

Детектор Плагіату v. 2961 - Звіт оригінальності: 15.06.2026 9:13:24

Перевірений документ: Бак_пояснювальна_Фурман_ІПЗ_4 (1).docx Ліцензія: ВОЛОДИМИР МАТІЄВСЬКИЙ

? Тип пошуку: Пошук Перепису ? Знайдено мову: Uk

? Тип перевірки: Інтернет-перевірка

ТЕЕ і кодування: DocX n/a

Детальний аналіз документа документа:

? Співвідношення:

Плагіат 1.21% Оригінал 95.43%
Цитати 3.36%



? Графік розподілу:



? Джерела плагіату: 9

2% 1686 1. <https://likestorm.com.ua/blog/telegram/yak-stvoriti-bota-v-telegram-z-nulya/>
2% 1537 2. <https://docs.aiogram.dev/>
2% 1274 3. <https://naurok.com.ua/zasobi-dlya-analizu-danih-v-elektronnih-tablicyah-ms-excel-39430...>

? Дані оброблених ресурсів: 204 - ОК / 10 - Помилок

? Важливі примітки:

Wikipedia:



[не знайдено]

Google Books:



[не знайдено]

Сервіси платних робіт:



[не знайдено]

Античіт:



Знайдена протидія!

? Звіт проти обману UACE:

- Статус: Аналізатор **Увімкнений** Нормалізатор **Увімкнений** схожість символів встановлено **100%**
- Виявлений відсоток забруднення UniCode: **4,8%** з обмеженням: 4%
- Документ не нормалізовано: відсотків не досягнуто 5%
- Усі підозрілі символи будуть позначені фіолетовим кольором: Abcd...
- Знайдено невидимі символи: 0

Рекомендація з оцінки:

Аналізу цього звіту слід приділити особливу увагу! Підозрюється, що цей документ містить значну кількість символів, чужих мові документа. Це є прямим вказівкою на те, що автор документа використав спеціальне програмне забезпечення\онлайн -веб -сервіс, щоб ефективно заплутати текст, намагаючись уникнути потенційного виявлення плагіату. Наполегливо рекомендується ескалювати цю справу! У разі сумнівів зверніться до служби підтримки детектора плагіату!

Статистика алфавіту та аналіз символів:

? Активні посилання (URL-адреси, витягнуті з документа):

1. <https://adminfinance.umw.edu/tess/files/2013/06/Excel-Manual1.pdf>
2. <https://core.telegram.org/bots/api>
3. <https://docs.aiogram.dev/>
4. <https://docs.python.org/>
5. <https://dou.ua/forums/topic/40575/>
6. <https://foxminded.ua/uml-diagramy/>
7. <https://openpyxl.readthedocs.io/>
8. <https://pandas.pydata.org/docs/>
9. <https://stackoverflow.com/>
10. <https://www.geeksforgeeks.org/python/python-programming-language-tutorial/>

? Виключено:

URL не знайдені

? Включено:

URL не знайдені

? Детальний аналіз документу:

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ДЗ

» Знайдено посилань: 0,01%

id: 1

«ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА»

Факультет технологій та інформаційних систем (назва факультету, інституту)
Інформаційних технологій та систем (назва кафедри) Пояснювальна записка до
дипломного проекту (роботи) БАКАЛАВРА (освітньо-кваліфікаційний рівень) на тему:
РОЗРОБКА [TELEGRAM](#)-БОТА ДЛЯ АВТОМАТИЗОВАНОЇ ОБРОБКИ ТА ФІЛЬТРАЦІЇ ДАНИХ З
[EXCEL](#)-ФАЙЛІВ ЗАСОБАМИ [PYTHON](#) Виконав: студент 4 курсу, групи ___ напряму підготовки
(спеціальності) 121

» Знайдено посилань: 0%

id: 2

«Інженерія програмного забезпечення»

(шифр і назва напряму підготовки, спеціальності) Фурман О.А. (прізвище та ініціали)
Керівник Семенов М.А. (прізвище та ініціали) Рецензент (прізвище та ініціали) Лубни – 2026
ЗМІСТ ВСТУПЗ РОЗДІЛ 1. АНАЛІЗ ТЕХНОЛОГІЙ ОБРОБКИ ТА НАДАННЯ ТАБЛИЧНИХ ДАНИХ В
ІНФОРМАЦІЙНИХ СИСТЕМАХ7 1.1. Особливості роботи зі структурованими табличними
даними та форматом [Excel](#)7 1.2. Інструменти [Python](#) для обробки табличних даних10 1.3.
Використання [Telegram](#)-ботів як інтерфейсу доступу до інформаційних систем14 Висновки
до розділу17 РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ОБРОБКИ ДАНИХ НА
ОСНОВІ [TELEGRAM](#)-БОТА18 2.1. Формування функціональних вимог до інформаційної
системи18 2.2. Моделювання структури та архітектури системи засобами [UML](#)21 2.3.
Проектування алгоритмів обробки та фільтрації даних25 Висновки до розділу28 РОЗДІЛ 3.
РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ30 3.1. Реалізація модуля зчитування
та обробки [Excel](#)-даних у середовищі [Python](#)30 3.2. Розробка [Telegram](#)-бота та реалізація
взаємодії з користувачем33 3.3. Розгортання та запуск інформаційної системи на
платформі [PythonAnywhere](#)38 3.4. Тестування інформаційної системи та аналіз результатів
її роботи42 3.5. Перспективи розвитку та масштабування системи48 Висновки до розділу50
ЗАГАЛЬНІ ВИСНОВКИ52 СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ54 ДОДАТОК А Лістинг
програмного коду56 ДОДАТОК Б Структура та приклад даних [Excel](#)-файлу60 ВСТУП
Сучасний етап розвитку інформаційних технологій характеризується стрімкою
діджиталізацією практично всіх сфер суспільного життя. Щодня зростає обсяг даних, що
зберігаються в електронному вигляді, а разом із цим підвищуються вимоги до швидкості їх
обробки, доступності та зручності отримання. Організації, установи та приватні
підприємства дедалі частіше використовують електронні таблиці для ведення обліку,
формування звітності та збереження персоналізованої інформації. Найпоширенішим
інструментом при цьому залишається формат [Excel](#), який поєднує простоту структури та
достатню гнучкість для роботи з даними різного типу. Паралельно з цим активно
розвиваються месенджери як універсальні платформи для комунікації. [Telegram](#)
перетворився не лише на засіб обміну повідомленнями, а й на повноцінне середовище для
створення сервісів автоматизованого інформування. [Telegram](#)-боти дозволяють
організувати зручний діалог між системою та користувачем без необхідності встановлення
додаткового програмного забезпечення або переходу на окремі веб-ресурси. У результаті
формується новий формат цифрової взаємодії - швидкий, інтуїтивний та максимально
наближений до звичного способу спілкування. Актуальність теми зумовлена потребою у
створенні доступних програмних рішень, що забезпечують автоматизований доступ до
структурованих даних без складної серверної інфраструктури. Для звичайних користувачів
подібні сервіси означають економію часу, зменшення кількості помилок при отриманні
інформації та можливість оперативної перевірки персональних показників. Для
програмістів тема є важливою з огляду на пошук оптимального балансу між
функціональністю, простотою реалізації та масштабованістю системи. Розробка
програмного забезпечення, яке інтегрує [Excel](#)-файл як джерело даних із [Telegram](#)-ботом як
інтерфейсом взаємодії, відкриває можливість створення легких, автономних та гнучких
інформаційних сервісів. Особливого значення набуває питання оптимізації процесу обробки
даних у реальному часі. Використання мови програмування [Python](#) у поєднанні з
бібліотекою

» Знайдено посилань: 0%

id: 3

«[pandas](#)»

дозволяє ефективно зчитувати, фільтрувати та аналізувати табличну інформацію. Інтеграція з [Telegram](#) через сучасні асинхронні фреймворки забезпечує стабільну обробку запитів навіть за наявності кількох одночасних звернень. Це створює основу для побудови практичного програмного продукту, придатного до застосування у навчальних закладах, невеликих організаціях або локальних інформаційних системах. Метою роботи є розробка інформаційної системи у вигляді [Telegram](#)-бота для автоматизованої обробки та фільтрації даних з [Excel](#)-файлів засобами [Python](#) із забезпеченням зручної, швидкої та коректної взаємодії з користувачем. Для досягнення поставленої мети передбачається виконання таких завдань: 1. Проаналізувати особливості роботи зі структурованими табличними даними формату [Excel](#) та розглянути інструменти [Python](#), що використовуються для їх обробки в інформаційних системах. 2. Виконати проектування інформаційної системи обробки даних на основі [Telegram](#)-бота, сформулювати функціональні вимоги, змодельовати архітектуру та алгоритми її роботи засобами [UML](#). 3. Реалізувати інформаційну систему для автоматизованої обробки та фільтрації даних з [Excel](#)-файлів, інтегровану з [Telegram](#)-ботом. 4. Провести тестування створеної системи, виконати аналіз результатів її функціонування та визначити можливості подальшого розвитку і масштабування. Об'єктом роботи виступає процес автоматизованої обробки структурованих табличних даних у середовищі [Python](#) із подальшим наданням результатів через [Telegram](#)-інтерфейс. Предметом роботи є програмна реалізація [Telegram](#)-бота, що забезпечує зчитування, фільтрацію та відображення даних із [Excel](#)-файлу відповідно до введених користувачем параметрів. Структура роботи включає вступ, три розділи, загальні висновки, список використаних джерел та додатки. У першому розділі розглянуто теоретичні основи обробки табличних даних, особливості використання формату [Excel](#) та інструментів [Python](#) для роботи зі структурованою інформацією, а також проаналізовано можливості застосування [Telegram](#)-ботів як інтерфейсу доступу до інформаційних систем. У другому розділі виконано проектування інформаційної системи, сформовано функціональні вимоги, розроблено архітектуру та змодельовано структуру системи із використанням [UML](#)-діаграм, а також описано алгоритми обробки та фільтрації даних. У третьому розділі виконано практичну реалізацію розробленої інформаційної системи засобами мови програмування [Python](#). Розглянуто особливості створення модуля зчитування та обробки даних з [Excel](#)-файлів, реалізацію [Telegram](#)-бота та механізми взаємодії з користувачем. Проведено тестування функціональних можливостей системи, проаналізовано результати її роботи та розглянуто процес розгортання інформаційної системи на платформі [PythonAnywhere](#) для забезпечення її безперервного функціонування без використання власного серверного обладнання. Робота спрямована на демонстрацію можливостей створення прикладного програмного забезпечення, яке поєднує простоту використання електронних таблиць із функціональністю сучасних месенджерних сервісів, що дозволяє сформувати зручний і ефективний інструмент цифрової взаємодії. У процесі виконання роботи використано комплекс методів, що забезпечують досягнення поставленої мети. Застосовано методи аналізу для вивчення предметної області обробки табличних даних та сучасних інформаційних систем. Використано методи моделювання інформаційних процесів під час побудови [UML](#)-діаграм та проектування структури системи. Під час реалізації програмної частини застосовано методи алгоритмізації та програмування, що дозволило ефективно організувати обробку та фільтрацію даних. Для оцінки працездатності системи використано методи тестування, зокрема перевірку функціональності та аналіз результатів роботи. Окрім цього, використано підхід комплексного аналізу із застосуванням комп'ютерних технологій, що дозволило об'єднати теоретичні положення та практичну реалізацію в межах єдиної інформаційної системи.

