НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

«На правах рукопису»
УДК 004.62

До захисту допущено:
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2024 р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-професійною програмою «Інформаційне забезпечення робототехнічних систем»
зі спеціальності 126 «Інформаційні системи та технології»
на тему: «Автоматизована система обробки структурованих даних»

Виконав:
студент 2 курсу, групи ІК-21мп
Лисенко Євгеній Олександрович _____

Керівник:
Доцент каф. ІСТ, к.т.н, доц.,
Батрак Євгеній Олександрович _____

Рецензент:
Професор каф. ІСТ ФІТ КНУ ім. Тараса Шевченка,
д.т.н., професор,
Дружинін Володимир Анатолійович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2024 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Рівень вищої освіти – другий (магістерський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційне забезпечення робототехнічних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____2024 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Лисенко Євгенію Олександровичу

1. Тема дисертації «Автоматизована система обробки структурованих даних», науковий керівник дисертації Батрак Євгеній Олександрович, Доцент каф. ІСТ, к.т.н, доц.,, затверджені наказом по університету від «07» 11 2023 р. № 5168-с.

2. Термін подання студентом дисертації «08» 01 2024 р.

3. Об’єкт дослідження: структуровані дані.

4. Вихідні дані: Обробка табличних даних, збереження налаштувань обробки, графічна організація процесу обробки даних, автоматизація обробки даних, інтуїтивність інтерфейсу.

5. Перелік завдань, які потрібно розробити: проаналізувати предметну область, проаналізувати існуючі рішення систем обробки даних, визначити вимоги до системи, спроєктувати архітектуру системи, створити реалізацію системи, протестувати розроблену реалізацію, провести маркетинговий аналіз стартап-проєкту.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: діаграма прецедентів, діаграма класів, діаграма потоків даних, 2 блок-схеми

алгоритмів, діаграма активності, діаграма послідовності, структурна схема.

7. Орієнтовний перелік публікацій

8. Дата видачі завдання 01.09.2023 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Огляд предметної області та існуючих рішень	15.09.2023	
2	Формування функціональних та нефункціональних вимог до системи	30.09.2023	
3	Проєктування процесів та інтерфейсів системи	10.10.2023	
4	Вибір та обґрунтування елементів та технологій	15.10.2023	
5	Проєктування структурної схеми системи	25.10.2023	
6	Реалізація системи	15.11.2023	
7	Тестування системи	20.11.2023	
8	Розробка стартап-проєкту	30.11.2023	
9	Оформлення матеріалів дисертації	17.12.2023	

Студент Євгеній ЛИСЕНКО

Науковий керівник Євгеній БАТРАК

РЕФЕРАТ

Лисенко Є.О. «Автоматизована система обробки структурованих даних»

Магістерська дисертація містить 111 сторінок, 18 рисунків, 36 таблиць, 9 додатків, 40 джерел.

Об’єкт дослідження: структуровані дані.

Предмет дослідження: автоматизація обробки структурованих даних.

Актуальність теми: щодня для прийняття рішень та аналізу минулих подій та прогнозування використовується обробка даних. Низька гнучкість інформаційних систем відносно постійних змін у світі викликає потребу у вторинній обробці даних, яка часто проводиться вручну. Дане дослідження спрямоване на створення програмного рішення, яке забезпечить інтуїтивний та простий у використанні функціонал з автоматизованої обробки структурованих даних невеликих масштабів, використання якого значно скоротить час обробки даних людиною та, відповідно, час отримання з них дорогоцінної інформації.

Мета дослідження: Метою дослідження є аналіз методів автоматизації обробки структуровних даних невеликого розміру та реалізація автоматизованої системи обробки структурованих даних.

Відповідно до мети дослідження були поставлені його задачі:

– проаналізувати предметну область та існуючі рішення проблеми;

– сформувати вимоги до системи;

– спроєктувати, розробити та протестувати автоматизовану систему обробки структурованих даних у відповідності до поставлених вимог;

– провести маркетинговий аналіз стартап-проєкту на основі розробленої системи.

Ключові слова: ОБРОБКА ДАНИХ, ФОРМАТИ ДАНИХ, СТРУКТУРОВАНІ ДАНІ, АВТОМАТИЗАЦІЯ, ІНТЕРФЕЙС КОРИСТУВАЧА, МАЛІ ДАНІ.

ABSTRACT

Lysenko Y.O. "Automated structured data processing system"

The Master's thesis contains: 111 pages, 18 figures, 36 tables, 9 appendices, 40 sources.

