	ID файла: 1015315473	Учебное заведение: National Aerospace University Kharkiv Aviation Institute	27 Источник	0.92%
3	B19	ID файла: 1015325439	Учебное заведение: National Aerospace University Kharkiv Aviation Institute		0.87%
4	B04	ID файла: 1015325356	Учебное заведение: National Aerospace University Kharkiv Aviation Institute	2 Источник	0.86%
5	B03	ID файла: 1011520403	Учебное заведение: National Aerospace University Kharkiv Aviation Institute		0.76%
6	B06	ID файла: 1004052294	Учебное заведение: National Aerospace University Kharkiv Aviation Institute	14 Источник	0.46%
7	B03	ID файла: 1008424246	Учебное заведение: National Aerospace University Kharkiv Aviation Institute	18 Источник	0.43%
8	B06	ID файла: 1011508710	Учебное заведение: National Aerospace University Kharkiv Aviation Institute		0.43%
9	M02	ID файла: 1008078725	Учебное заведение: National Aerospace University Kharkiv Aviation Institute		0.37%
10	B19	ID файла: 1004201604	Учебное заведение: National Aerospace University Kharkiv Aviation Institute	2 Источник	0.32%
11	B35	ID файла: 1004176905	Учебное заведение: National Aerospace University Kharkiv Aviation Institute	12 Источник	0.27%
12	B22	ID файла: 1011513030	Учебное заведение: National Aerospace University Kharkiv Aviation Institute		0.26%
13	M10	ID файла: 1009706501	Учебное заведение: National Aerospace University Kharkiv Aviation Institute		0.22%
14	B38	ID файла: 6049584	Учебное заведение: National Aerospace University Kharkiv Aviation Institute		0.16%
15	B34	ID файла: 6050045	Учебное заведение: National Aerospace University Kharkiv Aviation Institute	4 Источник	0.16%
16	B14	ID файла: 1011514205	Учебное заведение: National Aerospace University Kharkiv Aviation Institute		0.15%
17	B07	ID файла: 6010159	Учебное заведение: National Aerospace University Kharkiv Aviation Institute		0.14%
18	B14	ID файла: 1015278238	Учебное заведение: National Aerospace University Kharkiv Aviation Institute		0.12%
19	M03	ID файла: 1009696873	Учебное заведение: National Aerospace University Kharkiv Aviation Institute		0.08%
20	B24	ID файла: 1015324151	Учебное заведение: National Aerospace University Kharkiv Aviation Institute	3 Источник	0.07%