РОЗДІЛ 1. АНАЛІЗ ТЕХНОЛОГІЙ ОБРОБКИ ТА НАДАННЯ ТАБЛИЧНИХ ДАНИХ В ІНФОРМАЦІЙНИХ СИСТЕМАХ

1.1. Особливості роботи зі структурованими табличними даними та форматом [Excel](#)

Стрімкий розвиток цифрових технологій спричинив суттєве зростання обсягів інформації, що зберігається та обробляється у структурованому вигляді. У багатьох організаціях значна частина даних представлена у вигляді електронних таблиць, які використовуються для ведення обліку, формування звітності, зберігання персональних відомостей та аналізу різноманітних показників. Серед найбільш поширених форматів зберігання табличної інформації особливе місце займає [Excel](#), що поєднує простоту використання, гнучкість структури та широкі можливості для подальшої обробки даних. Разом із цим змінюються підходи до доступу користувачів до інформації. Традиційні способи взаємодії з інформаційними системами через складні програмні інтерфейси або спеціалізовані

програмні продукти поступово доповнюються більш зручними та інтерактивними рішеннями. Одним із таких підходів є використання месенджерів як платформи для надання інформаційних сервісів. [Telegram](#), який став одним із найпопулярніших засобів цифрової комунікації, надає широкі можливості для створення автоматизованих систем обробки запитів користувачів за допомогою спеціальних програмних агентів - ботів. Поєднання технологій обробки табличних даних із можливостями [Telegram](#)-ботів відкриває нові підходи до побудови інформаційних систем, здатних швидко надавати персоналізовану інформацію у відповідь на запити користувачів. Використання мови програмування [Python](#) та спеціалізованих бібліотек дозволяє ефективно реалізувати механізми зчитування, фільтрації та аналізу даних, що зберігаються у файлах формату [Excel](#), а інтеграція з [Telegram](#) забезпечує зручний інтерфейс доступу до цих даних. У сучасних інформаційних системах значна частина даних зберігається у структурованому вигляді. Це пов'язано з необхідністю впорядкування великого обсягу інформації, забезпечення швидкого доступу до неї та можливості подальшої автоматизованої обробки. Структуровані дані характеризуються наявністю чітко визначеної схеми організації, де кожен елемент має певне місце та логічний зв'язок з іншими елементами.

Найпоширенішою формою представлення таких даних є таблична структура, у якій інформація розміщується у вигляді рядків і стовпців [19]. Подібна модель організації даних широко використовується у різних сферах діяльності, починаючи від бухгалтерського обліку та статистичного аналізу і завершуючи освітніми, адміністративними та інформаційними сервісами. Таблична структура даних дозволяє ефективно організувати інформацію за допомогою чітко визначених атрибутів. Кожен рядок таблиці зазвичай представляє окремий запис або об'єкт, тоді як стовпці містять певні характеристики цього об'єкта. Такий підхід значно спрощує процеси сортування, фільтрації та аналізу інформації. Завдяки уніфікованій структурі дані можуть легко піддаватися автоматизованій обробці за допомогою програмних інструментів, що є важливою умовою для побудови сучасних інформаційних систем. Одним із найбільш поширених інструментів для роботи зі структурованими табличними даними є формат [Excel](#), який входить до складу програмного пакета [Microsoft Office](#) [4]. Електронні таблиці [Excel](#) отримали широке розповсюдження завдяки поєднанню простоти використання та значного функціонального потенціалу. Користувачі можуть створювати таблиці різної складності, виконувати математичні обчислення, використовувати формули, будувати графіки та проводити первинний аналіз даних. У результаті [Excel](#) став стандартним інструментом для роботи з табличною інформацією як у професійному середовищі, так і у повсякденній діяльності [9,18]. Файли [Excel](#) зазвичай мають розширення [.xlsx](#) або [.xls](#). У структурі такого файлу можуть міститися декілька робочих аркушів, кожен із яких представляє окрему таблицю. На кожному аркуші інформація організована у вигляді сітки, що складається з комірок, які мають власні координати, сформовані на основі номера рядка та позначення стовпця. Така структура забезпечує чітку ідентифікацію кожного елемента таблиці та дозволяє легко виконувати операції з даними. Однією з важливих переваг використання [Excel](#) у контексті інформаційних систем є можливість швидкого створення та редагування таблиць без необхідності спеціалізованих знань у сфері програмування або адміністрування баз даних. Для багатьох організацій електронні таблиці стають простим і доступним способом організації внутрішніх інформаційних ресурсів. Дані можуть зберігатися у вигляді списків клієнтів, реєстрів платежів, облікових журналів, результатів навчання або інших структурованих відомостей. Крім того, формат [Excel](#) добре підходить для інтеграції з різними програмними засобами. Сучасні мови програмування надають можливість безпосередньої роботи з електронними таблицями, що дозволяє автоматизувати процеси зчитування, обробки та аналізу інформації, а також спростити розробку, не використовуючи великі бази даних. Це особливо важливо у випадках, коли таблиці використовуються як джерело даних для інформаційних систем, що виконують автоматичну обробку запитів користувачів або формування відповідей на основі наявної інформації. Водночас робота з табличними даними має певні особливості, які необхідно враховувати при розробці інформаційних систем. Однією з таких особливостей є необхідність забезпечення коректної структури таблиці. Для ефективної автоматизованої обробки даних важливо, щоб кожен стовпець мав чітко визначене призначення, а всі записи відповідали єдиному формату. Наявність пропущених значень, змішаних типів даних або некоректних записів може ускладнювати процес обробки інформації та призводити до помилок під час виконання алгоритмів фільтрації або пошуку. Ще одним важливим аспектом є уніфікація назв стовпців та організація заголовків таблиці. У

більшості випадків перший рядок електронної таблиці використовується для зберігання назв полів, які визначають зміст відповідних стовпців. Такий підхід дозволяє програмним засобам правильно інтерпретувати структуру таблиці та використовувати назви полів для доступу до конкретних елементів даних. Важливо, щоб назви точно відповідали одна одній, адже тоді через код буде неможливо отримати інформацію зі стовпця або рядка таблиці. Окрему роль у роботі зі структурованими таблицями відіграє можливість виконання операцій фільтрації. Фільтрація дозволяє відбирати лише ті записи, які відповідають заданим умовам [10]. Наприклад, у таблиці з інформацією про користувачів може здійснюватися пошук запису за ідентифікаційним номером або іншим унікальним параметром. Подібні операції часто використовуються у системах автоматизованого доступу до персоналізованих даних, де користувач вводить певний параметр, а система знаходить відповідний запис у таблиці та надає необхідну інформацію. Важливим фактором також є можливість масштабування табличних даних. [Excel](#) дозволяє працювати з досить великими обсягами інформації, що робить цей формат придатним для багатьох практичних завдань [10]. Хоча у великих корпоративних системах зазвичай застосовуються спеціалізовані системи керування базами даних, електронні таблиці залишаються ефективним рішенням для невеликих або локальних інформаційних систем, де простота використання має вирішальне значення. У результаті електронні таблиці [Excel](#) займають важливе місце серед інструментів організації структурованих даних. Їхня популярність пояснюється поєднанням доступності, гнучкості та широких можливостей для обробки інформації. Використання такого формату як джерела даних дозволяє створювати інформаційні системи, що забезпечують швидкий доступ до структурованих відомостей, а також підтримують автоматизовану обробку запитів користувачів на основі наявної табличної інформації.

1.2. Інструменти [Python](#) для обробки табличних даних

Сучасні інформаційні системи дедалі частіше потребують інструментів, здатних швидко обробляти значні обсяги структурованих даних [12]. У багатьох випадках такі дані зберігаються у вигляді таблиць, що містять записи про користувачів, фінансові показники, статистичну інформацію або інші впорядковані відомості. Для автоматизації процесів роботи з подібними даними активно використовується мова програмування [Python](#), яка завдяки простому синтаксису, великій кількості бібліотек та широким можливостям інтеграції стала одним із найпопулярніших інструментів у сфері розробки інформаційних систем [1, 2]. [Python](#) надає зручні механізми для роботи з файлами різних форматів, включаючи текстові документи, електронні таблиці, бази даних та веб-ресурси. Особливу роль при цьому відіграють спеціалізовані бібліотеки, які значно спрощують обробку структурованої інформації. За їх допомогою можна автоматично зчитувати дані з таблиць, виконувати фільтрацію записів, сортувати інформацію, обчислювати статистичні показники або формувати нові структури даних. Завдяки цьому [Python](#) широко застосовується у створенні інформаційних систем, де необхідно забезпечити швидке отримання потрібної інформації на основі запитів користувачів [9]. Однією з найбільш відомих бібліотек для роботи з табличними даними у [Python](#) є

” Знайдено посилань: 0%

id: 4

«[pandas](#)» [13, 15]

. Цей інструмент забезпечує ефективну обробку структурованих наборів даних та підтримує різні формати файлів, включаючи [CSV](#), [Excel](#), [JSON](#) та інші. Основним елементом структури даних у [pandas](#) є об'єкт

” Знайдено посилань: 0%

id: 5

«[DataFrame](#)»,

який за своєю логікою нагадує електронну таблицю.

” Знайдено посилань: 0%

id: 6

«[DataFrame](#)»

складається з рядків і стовпців, що дозволяє виконувати різноманітні операції над даними, включаючи пошук, фільтрацію, об'єднання таблиць та статистичний аналіз. Важливою перевагою

” Знайдено посилань: 0%

id: 7

«[pandas](#)»

є можливість швидкого зчитування великих масивів даних. Наприклад, електронна

таблиця [Excel](#) може бути завантажена у вигляді

» Знайдено посилань: 0%

id: 8

«[DataFrame](#)»

лише за допомогою однієї функції [15]. Після цього всі записи таблиці стають доступними для програмної обробки. Такий підхід значно спрощує створення інформаційних систем, де таблиця використовується як джерело даних для подальшого аналізу або формування відповідей користувачам. Окрім

» Знайдено посилань: 0%

id: 9

«[pandas](#)»,

у [Python](#) існує низка інших бібліотек, що дозволяють працювати з табличними файлами. Наприклад, бібліотека

» Знайдено посилань: 0%

id: 10

«[openpyxl](#)»

використовується для читання та редагування [Excel](#)-файлів формату

» Знайдено посилань: 0%

id: 11

«[.xlsx](#)» [18]

. Вона надає можливість створювати нові таблиці, змінювати значення комірок, додавати формули або змінювати структуру документа. Іншим інструментом є бібліотека

» Знайдено посилань: 0%

id: 12

«[xlrd](#)»,

яка застосовується для читання файлів старіших форматів [Excel](#). Також для роботи з даними можуть використовуватися бібліотеки

» Знайдено посилань: 0%

id: 13

«[NumPy](#), [csv](#)»

та інші програмні модулі, що входять до стандартного набору інструментів [Python](#). Вибір конкретної бібліотеки залежить від поставлених завдань та характеру даних, з якими працює інформаційна система. У багатьох випадках використовується поєднання кількох інструментів, де одна бібліотека відповідає за зчитування файлу, а інша – за подальшу обробку інформації. Такий підхід дозволяє підвищити гнучкість системи та забезпечити більш ефективне виконання операцій над даними. Для наочного розуміння особливостей різних бібліотек доцільно розглянути їх основні характеристики та сфери використання у таблиці 1.1. Таблиця 1.1 Види деяких бібліотек, їх характеристики та сфери використання

Бібліотека Основне призначення Особливості використання [pandas](#) Обробка та аналіз структурованих даних Підтримує [DataFrame](#), забезпечує швидку фільтрацію, сортування та обробку великих таблиць [openpyxl](#) Робота з [Excel](#)-файлами формату [XLSX](#) Дозволяє читати, створювати та редагувати електронні таблиці [xlrd](#) Читання файлів [Excel](#) Використовується для роботи зі старими форматами таблиць [NumPy](#) Математична обробка даних Використовується для обчислень та роботи з масивами [csv](#) Робота з текстовими таблицями Забезпечує обробку даних у форматі [CSV](#) Наведена таблиця демонструє, що [Python](#) має широкий набір інструментів для роботи зі структурованими даними. Кожна бібліотека виконує певну функцію, але найбільш універсальним інструментом у більшості випадків залишається

» Знайдено посилань: 0%

id: 14

«[pandas](#)».