Object of research: structured data.

Subject of research: automation of structured data processing.

Relevance: data processing is used every day to make decisions, perform analysis of past event and prognosis of future ones. The often low flexibility of information systems causes the need for secondary data processing, often performed by hand. This thesis aims at creating a software solution providing a simple and intuitive functionality for automated small structured data processing. It’s usage will significantly decrease the time needed for a person to process moderate amounts of data and, as a result, also decrease the time needed to gain information from said data.

The aim of the study: analysis of methods for structured small data processing and creation of an implementation of an automated structured data processing system.

In order to achieve the aim of the study, following tasks have been set:

– analyse the subject area and existing solutions;

– formulate the requirements the system has to meet;

– design, implement and test the automated structured data processing system according to set requirements;

– conduct the marketing analysis for a startup project based on the implemented system.

Keywords: DATA PROCESSING, DATA FORMATS, STRUCTURED DATA, AUTOMATION, USER INTERFACE, SMALL DATA.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення;

MS – Microsoft (компанія);

API – Application Program Interface (програмний інтерфейс додатку);

WYSIWYG – What You See Is What You Get (що бачиш те і отримуєш, напр. текстовий редактор у якому вигляд сторінки у робочому вікні відповідає вигляду сторінки при друці);

IDE – Integrated Development Environment (інтегроване середовище розробки);

OLAP – OnLine Analytical Processing (аналітична обробка (даних) у реальному часі);

BLOB – Binary Large Object (великий бінарний об’єкт);

СУБД – система управління базою даних;

UX – User eXperience (досвід користувача в контексті використання певного програмного продукту);

UI – User Interface (інтерфейс користувача);

CEF – Common Event Format (текстовий формат безпекових логів, розроблений ArcSight Enterprise Security Manager);

CLF – NCSA Common Log Format (формат логів, що використовується веб-серверами);

ELF – Extended Log Format (формат логів для веб-застосунків).

ВСТУП

Одними з головних ресурсів сьогодення є інформація та дані. Фактично всі сучасні установи, компанії та підприємства використовують їх для різних цілей, зокрема:

– оптимізація бізнес процесів;

– допомога в прийнятті рішень;

– аналітика;

– ведення статистики;

– ведення обліку та звітності [1].

Інформаційні ресурси і розумне їх використання відіграють важливу роль у формуванні сучасного суспільства та умов життя [2]. Від швидкості та коректності обробки даних можуть залежати розміри прибутків, а в окремих випадках – життя та здоров'я людей.

Хоча обробка даних є важливою частиною роботи багатьох установ, не всі з них мають завершену, або мають взагалі, систему автоматизації, яка могла б проводити таку обробку [3, 4]. В таких випадках частіше за все покладаються на ручну обробку даних, що створює простір для помилок, викликаних людським фактором та обмежує об’єми даних, які можна обробити за одиницю часу.

Багатьом людям, що стикаються з необхідністю вручну обробляти дані числового (або іншого) виду на постійній основі без або з обмеженою допомогою автоматизованих систем доводиться витрачати на цю роботу значні зусилля та час, ресурсів яких часто в бізнес-середовищі при необхідності прийняття рішень обмаль або принаймні обмежена кількість.

Розповсюдженим рішенням цієї проблеми є автоматизація на рівні редактора документів, такого як MS Excel, Google Sheets та інші, що передбачає вивчення певної мови написання скриптів обробки даних а також ефективного їх застосування та тестування. Знання з автоматизації такого виду цінуються роботодавцями, так як збільшують продуктивність кожного з працівників, але в той же час роблять організацію залежною від тих, хто має ці знання і може

підтримувати написані скрипти за потреби внесення до них змін. Так як взаємозамінність персоналу є досить важливим фактором для виживання бізнесу, особливо в умовах сучасної України, такий підхід до автоматизації обробки даних вважаємо прийнятним, але не найкращим.

Натомість пропонується розробка такої системи, первинний функціонал якої полягав би в автоматизації обробки даних, наданих користувачем, графічними засобами, тобто з використанням інтуїтивно зрозумілих інструментів та графічного інтерфейсу, вивчення використання яких не потребувало б багато часу та технічних знань від працівника. Така розробка допомогла б позбутися як машинальної роботи, результати якої схильні містити помилки, так і явища «незамінних» працівників, без яких весь бізнес може зупинитись (варто помітити, що, як такі, ексклюзивні знання у працівників не є автоматично негативним явищем, і в абзаці вище мається на увазі ситуація, коли ефективна робота працівників, що мають однакові обов’язки, залежить від знань одного з них).