Саме ця бібліотека поєднує можливості зчитування файлів, обробки даних та виконання різноманітних операцій над таблицями, що робить її одним із ключових елементів сучасних інформаційних систем. Окрему увагу варто приділити можливостям фільтрації даних, які реалізовані у

» Знайдено посилань: 0%

id: 15

«[pandas](#)».

Використовуючи прості програмні конструкції, можна виконувати пошук записів за конкретними параметрами. Наприклад, система може отримати ідентифікаційний номер користувача та знайти відповідний рядок у таблиці, після чого сформулювати відповідь на

основі знайдених значень. Подібний механізм часто застосовується у сервісах автоматизованого доступу до інформації, де користувачі отримують персоналізовані дані на основі власного запиту. Ще однією важливою особливістю використання [Python](#) у роботі з табличними даними є можливість інтеграції з іншими програмними компонентами інформаційної системи. Оброблені дані можуть передаватися до веб-сервісів, зберігатися у базах даних або використовуватися для формування відповідей у системах автоматизованого спілкування. Це дозволяє створювати комплексні інформаційні системи, де [Python](#) виступає центральним елементом обробки даних. У результаті використання [Python](#) та його спеціалізованих бібліотек значно спрощується процес роботи зі структурованими табличними даними. Інструменти цієї мови програмування дозволяють автоматизувати операції зчитування, фільтрації, аналізу та передачі інформації, що є важливою складовою сучасних інформаційних систем [14]. Це створює сприятливі умови для реалізації сервісів, які забезпечують швидкий доступ користувачів до потрібної інформації на основі даних, що зберігаються у табличних файлах.

1.3. Використання [Telegram](#)-ботів як інтерфейсу доступу до інформаційних систем

У сучасних умовах розвитку інформаційних технологій особливого значення набувають засоби взаємодії користувача з інформаційними системами. Зручність доступу до даних, швидкість отримання відповіді та простота використання інтерфейсу безпосередньо впливають на ефективність функціонування системи. Традиційні підходи, що передбачають використання окремих програм або веб-інтерфейсів, поступово доповнюються новими рішеннями, серед яких важливе місце займають месенджери. Одним із найбільш перспективних інструментів у цьому напрямі є [Telegram](#)-боти. Вони являють собою спеціалізовані програмні компоненти, що функціонують у межах платформи [Telegram](#) та забезпечують автоматизовану взаємодію з користувачами у форматі діалогу. Такий підхід дозволяє організувати доступ до інформаційної системи без необхідності встановлення додаткового програмного забезпечення або використання складних інтерфейсів. Користувач отримує можливість взаємодіяти з системою через звичайні повідомлення, що значно підвищує зручність використання. [Telegram](#)-боти можуть виконувати широкий спектр функцій, починаючи від надання довідкової інформації і завершуючи обробкою складних запитів із використанням зовнішніх джерел даних. У контексті інформаційних систем, що працюють із табличними даними, боти можуть виступати як інтерфейс доступу до інформації, збереженої у файлах або базах даних. Користувач вводить запит, після чого система обробляє його та повертає відповідь у зручному текстовому форматі [2]. Однією з ключових переваг використання [Telegram](#)-ботів є їх інтерактивність. Взаємодія з користувачем будується у вигляді послідовного діалогу, що дозволяє реалізувати логіку уточнення даних, перевірки введеної інформації та формування відповіді на основі декількох кроків. Наприклад, система може спочатку отримати ідентифікаційний параметр, після чого уточнити додаткові дані та лише потім надати кінцевий результат. Такий підхід забезпечує більш точну обробку запитів та зменшує ймовірність помилок. Технічно [Telegram](#)-боти функціонують на основі спеціального програмного інтерфейсу [16]. Цей інтерфейс надає можливість отримувати повідомлення від користувачів, обробляти їх та надсилати відповіді у вигляді тексту, кнопок або інших елементів інтерфейсу. Для реалізації ботів у середовищі [Python](#) широко використовується бібліотека

” Знайдено посилань: 0%

id: 16

«[telegram](#)»,

яка забезпечує асинхронну обробку подій та дозволяє створювати ефективні інформаційні системи з підтримкою одночасної роботи кількох користувачів [2,8]. Особливу роль у підвищенні зручності використання [Telegram](#)-ботів відіграють елементи інтерфейсу, такі як клавіатури з кнопками (рис. 1.1). Використання готових варіантів відповіді дозволяє зменшити кількість помилок під час введення даних, а також пришвидшує взаємодію користувача із системою. Наприклад, кнопки можуть використовуватися для підтвердження дій, вибору варіантів або переходу між різними етапами обробки запиту. Рис. 1.1. Клавіатура з кнопками З точки зору архітектури інформаційної системи [Telegram](#)-бот виступає проміжною ланкою між користувачем та джерелом даних. Він приймає запити, передає їх на обробку відповідному модулю системи, а потім повертає результат у вигляді повідомлення. Такий підхід дозволяє розділити логіку взаємодії та логіку обробки даних, що підвищує гнучкість і масштабованість системи. Для більш детального розуміння можливостей [Telegram](#)-ботів доцільно розглянути їх основні функціональні характеристики у таблиці 1.2. Таблиця 1.2 Основні можливості [Telegram](#)-ботів в інформаційних системах

Можливість Опис Обробка повідомлень Прийом текстових запитів користувача та їх подальша обробка Інтерактивна взаємодія Реалізація діалогового режиму спілкування з користувачем Використання кнопок Спрощення введення даних за допомогою готових варіантів Інтеграція з джерелами даних Отримання інформації з файлів, баз даних або веб-сервісів Асинхронна обробка Підтримка одночасної роботи з декількома користувачами Наведені можливості свідчать про те, що [Telegram](#)-боти є ефективним інструментом для побудови інтерфейсів доступу до інформаційних систем. Вони дозволяють організувати зручну взаємодію з користувачем, забезпечують швидке отримання необхідної інформації та можуть бути інтегровані з різними джерелами даних [16]. Використання [Telegram](#)-ботів особливо доцільне у випадках, коли необхідно забезпечити оперативний доступ до персоналізованої інформації. Наприклад, користувач може отримати дані про власні показники, результати обліку або інші відомості, просто надіславши відповідний запит у месенджері. Це дозволяє значно спростити процес отримання інформації та зробити його максимально доступним для широкого кола користувачів. У результаті [Telegram](#)-боти виступають важливим елементом сучасних інформаційних систем, забезпечуючи ефективний та зручний інтерфейс взаємодії. Їх використання у поєднанні з інструментами обробки табличних даних відкриває широкі можливості для створення автоматизованих сервісів, здатних швидко реагувати на запити користувачів і надавати актуальну інформацію у зручному форматі [10]. Висновки до розділу У межах першого розділу розглянуто ключові аспекти, пов'язані з обробкою та використанням структурованих табличних даних в інформаційних системах. Показано, що таблична форма організації інформації є однією з найбільш зручних та поширених завдяки своїй наочності, простоті структурування та можливості швидкого виконання операцій аналізу, сортування й фільтрації. Формат [Excel](#) визначено як ефективний інструмент зберігання даних, який поєднує доступність для користувачів і достатній функціонал для подальшої автоматизованої обробки. Проаналізовано інструментальні засоби мови програмування [Python](#), призначені для роботи з табличними даними. Особливу увагу приділено бібліотеці

» Знайдено посилань: 0%

id: 17

«[pandas](#)»,

яка забезпечує зручне представлення даних у вигляді структур типу

» Знайдено посилань: 0%

id: 18

«[DataFrame](#)»

та дозволяє виконувати широкий спектр операцій над ними [14]. Розглянуто також інші бібліотеки, що доповнюють функціональність обробки [Excel](#)-файлів і дозволяють адаптувати систему до різних форматів і сценаріїв використання. Окремо висвітлено роль [Telegram](#)-ботів як інтерфейсу доступу до інформаційних систем [15]. Показано, що використання месенджера як платформи взаємодії значно спрощує процес отримання інформації для користувача, забезпечує швидкий обмін даними та дозволяє реалізувати логіку покрокового діалогу. Інтеграція таких ботів із модулями обробки табличних даних формує основу для створення ефективних інформаційних систем із високим рівнем доступності та зручності використання. У результаті проведеного аналізу сформовано теоретичну базу, необхідну для подальшого проектування інформаційної системи, що поєднує обробку даних з [Excel](#)-файлів та взаємодію з користувачем через [Telegram](#)-бота. Отримані положення створюють підґрунтя для переходу до етапу моделювання та розробки архітектури системи. РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ОБРОБКИ ДАНИХ НА ОСНОВІ [TELEGRAM](#)-БОТА 2.1. Формування функціональних вимог до інформаційної системи Після аналізу сучасних підходів до обробки табличних даних та засобів взаємодії з користувачем постає необхідність переходу до етапу проектування інформаційної системи. Саме на цьому етапі формується загальна структура майбутньої системи, визначаються її функціональні можливості, а також встановлюються принципи взаємодії між окремими компонентами. Якість проектування безпосередньо впливає на ефективність, стабільність і зручність подальшого використання системи. Особливістю даної інформаційної системи є поєднання двох ключових складових: обробки структурованих даних, що зберігаються у форматі [Excel](#), та організації взаємодії з користувачем через [Telegram](#)-бота. Такий підхід дозволяє створити гнучку та доступну систему, у якій складні внутрішні процеси обробки даних залишаються прихованими від користувача, а взаємодія відбувається у звичному та зрозумілому середовищі месенджера. Етап формування функціональних вимог відіграє визначальну роль у процесі проектування

інформаційної системи, оскільки саме на цьому рівні задається логіка її роботи, визначається перелік можливостей та окреслюються межі майбутньої реалізації [20]. Чітко сформульовані вимоги дозволяють уникнути неоднозначностей, забезпечують узгодженість між окремими компонентами системи та створюють основу для подальшого моделювання й реалізації. У контексті розробки інформаційної системи на основі [Telegram](#)-бота особливого значення набуває правильне визначення сценаріїв взаємодії з користувачем, а також механізмів обробки табличних даних. Функціональні вимоги відображають ті дії, які система повинна виконувати у відповідь на запити користувачів або внутрішні події [11,12]. Вони визначають поведінку системи та описують її основні можливості. У даному випадку інформаційна система орієнтована на обробку структурованих даних з [Excel](#)-файлу та надання персоналізованої інформації користувачам через інтерфейс [Telegram](#)-бота [7]. Це означає, що ключовим завданням є забезпечення коректного прийому вхідних даних, їх обробки та формування відповіді у зрозумілому форматі. Однією з базових функцій системи є ініціалізація взаємодії з користувачем. Після запуску [Telegram](#)-бота користувач повинен отримати початкове повідомлення, яке пояснює принцип роботи системи та пропонує виконати певну дію. Це дозволяє сформувати зрозумілу точку входу та забезпечити зручність подальшого використання. Важливим елементом при цьому є використання інтерактивних кнопок (рис. 1.1), які спрощують процес взаємодії та зменшують ймовірність помилок при введенні даних. Наступним важливим аспектом є обробка введених користувачем параметрів. У межах інформаційної системи передбачається використання унікального ідентифікатора, за допомогою якого здійснюється пошук відповідного запису у таблиці. Система повинна перевіряти коректність введених даних, визначати їх формат та виконувати пошук у структурованому масиві інформації. У разі відсутності відповідного запису користувач має отримати повідомлення про помилку з можливістю повторного введення даних. Окрему увагу приділено механізму підтвердження особи користувача. Після знаходження відповідного запису система повинна відобразити частину персональних даних і запропонувати підтвердити їх правильність. Це дозволяє уникнути ситуацій, коли інформація надається сторонній особі. У разі підтвердження система переходить до наступного етапу - надання результату, а у випадку відмови пропонує повторити введення параметрів. Важливою функціональною складовою є формування відповіді на запит користувача. Отримані з [Excel](#)-файлу дані повинні бути представлені у зрозумілому та компактному вигляді. Це передбачає використання текстових повідомлень, які містять необхідну інформацію без перевантаження зайвими деталями. Після завершення обробки запиту система повинна повернути користувача до початкового стану, що дозволяє виконати новий запит без повторного запуску бота. Не менш важливим є забезпечення безперервності роботи системи. Обробка помилок, некоректних введень або відсутності даних повинна виконуватися без припинення функціонування бота. Система повинна залишатися стабільною незалежно від дій користувача, що є важливою умовою для її практичного застосування. Для систематизації функціональних можливостей доцільно представити їх у вигляді узагальненої таблиці, що відображає ключові функції та їх призначення. Таблиця 2.1 Функціональні вимоги інформаційної системи № Функціональна вимога Опис 1 Ініціалізація взаємодії Відображення стартового повідомлення та меню взаємодії 2 Прийом даних від користувача Отримання введеного ідентифікатора (ІНН або іншого параметра) 3 Перевірка коректності введення Валідація формату даних та обробка помилок 4 Пошук даних у таблиці Фільтрація записів у [Excel](#)-файлі за заданим параметром 5 Підтвердження особи Запит підтвердження знайдених персональних даних 6 Формування відповіді Виведення результату обробки у вигляді текстового повідомлення 7 Обробка помилок Реагування на некоректні запити або відсутність даних 8 Повторна взаємодія Повернення користувача до початкового стану системи Наведена таблиця демонструє, що інформаційна система має чітко визначений набір функцій, кожна з яких відповідає за певний етап обробки запиту користувача [1]. Така структуризація дозволяє спростити подальше моделювання системи та забезпечує логічну послідовність її роботи. Після визначення функціональних вимог стає можливим перехід до етапу моделювання структури системи. Чітке розуміння функцій дозволяє відобразити їх у вигляді [UML](#)-діаграм, що забезпечує наочне представлення логіки взаємодії між компонентами. Це, у свою чергу, створює основу для подальшої реалізації інформаційної системи та забезпечує узгодженість між її окремими частинами. 2.2. Моделювання структури та архітектури системи засобами [UML](#) Етап моделювання інформаційної системи дозволяє сформувати цілісне уявлення про її структуру, логіку функціонування та взаємодію між основними

компонентами ще до безпосередньої реалізації. Використання [UML](#)-діаграм забезпечує наочність представлення системи, спрощує розуміння її внутрішніх процесів та дозволяє уникнути помилок на етапі проєктування [20]. У межах даного підпункту виконано моделювання інформаційної системи обробки табличних даних із використанням [Telegram](#)-бота як інтерфейсу взаємодії з користувачем. Першим кроком у моделюванні є побудова діаграми варіантів використання ([Use Case Diagram](#)), яка відображає основні сценарії взаємодії користувача з інформаційною системою (рис. 2.1) [21]. У даній системі ключовим актором виступає користувач [Telegram](#), який взаємодіє з ботом через надсилання повідомлень. Основні варіанти використання включають ініціалізацію взаємодії, введення ідентифікаційного параметра, підтвердження особи та отримання результату обробки даних. Така діаграма дозволяє чітко визначити межі системи та перелік функцій, доступних користувачу. Рис. 2.1. Діаграма варіантів використання Як видно з діаграми варіантів використання, система має лінійну та зрозумілу логіку взаємодії. Користувач ініціює роботу бота, після чого послідовно проходить етапи введення даних і отримання результату. Наявність етапу підтвердження дозволяє підвищити достовірність обробки запитів і запобігти некоректному наданню інформації. Такий підхід забезпечує простоту використання та мінімізує кількість помилок під час взаємодії. Наступним етапом є побудова діаграми діяльності ([Activity Diagram](#)), яка відображає внутрішню логіку роботи системи та послідовність виконання операцій (рис. 2.2). Ця діаграма дозволяє простежити повний цикл обробки запиту - від моменту отримання повідомлення до формування відповіді користувачу. У рамках реалізованої інформаційної системи передбачено перевірку введених даних, пошук відповідного запису в таблиці, обробку результатів та формування відповіді. Рис. 2.2. Діаграма діяльності Діаграма діяльності демонструє, що процес обробки запиту має чітко визначену послідовність дій із можливими розгалуженнями [11]. Наприклад, у випадку некоректного введення або відсутності даних у таблиці система повертає повідомлення про помилку та пропонує повторити введення. У разі успішного знаходження запису відбувається перехід до етапу підтвердження, після чого формується відповідь. Така логіка дозволяє забезпечити гнучкість системи та її стійкість до помилкових дій користувача. Завершальним етапом моделювання є побудова діаграми компонентів ([Component Diagram](#)), яка відображає архітектуру інформаційної системи та взаємозв'язки між її складовими (рис. 2.3) [20, 21]. У даній системі можна виділити декілька ключових компонентів: [Telegram](#)-клієнт користувача, [Telegram Bot API](#), програмний модуль на [Python](#), бібліотеки обробки даних та файл [Excel](#) як джерело інформації [5]. Кожен із цих компонентів виконує свою функцію та взаємодіє з іншими елементами системи. Рис. 2.3. Діаграма компонентів З аналізу діаграми компонентів видно, що система має модульну структуру. [Telegram](#) виступає зовнішнім інтерфейсом, через який користувач надсилає запити. Далі запити передаються до програмного модуля, реалізованого мовою [Python](#) із використанням бібліотеки [aiogram](#). Цей модуль виконує обробку запитів, звертається до [Excel](#)-файлу через бібліотеку [pandas](#) та формує відповідь, яка повертається користувачу через [Telegram](#) [2]. Така архітектура дозволяє розділити функції між компонентами та забезпечує зручність масштабування системи. У результаті виконаного моделювання сформовано повне уявлення про функціонування інформаційної системи. Діаграми варіантів використання, діяльності та компонентів відображають ключові аспекти її роботи, дозволяють виявити взаємозв'язки між елементами та створюють основу для переходу до етапу реалізації.

2.3. Проєктування алгоритмів обробки та фільтрації даних

Після визначення функціональних вимог та моделювання структури інформаційної системи наступним етапом є проєктування алгоритмів обробки та фільтрації даних. Саме алгоритмічна складова визначає, яким чином система реагує на дії користувача, як виконується пошук інформації у табличному джерелі та у якому вигляді формується відповідь. Чітко продумана логіка обробки забезпечує коректність результатів, стабільність функціонування та зручність взаємодії. Основу роботи інформаційної системи становить обробка запиту користувача, що надходить через [Telegram](#)-бот. У процесі взаємодії передбачено послідовне виконання декількох етапів, кожен із яких відповідає за окрему частину загального алгоритму. Важливою особливістю є те, що алгоритм має враховувати як стандартні сценарії, так і можливі відхилення, пов'язані з некоректним введенням або відсутністю даних. Першим етапом є ініціалізація процесу обробки. Після отримання повідомлення від користувача система повинна визначити його зміст та встановити, до якого типу запитів воно належить. У випадку введення ідентифікаційного параметра виконується перевірка формату даних. Це дозволяє відсіяти некоректні значення ще до початку основної обробки та запобігти помилкам на наступних етапах.

Наступним кроком є зчитування даних із табличного джерела. Інформаційна система використовує файл формату [Excel](#), який містить структуровані записи. Під час ініціалізації або при необхідності оновлення даних виконується завантаження таблиці у внутрішню структуру, що дозволяє здійснювати подальші операції обробки. Для цього використовується відповідний інструментарій, який забезпечує швидкий доступ до рядків і стовпців таблиці. Ключовим етапом алгоритму є фільтрація даних [10]. Вона полягає у відборі тих записів, які відповідають введеному користувачем параметру. У даній інформаційній системі фільтрація виконується за унікальним ідентифікатором, що дозволяє знайти конкретний запис серед множини інших [10]. Процес фільтрації передбачає порівняння значення, введеного користувачем, із відповідними значеннями у таблиці. У разі збігу система отримує доступ до повного набору даних, пов'язаних із цим записом. Особливу увагу приділено обробці результатів фільтрації. Якщо запис знайдено, система переходить до етапу уточнення даних. Відображається частина інформації, яка дозволяє користувачу підтвердити свою ідентифікацію. Такий підхід підвищує рівень достовірності обробки та зменшує ризик надання інформації сторонній особі. У разі підтвердження система переходить до формування відповіді. Формування відповіді є завершальним етапом алгоритму. На основі отриманих даних система генерує повідомлення, яке містить необхідну інформацію у зрозумілому форматі. При цьому важливо забезпечити лаконічність та інформативність відповіді, щоб користувач міг швидко отримати потрібні відомості без зайвих деталей. Після цього система повертається до початкового стану, що дозволяє обробляти нові запити. У випадку, якщо під час фільтрації не знайдено відповідного запису, алгоритм передбачає альтернативний сценарій. Користувач отримує повідомлення про відсутність даних із пропозицією повторити введення параметра. Аналогічно обробляються ситуації з некоректним введенням або відмовою від підтвердження. Це забезпечує гнучкість алгоритму та дозволяє підтримувати безперервність роботи системи. Окремо варто відзначити роль тимчасового збереження даних у процесі обробки. Інформація, отримана на одному етапі, може використовуватися на наступних кроках алгоритму. Це дозволяє реалізувати багатокрокову логіку взаємодії, де кожен етап залежить від попереднього. Такий підхід особливо важливий для систем, що працюють у режимі діалогу з користувачем. Для наочного представлення логіки роботи інформаційної системи доцільно використати блок-схему алгоритму обробки та фільтрації даних, яка зображена на рисунку 2.4 та відображає послідовність основних етапів взаємодії з користувачем та обробки запиту. Рис. 2.4. Алгоритм обробки та фільтрації даних На представленій схемі відображено повний цикл роботи інформаційної системи - від моменту запуску [Telegram](#)-бота до отримання користувачем результату. Початковим етапом є введення ідентифікаційного параметра, після чого виконується перевірка наявності відповідного запису у табличному джерелі даних. У разі успішного знаходження відображаються персональні дані для підтвердження, що дозволяє забезпечити коректність подальшої обробки. Подальший етап передбачає отримання значення заборгованості та формування відповіді користувачу. У випадках, коли запис не знайдено або введені дані не підтверджено, алгоритм передбачає повернення до етапу повторного введення. Це забезпечує гнучкість роботи системи та її стійкість до помилкових дій користувача. Представлення алгоритму у вигляді блок-схеми дозволяє чітко простежити логіку функціонування системи та взаємозв'язок між окремими етапами обробки даних. Важливою характеристикою розробленого алгоритму є його послідовність та передбачуваність. Кожен етап має чітко визначене місце у загальній структурі, що дозволяє легко відслідковувати процес обробки запиту. Крім того, алгоритм враховує можливі варіанти розвитку подій, що забезпечує стійкість системи до помилкових дій користувача. У результаті проєктування алгоритмів обробки та фільтрації даних сформовано логічно завершену модель функціонування інформаційної системи. Вона охоплює всі основні етапи взаємодії з користувачем, забезпечує коректну обробку табличних даних та створює основу для подальшої реалізації системи з використанням обраних технологій. Висновки до розділу У межах другого розділу розкрито процес проєктування інформаційної системи обробки табличних даних із використанням [Telegram](#)-бота як інтерфейсу взаємодії. Основну увагу зосереджено на формуванні функціональних вимог, моделюванні структури системи та розробці алгоритмів її роботи. Це дозволило сформувати цілісне уявлення про принципи функціонування системи ще до етапу реалізації. У процесі визначення функціональних вимог встановлено ключові можливості системи, зокрема прийом і обробку запитів користувачів, перевірку введених даних, пошук інформації у табличному джерелі та формування відповіді. Чітка структуризація функцій