Варто також зазначити що мова не йде про обробку Великих даних, так як для такого роду діяльності необхідна відмінна за масштабом та формою інфраструктура а також при цьому переслідуються інші цілі аніж при обробці «людських»,– в сенсі таких, що є осяжними для людини,– об’ємів даних.

При обробці даних в невеликих масштабах, наприклад, формування звітів різного виду всередині організації, часто згадуються так звані «Малі дані» (англ. «Small Data). Цей вид даних часто створюється локально самим підприємством, його безпосередніми клієнтами чи партнерами і використовуються для оптимізації його роботи у малому масштабі. Часто на основі обробки малих даних приймаються рішення по відношенню до окремих клієнтів, інцидентів, кампаній чи звітних періодів. Крім того, корисною властивістю таких даних є те, що вони містять в собі статистику та показники роботи підприємства фактично (залежно від розміру зрізу даних та періодичності збору) у реальному часі (відносно контексту збору даних) і через це їх обробка дає швидші та більш наглядні результати, що спонукають до дії та допомагають приймати поточні рішення [5].

Не варто також забувати, що наявні великі автоматизовані системи, що належать тим чи іншим організаціям часто є неповороткими та використовують застарілі технології. Це призводить до того що безпосередня вигода від модернізації таких систем для задоволення змінних потреб їх користувачів, яка виражається в прискоренні бізнес-процесів та зменшенні кількості помилок виявляється значно меншою за можливі ризики з виникнення після та під час розробки неполадок, що можуть повністю зупинити роботу бізнесу на невизначений термін або просто за вартість виконання необхідних доробок, що значно збільшується від віку та застарілості технологій, що використовуються в таких системах.

Зважаючи на викладене в попередньому абзаці, можна заключити, що, незважаючи на технічну можливість здійснення будь-яких доробок в існуючих системах автоматизації бізнесу (зокрема в частині обробки даних) а також теоретичну можливість використання працівниками алгоритмічних методів обробки даних, розробка системи, яка б дозволила зробити автоматизацію обробки даних в невеликих масштабах простою та інтуїтивною має як фінансовий так і чисто практичний сенс. Така система дозволить з’єднати (тимчасово або на постійній основі) застарілі системи з реальністю їх використання та організувати обробку даних малих масштабів без зайвих витрат часу та здобування побічних компетенцій.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

1.1 Малі дані

Великі сучасні компанії (Google, Amazon, American Express, Apple, Spotify, Meta, Starbucks, Netflix та ін.) використовують великі дані (англ. Big Data) для виконання аналізу бізнесу та підтримки прийняття рішень [6]. Наданий список далеко не вичерпний, але допомагає сформувати уявлення про розповсюдженість використання великих даних у бізнесі.

Проте іноді компанії зацікавлені у аналізі т.з. малих даних (англ. Small Data) для оптимізації бізнес-моделей та аналітики. Малі дані мають ряд переваг перед великими, зокрема:

– використання та обробка великих даних потребує розвиненої та сучасної інформаційної інфраструктури на місцях, або оренди таких потужностей;

– збір та обробка великих даних дають історичну, тобто узагальнену картину. Такі результати потребують подальшого аналізу та регресії для їх застосування у прийнятті рішень;

– перед обробкою великі дані треба з власної ініціативи збирати з джерел, які мають також бути надійними та у достатній мірі репрезентативними для кожного конкретного випадку. Крім того, після збору великі дані необхідно принаймні тимчасово зберігати;

– малі дані характеризуються вищою персоналізацією, як на рівні бізнесу, так і для кожного конкретного клієнта чи користувача;

– за короткий час необхідний для аналізу малих можна вже робити певні висновки, а на основі них прийняти рішення в областях роботи організації, які відповідають зібраним даним;

– малі дані можуть надавати інформацію практично в реальному часі, що дозволяє швидко приймати рішення, які відповідатимуть змінам у середовищі роботи організації, команди, або навіть одного співробітника;

– малі дані можна збирати без агрегації та майже без обробки, що значно полегшує автоматизацію цього процесу а також роботу аналітиків [5, 7, 8].

Збір малих даних часто є окремим для кожного відділу або команди в організації, що проілюстровано на рисунку 1.1.