забезпечила логічну послідовність роботи системи та створила основу для подальшого моделювання. Побудова [UML](#)-діаграм дозволила наочно відобразити як зовнішню взаємодію з користувачем, так і внутрішню логіку роботи системи. Діаграма варіантів використання визначила основні сценарії взаємодії, діаграма діяльності показала послідовність виконання операцій, а діаграма компонентів відобразила архітектуру системи та взаємозв'язок її складових [11, 5]. Додаткове використання діаграми послідовності дало можливість простежити обмін повідомленнями між компонентами у процесі обробки запиту. У рамках проектування алгоритмів обробки та фільтрації даних сформовано чітку логіку функціонування системи. Алгоритм передбачає послідовну обробку запиту користувача, перевірку коректності введених даних, пошук інформації у таблиці, підтвердження особи та формування результату. Передбачено також обробку альтернативних сценаріїв, що забезпечує стійкість системи до помилкових дій та підвищує надійність її роботи. У результаті виконаних дій сформовано повноцінну модель інформаційної системи, яка поєднує обробку табличних даних із зручним інтерфейсом взаємодії через [Telegram](#). Отримані результати створюють обґрунтовану основу для переходу до етапу програмної реалізації та подальшого тестування розробленої системи.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1. Реалізація модуля зчитування та обробки [Excel](#)-даних у середовищі [Python](#)

Після завершення етапу проектування постає практичне завдання реалізації інформаційної системи, у межах якого теоретичні напрацювання набувають конкретного програмного вигляду. Саме на цьому етапі перевіряється життєздатність обраних підходів, ефективність алгоритмів та коректність побудованої архітектури. Реалізація дозволяє оцінити, наскільки розроблена модель відповідає поставленим вимогам та чи забезпечує вона необхідний рівень функціональності. Особливістю даної інформаційної системи є поєднання обробки табличних даних із використанням [Python](#) та організації взаємодії через [Telegram](#)-бота. Такий підхід дозволяє створити зручний інструмент доступу до інформації, де користувач отримує результат у простому та зрозумілому форматі, не взаємодіючи безпосередньо зі складними внутрішніми процесами. Реалізація модуля обробки табличних даних є ключовим етапом створення інформаційної системи, оскільки саме цей компонент відповідає за отримання, аналіз і підготовку даних до подальшого використання. У межах розробленої системи джерелом інформації виступає [Excel](#)-файл, що містить структуровані записи. Відповідно, основне завдання полягає у забезпеченні коректного зчитування даних, їх обробки та підготовки до використання у взаємодії з [Telegram](#)-ботом. Для реалізації даного модуля використано мову програмування [Python](#), яка забезпечує широкий набір інструментів для роботи з табличними даними. Зокрема, застосовано бібліотеку

” Знайдено посилань: 0%

id: 19

«[pandas](#)»,

що дозволяє ефективно працювати зі структурованими наборами даних та виконувати операції фільтрації, пошуку та аналізу. Завдяки цьому забезпечується висока швидкість обробки інформації та зручність реалізації алгоритмів. Першим етапом є підключення необхідних бібліотек та підготовка середовища виконання. На цьому етапі здійснюється імпорт модулів, що відповідають за обробку [Excel](#)-файлів та організацію роботи з даними. Відповідний фрагмент програмного коду наведено на рис. 3.1. Приклад структури [Excel](#)-файлу представлений в кінці роботи (див. додаток Б) Рис. 3.1. Підключення необхідних бібліотек Наступним кроком є зчитування даних із [Excel](#)-файлу. Для цього використовується функціональність бібліотеки

” Знайдено посилань: 0%

id: 20

«[pandas](#)»,

яка дозволяє завантажити таблицю у вигляді структури

” Знайдено посилань: 0%

id: 21

«[DataFrame](#)».

Для підвищення стабільності роботи системи процес завантаження [Excel](#)-файлу було реалізовано із використанням конструкції

” Знайдено посилань: 0%

id: 22

«[try/except](#)»,

що дозволяє виявляти помилки під час відкриття файлу або пошкодження структури

даних. Такий підхід забезпечує зручний доступ до даних та можливість виконання подальших операцій над ними. Процес зчитування файлу та ініціалізації структури даних представлено на рис. 3.2. Рис. 3.2. Завантаження даних Після завантаження даних виконується їх попередня обробка. Вона включає перевірку коректності структури таблиці, визначення назв стовпців та приведення даних до необхідного формату. Це дозволяє уникнути помилок під час подальшої роботи з інформацією. На цьому етапі також може виконуватися очищення даних від пропущених або некоректних значень. Ключовим етапом роботи модуля є реалізація механізму пошуку даних за заданим параметром. У даній системі таким параметром виступає ідентифікаційний номер, введений користувачем. Пошук здійснюється шляхом фільтрації

” Знайдено посилань: 0%

id: 23

«DataFrame»

за відповідним стовпцем [14]. У разі знаходження запису система отримує доступ до повного набору пов'язаних даних. Відповідний фрагмент коду, що реалізує пошук, представлено на рис. 3.3. Рис. 3.3. Пошук даних за заданим параметром У випадку, якщо запис не знайдено, передбачено обробку цієї ситуації. Система повинна коректно визначити відсутність результату та передати відповідну інформацію для подальшого використання у взаємодії з користувачем. Це дозволяє забезпечити стабільність роботи системи та уникнути помилок під час виконання запитів. Приклад реалізації перевірки результату пошуку наведено на рис. 3.3. Якщо ж вхідні дані введено невірно, наприклад, замість цифр – літери, тоді інформаційна система повинна видати правильну поправку. Для цього було створено фрагмент коду, який зображено на рисунку 3.4, що відповідає за вирішення цієї проблеми. Рис. 3.4. Обробник вхідної інформації Після успішного знаходження необхідного запису виконується отримання конкретних значень із таблиці. Зокрема, здійснюється вибір потрібних полів, які будуть використані для формування відповіді користувачу. А саме це персональні дані та значення заборгованості. Доступ до таких даних реалізується через відповідні методи роботи зі структурою

” Знайдено посилань: 0%

id: 24

«DataFrame».

Фрагмент коду, що демонструє отримання значень, представлено на рисунку 3.5 та рисунку 3.6. Рис. 3.5. Отримання персональної інформації Рис. 3.6. Отримання заборгованості Окрему увагу приділено оптимізації роботи з даними. Зчитування [Excel](#)-файлу може виконуватися один раз під час запуску системи, що дозволяє уникнути повторного доступу до файлу при кожному запиті [17]. Це значно підвищує продуктивність та зменшує затримки під час обробки запитів користувачів. Такий підхід є особливо важливим у випадках, коли система обробляє велику кількість звернень [6]. Важливою характеристикою реалізованого модуля є його простота та ефективність. Використання готових інструментів для роботи з табличними даними дозволило значно скоротити обсяг програмного коду та забезпечити високу читабельність реалізації. Крім того, структура коду дозволяє легко вносити зміни у разі оновлення формату даних або розширення функціональності системи [3]. У результаті реалізації модуля зчитування та обробки [Excel](#)-даних забезпечено коректну роботу з табличною інформацією, що є основою функціонування всієї інформаційної системи. Отримані дані можуть бути ефективно використані для формування відповідей користувачам та інтеграції з іншими компонентами системи, зокрема [Telegram](#)-ботом. 3.2. Розробка [Telegram](#)-бота та реалізація взаємодії з користувачем Розробка [Telegram](#)-бота є важливим етапом реалізації інформаційної системи, оскільки саме цей компонент забезпечує безпосередню взаємодію з користувачем. У межах створеної системи бот виконує роль інтерфейсу доступу до табличних даних, дозволяючи отримувати необхідну інформацію у зручному форматі без використання додаткових програмних засобів. Такий підхід значно спрощує процес роботи з системою та робить її доступною для широкого кола користувачів. Спочатку для створення [Telegram](#)-бота, потрібно зайти в сам месенджер [Telegram](#) та ввести у поле пошуку

” Знайдено посилань: 0%

id: 25

«@BotFather»

(рис. 3.7).

” Знайдено посилань: 0%

id: 26

«[BotFather](#)»

- це єдиний бот, який керує всіма. Його потрібно використовувати для створення нових облікових записів ботів та керування існуючими. Рис. 3.7. Пошук [BotFather](#) Далі потрібно ввести команду

» Знайдено посилань: 0%

id: 27

«[/start](#)»

і буде представлено список існуючих команд, наприклад рисунок 3.8. Використовуючи наступні дії, що зображені на рисунку 3.9, можливо отримати власного бота в [Telegram](#). Рис. 3.8. Приклад переліку існуючих команд Рис. 3.9. Дії для створення [Telegram](#)-бота Для реалізації [Telegram](#)-бота використано мову програмування [Python](#) та спеціалізовану бібліотеку

» Знайдено посилань: 0%

id: 28

«[aiogram](#)»,

яка забезпечує обробку повідомлень у асинхронному режимі [16]. Це дозволяє системі одночасно обслуговувати декілька запитів користувачів без втрати продуктивності. Крім того, використання цієї бібліотеки спрощує реалізацію логіки взаємодії та дозволяє структурувати програмний код відповідно до функціональних вимог системи. Першим етапом розробки є налаштування середовища та створення основи [Telegram](#)-бота. На цьому етапі виконується підключення необхідних модулів, ініціалізація об'єкта бота та налаштування обробників повідомлень. Також здійснюється інтеграція з [Telegram Bot API](#), що дозволяє отримувати повідомлення від користувачів та надсилати відповіді.

Інформацію про бота (токен) потрібно взяти у чаті з [Telegram](#)-ботом

» Знайдено посилань: 0%

id: 29

«[BotFather](#)»,

приклад повідомлення якого представлено на рисунку 3.10. Фрагмент програмного коду, що відповідає за ініціалізацію бота, представлено на рис. 3.11. Рис. 3.10. Отримання [Telegram Bot API](#) Рис. 3.11. Ініціалізація бота в коді Наступним кроком є реалізація обробника команди запуску. Після введення користувачем команди початку роботи система повинна сформувати привітальне повідомлення та відобразити основні можливості бота. Для підвищення зручності взаємодії використовується клавіатура з кнопками, яка дозволяє швидко обрати необхідну дію. Відповідний приклад реалізації наведено на рис. 3.12. Рис. 3.12. Обробник команди запуску Особливу увагу приділено організації діалогової взаємодії. Після вибору функції перевірки даних користувачеві пропонується ввести ідентифікаційний параметр. Отримане повідомлення передається до обробника, який виконує перевірку введених даних та передає їх до модуля обробки [Excel](#)-файлу. Такий підхід забезпечує чіткий розподіл відповідальності між компонентами системи та підвищує її гнучкість. Реалізацію обробника введення даних представлено на рис. 3.13. Рис. 3.13. Організації діалогової взаємодії Після отримання результату обробки система переходить до етапу підтвердження даних. Користувачеві відображається частина інформації, яка дозволяє перевірити правильність введених даних. Для цього також використовуються інтерактивні кнопки, що спрощують процес підтвердження. У випадку позитивної відповіді система переходить до формування кінцевого результату, у протилежному випадку пропонується повторне введення параметра. Фрагмент коду, що реалізує цей етап, наведено на рис. 3.14. Рис. 3.14. Фрагмент коду реалізації підтвердження даних Важливим елементом є формування відповіді користувачу. Інформація, отримана з табличного джерела, повинна бути представлена у зрозумілому та зручному вигляді. Це досягається шляхом формування текстового повідомлення, яке містить лише необхідні дані. Такий підхід дозволяє уникнути перевантаження інформацією та забезпечує швидке сприйняття результату. Приклад реалізації формування відповіді представлено на рис. 3.15. Рис. 3.15. Фрагмент коду для формування кінцевої відповіді Не менш важливою є обробка помилок та нестандартних ситуацій. У процесі роботи користувач може вводити некоректні дані або виконувати дії, які не передбачені основним сценарієм. У таких випадках система повинна коректно реагувати, повідомляючи про помилку та пропонуючи варіанти подальших дій. Це забезпечує стабільність функціонування бота та підвищує зручність його використання. Реалізацію обробки помилок наведено на кожному з попередніх рисунка та зазначено у додатку А. Варто

зазначити, що логіка роботи [Telegram](#)-бота безпосередньо пов'язана з алгоритмами, розробленими у попередньому розділі. Зокрема, послідовність дій, що реалізована у програмному коді, відповідає етапам, відображеним на діаграмі діяльності та блок-схемі алгоритму обробки даних. Це забезпечує узгодженість між теоретичною моделлю та практичною реалізацією системи. У результаті реалізації [Telegram](#)-бота створено ефективний інтерфейс доступу до інформаційної системи, який забезпечує зручну взаємодію з користувачем та дозволяє швидко отримувати необхідні дані. Інтеграція з модулем обробки [Excel](#)-файлів дозволила об'єднати всі компоненти системи в єдину структуру, що забезпечує її цілісність та функціональність [2]. 3.3. Розгортання та запуск інформаційної системи на платформі [PythonAnywhere](#) Після завершення етапу реалізації інформаційної системи важливим завданням є забезпечення її стабільної та безперервної роботи. У випадку [Telegram](#)-ботів постає проблема постійного запуску програмного середовища, оскільки для коректної обробки повідомлень бот повинен безперервно працювати та підтримувати з'єднання із серверами [Telegram](#). Якщо запуск здійснюється лише на локальному комп'ютері, виникає необхідність постійно тримати пристрій увімкненим, що є незручним та неефективним у практичному використанні. Одним із рішень даної проблеми є використання хмарних платформ для розгортання програмних систем. Такі сервіси дозволяють запускати програмний код на віддалених серверах без необхідності використання власного серверного обладнання. Для реалізації даної інформаційної системи було використано безкоштовну платформу

» Знайдено посилань: 0%

id: 30

«[PythonAnywhere](#)»,

яка надає можливість запуску [Python](#)-застосунків у хмарному середовищі.

» Знайдено посилань: 0%

id: 31

«[PythonAnywhere](#)»

є спеціалізованою онлайн-платформою для розробки та запуску [Python](#)-проектів. Сервіс дозволяє працювати з файлами, запускати скрипти, налаштовувати віртуальне середовище та підтримувати безперервне функціонування програм. Однією з важливих переваг платформи є наявність безкоштовного тарифного плану, якого достатньо для запуску невеликих інформаційних систем або [Telegram](#)-ботів навчального чи демонстраційного призначення, а саме одного файлу з розширенням

» Знайдено посилань: 0%

id: 32

«[Python](#)».

Використання

» Знайдено посилань: 0%

id: 33

«[PythonAnywhere](#)»

дозволяє уникнути необхідності постійного використання власного комп'ютера як сервера. Крім того, відсутня потреба у придбанні окремого хостингу або налаштуванні власної серверної інфраструктури. Це особливо актуально для невеликих проектів, де важливими факторами є простота розгортання, доступність та мінімальні фінансові витрати. Перед початком розгортання інформаційної системи на платформі

» Знайдено посилань: 0%

id: 34

«[PythonAnywhere](#)»

необхідно створити обліковий запис (рис. 3.16) та підготувати програмні файли. До складу проекту входять програмний код [Telegram](#)-бота, [Excel](#)-файл із даними, а також список необхідних бібліотек для роботи системи. Після створення акаунта користувач отримує доступ до файлової структури та веб-консолі, через яку виконується подальше налаштування середовища. Рис. 3.16. Обліковий запис на сайті

» Знайдено посилань: 0%

id: 35

«[PythonAnywhere](#)»

Першим етапом інтеграції є завантаження програмних файлів на сервер платформи. Це може виконуватися через файловий менеджер [PythonAnywhere](#) або за допомогою систем контролю версій. Після завантаження необхідно перевірити правильність структури директорій та доступність усіх файлів, що використовуються системою. Відповідний

приклад файлової структури представлено на рис. 3.17. Рис. 3.17. Файлова структура Наступним кроком є налаштування планувальника задач для того щоб запустити бота у конкретний час та для його перезапуску в один і той самий час кожного дня. Для цього на вкладці

» Знайдено посилань: 0%

id: 36

«Tasks»

створюємо задачу, задаємо час, надаємо посилання на файл у файловій структурі та опціонально можна записати нотатки до задачі. Фрагмент процесу створення завдання представлено на рис. 3.18. Рис. 3.18. Створення завдання для запуску файлу Після налаштування відображається готове завдання з інформацією про термін дії завдання, кнопки редагування, видалення, зупинки або продовження терміну дії завдання (рис. 3.19). важливо зазначити дві деталі. По-перше, у безкоштовному тарифному плані можна запустити лише одне завдання та користуватися вічно ним. По-друге, термін дії завдання не припиниться раптом, адже за 2 дні до кінця терміну на пошту приходить повідомлення з попередженням про закінчення терміну дії завдання, приклад якого зображено на рисунку 3.20. Рис. 3.19. Вид готового завдання Рис. 3.20. Попередження про закінчення терміну дії завдання Окрему увагу приділено забезпеченню постійної роботи системи. У локальному середовищі робота бота припиняється після вимкнення комп'ютера або завершення процесу виконання програми. Використання рішення

» Знайдено посилань: 0%

id: 37

«PythonAnywhere»

дозволяє підтримувати роботу [Telegram](#)-бота незалежно від стану локального пристрою користувача. Це забезпечує доступність системи у будь-який момент часу та підвищує зручність її використання. Під час інтеграції також враховано питання безпеки та конфіденційності. На жаль, неможливо зберігати токен [Telegram](#)-бота окремо від основного коду, тому це дозволяє збільшити ризик несанкціонованого доступу, але всі дані зберігаються на захищених серверах

» Знайдено посилань: 0%

id: 38

«PythonAnywhere».

Крім того, використання хмарного середовища дозволяє централізовано керувати файлами та оновленнями системи. Для узагальнення процесу розгортання інформаційної системи можна виділити основні етапи інтеграції: Створення облікового запису на платформі [PythonAnywhere](#); Завантаження програмного коду та [Excel](#)-файлу; Налаштування завдання запуску; Перевірка працездатності системи. У результаті використання платформи [PythonAnywhere](#) забезпечено стабільне функціонування інформаційної системи без необхідності використання власного серверного обладнання або постійно увімкненого персонального комп'ютера. Обране рішення дозволило значно спростити процес розгортання системи, забезпечити її доступність та створити зручні умови для подальшої експлуатації й розвитку. 3.4. Тестування інформаційної системи та аналіз результатів її роботи Перед початком функціонального тестування інформаційної системи було виконано налаштування програмного середовища та встановлення всіх необхідних бібліотек, що використовуються в роботі [Telegram](#)-бота. Оскільки система реалізована мовою програмування [Python](#) і використовує сторонні програмні модулі для роботи з [Telegram](#) та [Excel](#)-файлами, важливим етапом стала перевірка наявності відповідних компонентів у середовищі виконання. Під час першого запуску програмного коду можуть виникати помилки, пов'язані з відсутністю необхідних бібліотек. Зокрема, у процесі тестування було виявлено відсутність бібліотеки

» Знайдено посилань: 0%

id: 39

«pandas»,

яка використовується для зчитування та обробки даних з [Excel](#)-файлів. Для усунення проблеми було виконано встановлення пакета за допомогою менеджера пакетів

» Знайдено посилань: 0%

id: 40

«pip».

Рис. 3.21. для роботи з табличними даними Після усунення першої помилки було проведено повторний запуск системи. Наступним етапом стала перевірка наявності бібліотеки

Знайдено посилань: 0% id: 41 «aiogram»,
яка забезпечує взаємодію розробленої інформаційної системи з Telegram. Відсутність даного модуля також призводить до неможливості запуску Telegram-бота, тому було виконано його встановлення. Рис. 3.22. Встановлення бібліотеки
Знайдено посилань: 0% id: 42 «aiogram»
Додатково під час роботи з Excel-файлами виникла необхідність встановлення бібліотеки
Знайдено посилань: 0% id: 43 «openpyxl»,
яка використовується бібліотекою pandas для зчитування файлів формату *.xlsx. Після встановлення даного компонента система отримала можливість коректно працювати з електронними таблицями та виконувати пошук необхідної інформації. Рис. 3.23. Встановлення бібліотеки
Знайдено посилань: 0% id: 44 «openpyxl»
Після встановлення всіх необхідних компонентів було виконано оновлення пакетів та перевірку їх актуальних версій. Для спрощення процедури налаштування програмного середовища може використовуватися також єдина команда:
Знайдено посилань: 0,01% id: 45 «pip install pandas aiogram openpyxl -upgrade».
Рис. 3.24. Спрощена команда для налаштування Виконані дії дозволили підготувати програмне середовище до подальшого тестування та забезпечити стабільний запуск усіх функціональних модулів інформаційної системи. Перевірка залежностей є важливим етапом впровадження будь-якого програмного продукту, оскільки навіть незначна відсутність необхідної бібліотеки може призвести до повної непридатності системи. Після завершення етапу реалізації інформаційної системи важливим кроком є проведення тестування, яке дозволяє перевірити коректність роботи всіх її компонентів, виявити можливі недоліки та оцінити загальну ефективність функціонування. Тестування охоплює як окремі модулі, так і систему в цілому, включаючи взаємодію між Telegram-ботом та модулем обробки табличних даних. Основною метою тестування є перевірка відповідності реалізованої системи сформованим функціональним вимогам. Особлива увага приділяється перевірці коректності обробки запитів користувачів, точності пошуку даних у Excel-файлі, а також стабільності роботи Telegram-бота при різних сценаріях використання. Для цього використовуються як стандартні, так і граничні вхідні дані, що дозволяє оцінити поведінку системи у різних умовах. Першим етапом тестування є перевірка базового сценарію роботи. Він включає запуск бота, введення коректного ідентифікаційного номера, підтвердження особи та отримання результату. У ході виконання цього сценарію перевіряється правильність обробки даних, відповідність отриманої інформації вмісту Excel-файлу та швидкість формування відповіді. Результати на рисунку 3.25 показують, що система коректно виконує всі етапи обробки та надає точні дані. Рис. 3.25. Результат роботи першого етапу тестування Другим етапом є тестування обробки помилкових ситуацій. Зокрема, перевіряється реакція системи на введення неіснуючого ідентифікаційного номера, некоректного формату даних або відмови від підтвердження. У таких випадках система повинна коректно повідомляти про помилку та пропонувати повторити введення, що продемонстровано на рисунку 3.26. Проведене тестування показало, що реалізовані механізми обробки помилок працюють стабільно та забезпечують зручну взаємодію з користувачем навіть у нестандартних ситуаціях. Рис. 3.26. Результат роботи другого етапу тестування Окремо проведено тестування продуктивності системи. Було оцінено час обробки запитів при різній кількості записів у таблиці та при одночасному зверненні кількох користувачів. Використання ефективних інструментів обробки даних дозволило забезпечити швидку реакцію системи, що є важливим фактором для практичного використання. Для узагальнення результатів тестування доцільно представити їх у вигляді таблиці, що дозволяє наочно оцінити ефективність роботи системи при різних сценаріях. Таблиця 3.1 Результати тестування інформаційної системи № Тестовий сценарій Вхідні дані Очікуваний результат Фактичний результат Оцінка 1

Базовий сценарій Коректний ІНН Виведення даних користувача та суми боргу Дані відображено коректно Успішно 2 Невірний ІНН Неіснуючий номер Повідомлення про відсутність даних Виведено повідомлення про помилку Успішно 3 Некоректний формат Символи замість цифр Повідомлення про помилку введення Помилка оброблена коректно Успішно 4 Відмова від підтвердження Натискання

Знайдено посилань: 0%

id: 46

"Hi"

Запит повторного введення Повторний запит виконано Успішно 5 Навантаження Декілька запитів одночасно Стабільна робота системи Затримки відсутні Успішно Представлені результати свідчать про те, що інформаційна система успішно справляється з основними сценаріями використання та демонструє стабільну роботу навіть у складніших умовах. Усі ключові функції працюють відповідно до заданих вимог, а механізми обробки помилок забезпечують надійність взаємодії з користувачем. Проведене тестування підтверджує ефективність обраних підходів до реалізації системи. Отримані результати дозволяють зробити висновок про готовність системи до практичного використання та створюють основу для подальшого розвитку і вдосконалення її функціональних можливостей. 3.5. Перспективи розвитку та масштабування системи Розроблена інформаційна система, що поєднує обробку табличних даних [Excel](#) із використанням [Telegram](#)-бота, демонструє ефективність обраного підходу та підтверджує доцільність використання подібних рішень у практичній діяльності. Водночас сучасні вимоги до інформаційних систем передбачають їх постійне вдосконалення, розширення функціональних можливостей та адаптацію до нових умов використання [22]. Це відкриває широкі перспективи для подальшого розвитку розробленої системи. Одним із напрямів удосконалення є розширення можливостей обробки даних. На поточному етапі система орієнтована на пошук інформації за одним ідентифікаційним параметром, однак у майбутньому може бути реалізована підтримка складніших запитів. Зокрема, передбачено можливість фільтрації за декількома параметрами, виконання сортування даних або формування узагальнених звітів. Це дозволить підвищити гнучкість системи та розширити сферу її застосування. Важливим напрямом розвитку є вдосконалення структури зберігання даних. Використання [Excel](#)-файлів є зручним для невеликих обсягів інформації, однак зі збільшенням кількості записів виникає потреба у використанні більш продуктивних рішень. У цьому контексті доцільним є перехід до систем керування базами даних, що забезпечують швидший доступ до інформації, підтримку багатокористувацького режиму та підвищений рівень надійності зберігання даних. Окрему увагу можна приділити вдосконаленню інтерфейсу взаємодії з користувачем. Хоча [Telegram](#)-бот уже забезпечує зручний доступ до інформації, існує можливість розширення його функціональності за рахунок додавання нових елементів інтерфейсу. Наприклад, можуть бути реалізовані інтерактивні меню, багаторівневі сценарії взаємодії або автоматичні підказки для користувачів. Це сприятиме підвищенню зручності використання системи та зменшенню кількості помилок під час введення даних. Ще одним перспективним напрямом є підвищення рівня безпеки інформаційної системи. Оскільки система працює з персональними даними, важливо забезпечити їх захист від несанкціонованого доступу. Це може бути досягнуто шляхом впровадження додаткових механізмів автентифікації, обмеження доступу до даних або використання шифрування, а також перегляду логіки спілкування з ботом, адже зараз будь-який користувач може підібрати чужий ІПН та отримати дані про особу. Такі заходи дозволять підвищити довіру до системи та забезпечити відповідність сучасним вимогам інформаційної безпеки. Також варто розглянути можливість інтеграції системи з іншими інформаційними ресурсами. Наприклад, система може бути доповнена можливістю отримання даних із зовнішніх сервісів або обміну інформацією з іншими програмними продуктами. Це дозволить створити більш комплексне рішення, яке забезпечує не лише доступ до даних, але й їх активне використання у різних процесах. Перспективним є також розвиток функцій аналітики. На основі накопичених даних система може виконувати аналіз та формувати статистичні звіти. Це відкриває можливості для прийняття обґрунтованих рішень та підвищення ефективності використання інформаційних ресурсів. Реалізація таких функцій дозволить перетворити систему з інструменту доступу до даних у повноцінний засіб їх аналізу. Окремо можна виділити можливість масштабування системи. Розроблена архітектура дозволяє адаптувати систему до збільшення кількості користувачів або обсягу даних без суттєвих змін у її структурі. Це забезпечує довгострокову перспективу використання та дозволяє розширювати функціональність відповідно до зростаючих

потреб. У результаті розгляду перспектив розвитку можна зробити висновок, що створена інформаційна система має значний потенціал для подальшого вдосконалення. Розширення функціональності, підвищення продуктивності, покращення інтерфейсу та забезпечення безпеки даних дозволяють адаптувати систему до більш складних умов використання та забезпечити її ефективне застосування у різних сферах діяльності. Висновки до розділу У межах третього розділу розглянуто практичну реалізацію інформаційної системи, що забезпечує обробку табличних даних та взаємодію з користувачем через [Telegram](#)-бота. Основну увагу приділено створенню програмних модулів, які реалізують зчитування та обробку даних з [Excel](#)-файлів, організацію діалогової взаємодії та формування відповідей на запити користувачів. Це дозволило перевести розроблену на попередніх етапах модель у працездатний програмний вигляд. У процесі реалізації модуля обробки даних забезпечено коректне зчитування інформації з табличного джерела, її структурування та виконання операцій фільтрації за заданими параметрами. Використання відповідних інструментів дозволило досягти високої швидкості обробки запитів і забезпечити точність отриманих результатів. Реалізований підхід підтвердив ефективність використання табличних даних як джерела інформації для інформаційних систем невеликого масштабу. Розробка [Telegram](#)-бота забезпечила створення зручного інтерфейсу взаємодії, який дозволяє користувачам отримувати необхідну інформацію у простому та доступному форматі. Реалізована логіка діалогу відповідає попередньо спроектованим алгоритмам та забезпечує послідовне виконання всіх етапів обробки запиту. Використання інтерактивних елементів дозволило підвищити зручність роботи з системою та зменшити ймовірність помилок. Проведене тестування підтвердило працездатність інформаційної системи в різних умовах використання. Перевірка базових сценаріїв, обробки помилок та роботи під навантаженням показала стабільність функціонування та відповідність системи встановленим вимогам. Отримані результати свідчать про надійність реалізованих рішень та можливість їх практичного застосування. Розглянуті перспективи розвитку демонструють потенціал подальшого вдосконалення системи. Передбачено можливість розширення функціональності, підвищення продуктивності, вдосконалення інтерфейсу та покращення захисту даних [4]. Це створює передумови для адаптації системи до більш складних умов використання та розширення сфери її застосування. У результаті виконаних робіт реалізовано повноцінну інформаційну систему, яка поєднує обробку табличних даних із зручним інтерфейсом взаємодії. Отримані результати підтверджують доцільність обраних підходів і створюють основу для подальшого розвитку та вдосконалення системи.

ЗАГАЛЬНІ ВИСНОВКИ У процесі виконання дипломної роботи було розглянуто особливості створення інформаційної системи для автоматизованої обробки та фільтрації даних із [Excel](#)-файлів із використанням [Telegram](#)-бота та засобів мови програмування [Python](#). Актуальність обраної тематики підтверджується стрімким розвитком цифрових технологій, активним використанням месенджерів у повсякденному житті та зростанням потреби в оперативному доступі до структурованої інформації без необхідності використання складних програмних комплексів або окремих серверних рішень. Під час виконання роботи було проаналізовано сучасні підходи до організації табличних даних, особливості використання [Excel](#)-файлів у якості джерела інформації та можливості бібліотек [Python](#) для автоматизованої обробки даних. Окрему увагу приділено бібліотеці

» Знайдено посилань: 0%

id: 47

«[pandas](#)»,

яка забезпечує ефективне зчитування, фільтрацію та обробку великих обсягів структурованої інформації. Розглянуто переваги використання [Telegram](#)-ботів як зручного інтерфейсу доступу до інформаційних систем, що дозволяє значно спростити процес взаємодії користувача із системою. У ході роботи було сформовано функціональні вимоги до інформаційної системи, визначено структуру взаємодії між її компонентами та реалізовано архітектуру [Telegram](#)-бота. Для візуалізації логіки роботи системи використано [UML](#)-моделювання, зокрема діаграми прецедентів, діяльності, компонентів та послідовності. Це дозволило наочно відобразити алгоритми роботи системи, сценарії взаємодії користувача із ботом та внутрішню структуру програмного рішення. У результаті реалізації практичної частини було створено інформаційну систему, здатну автоматично обробляти [Excel](#)-файли, здійснювати пошук записів за ідентифікаційним номером користувача та формувати відповідь у вигляді повідомлення [Telegram](#)-бота. Реалізовано механізм перевірки коректності введених даних, підтвердження особи користувача та обробки помилок введення. Для забезпечення стабільності роботи системи використано

перевірку типів даних, очищення введених значень від зайвих символів та механізми обробки винятків. Особливу увагу приділено тестуванню інформаційної системи. Було перевірено коректність роботи основних функціональних модулів, швидкість обробки запитів, стабільність роботи [Telegram](#)-бота та правильність взаємодії з [Excel](#)-файлами. Проведене тестування показало, що система коректно реагує як на правильні, так і на помилкові вхідні дані, забезпечує швидкий пошук необхідної інформації та стабільно працює під час повторних звернень користувачів. У роботі також розглянуто питання практичного розгортання [Telegram](#)-бота на платформі [PythonAnywhere](#). Використання даного сервісу дозволило реалізувати цілодобову роботу інформаційної системи без необхідності постійного використання власного комп'ютера або придбання окремого серверного обладнання. Описано основні етапи налаштування середовища, встановлення бібліотек, завантаження програмного коду та запуску [Telegram](#)-бота у віддаленому середовищі. Результати підтверджують ефективність використання мови програмування [Python](#) та [Telegram](#) для створення інформаційних систем автоматизованої обробки даних. Розроблена система характеризується простотою використання, достатнім рівнем функціональності, можливістю подальшого розширення та адаптації до інших типів структурованих даних. Реалізований підхід дозволяє автоматизувати процес пошуку та надання інформації, зменшити час обробки запитів і підвищити зручність взаємодії користувачів із системою. СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ Друковані джерела Алексєєв В. Є. Основи програмування на [Python](#). – Київ: Видавництво

» Знайдено посилань: **0%**

id: **48**

«Наука»,

2020. – 320 с. Бондаренко М. Ф. Інформаційні системи та технології: навч. посіб. – Харків: ХНУРЕ, 2019. – 280 с. Головка В. А. Основи розробки програмного забезпечення. – Київ: КНУ, 2018. – 356 с. Грицюк Ю. І. Бази даних та інформаційні системи. – Львів: ЛНУ ім. І. Франка, 2021. – 310 с. Дорошенко А. Ю. Алгоритми та структури даних. – Київ: ВПЦ

» Знайдено посилань: **0%**

id: **49**

«Київський університет»,

2017. – 295 с. Ковальчук В. П. Інформаційні технології в сучасному суспільстві. – Одеса: ОНПУ, 2020. – 270 с. Лаптев О. А. Розробка інформаційних систем: навч. посіб. – Київ: КПІ, 2019. – 330 с. Марченко О. І. Мови програмування та їх застосування. – Харків: ХНЕУ, 2018. – 260 с. Норвіг П., Рассел С. Штучний інтелект: сучасний підхід. – Київ: Діалектика, 2021. – 1408 с. Петров В. В. Основи програмування на [Python](#). – Київ: BHV, 2020. – 384 с. Sommerwil I. Інженерія програмного забезпечення. – Київ: Вільямс, 2016. – 912 с. Трофименко О. Г. Інформаційні системи та технології. – Київ: Центр учбової літератури, 2020. – 300 с. Електронні ресурси [Python Software Foundation](#). [Python Documentation](#) [Електронний ресурс]. – Режим доступу: <https://docs.python.org/>

🔗 Секція з посиланням: **0,06%** відмічено посиланнями: <https://docs.aiogram.dev/>

id: **50**

[python.org/](https://docs.aiogram.dev/) (дата звернення: 10.04.2026). [pandas Documentation](#). [Powerful data analysis tools](#) [Електронний ресурс]. – Режим доступу: <https://pandas.pydata.org/docs/> (дата звернення: 11.04.2026). [Telegram Bot API](#) [Електронний ресурс]. – Режим доступу: <https://core.telegram.org/bots/api> (дата звернення: 12.04.2026). [aiogram Documentation](#) [Електронний ресурс]. – Режим доступу: [https://docs.aiogram](https://docs.aiogram.dev/)

🚫 Знайдено плагіату: **0,02%**

id: **51**

<https://naurok.com.ua/zasobi-dlya-analizu-danih-v-elektronn-1-8-resypci-1-dev/> (дата звернення: 12.04.2026). [openpyxl Documentation](#) [Електронний ресурс]. – Режим доступу: [https://openpyxl](https://openpyxl.readthedocs.io/)

🔗 Секція з посиланням: **0%** відмічено посиланнями: <https://openpyxl.readthedocs.io/>

id: **52**

[readthedocs](https://openpyxl.readthedocs.io/)

🚫 Знайдено плагіату: **0,04%**

id: **53**

<https://naurok.com.ua/zasobi-dlya-analizu-danih-v-elektronn-1-8-resypci-1-io/> (дата звернення: 13.04.2026). [Microsoft Excel documentation](#) [Електронний ресурс]. – Режим доступу: <https://adminfinance.umw.edu/less/files/2013/06/Excel-Manual1.pdf> (дата звернення: 13.04.2026). [UML-діаграми](#) [Електронний ресурс]. – Режим доступу: <https://dou>

Секція з посиланням: 0% відмічено посиланнями: https://dou.ua/forums/topic/40575/	id: 54
a/forums/topic/40575/ (дата звернення: 14.04.2026). FoxmindEd .	
Секція з посиланням: 0,01% відмічено посиланнями: https://foxminded.ua/uml-diagramy/	id: 55
Для чого потрібні UML діаграми? [Електронний ресурс]. – Режим доступу: https://foxminded.ua/uml-diagramy/	
Секція з посиланням: 0% відмічено посиланнями: https://foxminded.ua/uml-diagramy/	id: 56
u diagramy/	
 Знайдено плагіату: 0,04%	id: 57
https://naurok.com.ua/zasobi-dlya-analizu-danih-v-elektronn-um-diagramy/ (дата звернення: 14.04.2026). Stack Overflow: Programming O & A Community [Електронний ресурс]. – Режим доступу: https://stackoverflow.com/ (дата звернення: 15.04.2026). GeeksforGeeks . Python Programming Resources [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/python/python-programming-language-tutorial/	
Секція з посиланням: 0% відмічено посиланнями: https://www.geeksforgeeks.org/python/python-programming-language-tutorial/	id: 58
geeksforgeeks.org/python/python-programming-language-tutorial/ (дата звернення: 15.04.2026). ДОДАТОК А Лістинг програмного коду	
Секція з посиланням: 0,03% відмічено посиланнями: https://docs.aiogram.dev/	id: 59
<code>import logging import pandas as pd import asyncio from aiogram import Bot, Dispatcher, types, Router, F from aiogram.filters import Command from aiogram.types import ReplyKeyboardMarkup, KeyboardButton # Налаштування TOKEN =</code>	
 Знайдено посилань: 0%	id: 60
<code>"7687947367:AAE1WhhntvC4mfvr7mT_o2sCwd708Zn58dw"</code>	
<code>file_path =</code>	
 Знайдено посилань: 0%	id: 61
<code>"Документ з долгами.xlsx"</code>	
<code># Завантажуємо дані try: df = pd.read_excel(file_path, sheet_name=</code>	
 Знайдено посилань: 0%	id: 62
<code>"Лист1",</code>	
<code>skiprows=1) df.columns = [</code>	
 Знайдено посилань: 0%	id: 63
<code>"П.І.Б.",</code>	
 Знайдено посилань: 0%	id: 64
<code>"Дата народження",</code>	
 Знайдено посилань: 0%	id: 65
<code>"ІПН",</code>	
 Знайдено посилань: 0%	id: 66
<code>"Заборгованість"</code>	
<code>] # Перетворюємо ІПН на рядки одразу, щоб уникнути проблем зі співпадінням типів df[</code>	
 Знайдено посилань: 0,01%	id: 67
<code>"ІПН"] = df["ІПН"]</code>	
<code>.astype(str).str.strip() except Exception as e: print(f</code>	
 Знайдено посилань: 0%	id: 68

```

"Помилка завантаження Excel: {e}"
) logging.basicConfig(level=logging.INFO) bot = Bot(token=TOKEN) dp = Dispatcher() router =
Router() # Словник для збереження стану користувача user_data = {} # 2. Клавіатури
main_menu = ReplyKeyboardMarkup( keyboard=[[KeyboardButton(text=
"Знайдено посилань: 0% id: 69
"Перевірити заборгованість"
)], resize_keyboard=True ) confirm_menu = ReplyKeyboardMarkup( keyboard=
[[KeyboardButton(text=
"Знайдено посилань: 0% id: 70
"Так"
), KeyboardButton(text=
"Знайдено посилань: 0% id: 71
"Hi"
)], resize_keyboard=True, one_time_keyboard=True ) # 3. Обробники # Команда /start
@router.message(Command(
"start" id: 72
)) async def start_command(message: types.
Message): await message.answer( "Вітаємо! 🎉🎉 Натисніть кнопку нижче або просто
введіть ваш ІПН для перевірки:", reply_markup=main_menu ) # Кнопка головного меню
@router.message(F.text == "Перевірити заборгованість") async def check_debt_btn(message:
types.Message): await message.answer("Будь ласка, введіть ваш Ідентифікаційний номер
(ІПН):") # Обробка кнопок "Так" або "Hi" (Мають бути ВИЩЕ за загальний ввід тексту)
@router.message(F.text.in_(["Так", "Hi"])) async def confirm_identity(message: types.Message):
chat_id = message.chat
if message.text ==
"Знайдено посилань: 0% id: 74
"Так"
and chat_id in user_data: inn = user_data[chat_id][
"Знайдено посилань: 0% id: 75
"ІПН"
] # Шукаємо заборгованість за збереженим ІПН user_row = df[df[
"Знайдено посилань: 0% id: 76
"ІПН"
] == str(inn)] if not user_row.empty: debt = user_row.iloc[0][
"Знайдено посилань: 0% id: 77
"Заборгованість"
] await message.answer(f
"Знайдено посилань: 0,01% id: 78
"🎉🎉 Ваша заборгованість: {debt} грн",
reply_markup=main_menu) else: await message.answer(
"Знайдено посилань: 0,01% id: 79
"Помилка: дані не знайдено. Спробуйте ввести ІПН знову.",
🚫 Знайдено плагіату: 0,02% id: 80
https://likestorm.com.ua/blog/telegram/yak-stvoriti-bota-v-t...
reply_markup=main_menu) # Очищуємо тимчасові дані user_data.pop(chat_id, None) else:
await message.answer
(
"Знайдено посилань: 0% id: 81

```

"Добре, спробуйте ввести коректний ІПН знову:",	id: 81
 Знайдено плагіату: 0,05%	id: 82
<code>https://likestorm.com.ua/blog/telegram/yak-stvoriti-bota-v-t reply_markup=main_menu) # Універсальний обробник вводу (ІПН + перевірка на помилки) @router.message() async def process_text_input(message: types.Message): inn = message.text.strip() # Перевірка: чи складається ввід тільки з цифр if not inn.isdigit(): await message .answer("❗ Помилка введення! ІПН має складатися лише з цифр. Будь ласка, спробуйте ще раз.") return # Якщо це цифри - шукаємо в базі user_row = df[df["ІПН"]== inn] if not user_row.empty: full_name = user_row.iloc[0]["П.І.Б."] # Зберігаємо ІПН у словник, щоб використати його після натискання "Так" user_data[message. Секція з посиланням: 0,08% відмічено посиланнями: https://docs.aiogram.dev/ chat.id] = {"ІПН": inn, "П.І.Б.": full_name} await message.answer("Знайдено запис:\nПІБ: {full_name}\nЦе ви?", reply_markup=confirm_menu) else: await message.answer("❗ Ідентифікаційний номер не знайдено в базі даних. Перевірте правильність і спробуйте ще раз.") # 4. Запуск async def main(): dp.include_router(router) await bot.delete_webhook(drop_pending_updates=True) await dp.start_polling(bot) if __name__ == "__main__": try: asyncio .run(main()) except (KeyboardInterrupt, SystemExit): logging.info("Бот зупинений") ДОДАТОК Б Структура та приклад даних Excel-файлу Рис. Б.1. Таблиця даних Excel-файлу</code>	

Відмова від відповідальності:

Цей звіт повинен бути правильно інтерпретований та проаналізований кваліфікованою особою, яка несе відповідальність за оцінку!

Будь-яка інформація, наведена у цьому звіті, не є остаточною та підлягає ручному огляду та аналізу. Дотримуйтесь вказівок: [Рекомендації з оцінки](#)

Детектор Плагіату - Право знати автора! ☐ SkyLine LLC