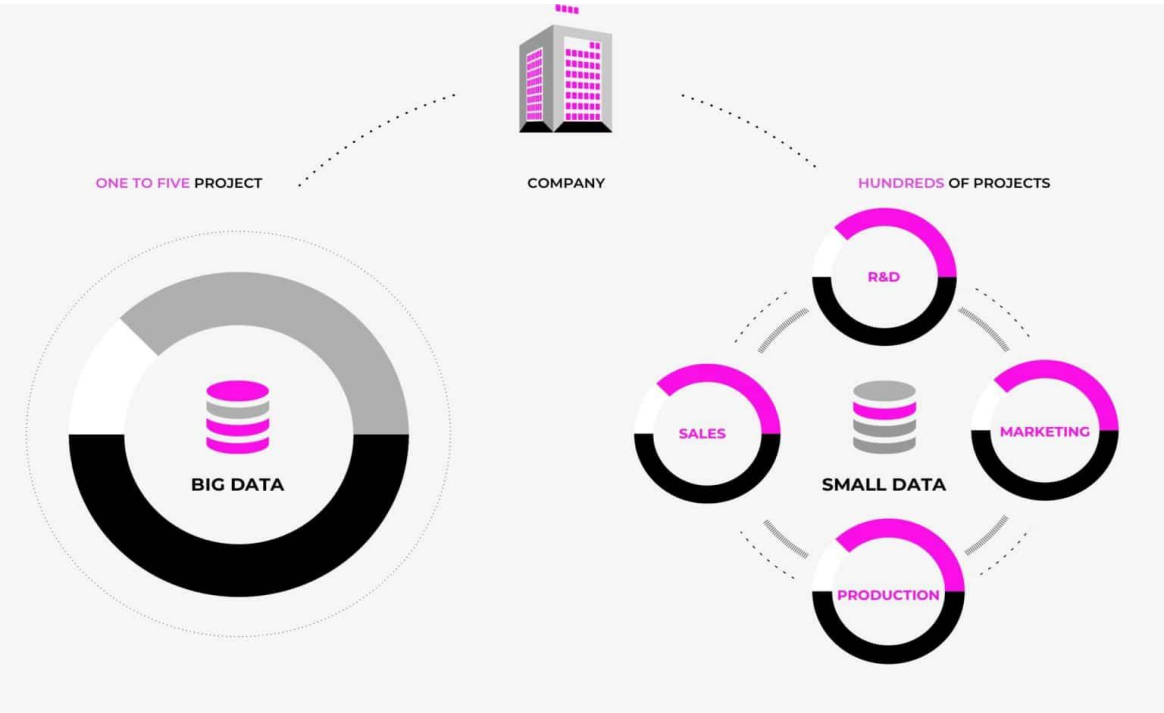


Рисунок 1.1 – Різниця між збором малих та великих даних [9]

Проте, варто зазначити що малі та великі дані не є суттєво різними поняттями. Велику кількість малих даних можна назвати великими. Очевидно, що різниця досить семантична, але, тим не менше, відмінні масштаби та об’єми збору та обробки даних вже ставлять перед аналітиками та розробниками принципово різні задачі при роботі з великими та малими даними. Звідси виникають відмінні методи обробки, аналізу та інтерпретації отриманої інформації для прийняття рішень. Щодо описаної схожості доцільно навести цитату (переклад з англ.):

«Так само як зараз було б дивно говорити про «Велике програмне забезпечення», – ніби сам по собі розмір є мірою цінності, – нам слід, і одного дня так і буде, дивуватися назві «Великі дані». Сам по собі розмір не має значення – важливе саме володіння даними будь-якого розміру, які допомагають нам розв’язати проблему чи відповісти на наше питання», Руфус Поллок (Rufus Pollock) в інтерв’ю The Guardian [10] (Руфус Поллок - член та засновник the Open Knowledge Foundation, економіст).

1.2 Обробка табличних даних

Особлива увага під час розробки системи приділятиметься обробці табличних даних, так як таблиці різного виду є одним з найрозповсюдженіших видів структурованих даних.

Таблиця – це спосіб розміщення інформації чи даних, частіше за все у рядках і стовпцях або у складніших структурах. Вони широко використовуються в комунікації, дослідженнях, аналізі даних, з'являються в друкованих ЗМІ, рукописних нотатках, комп'ютерному ПЗ та інших місцях.

Точні конвенції та термінологія для опису таблиць відрізняються залежно від контексту їх використання. Таблиці часто сильно відрізняються за структурою, гнучкістю, видом запису, представленням та використанням. У книгах і технічних статтях таблиці зазвичай представлені окремо від основного тексту нумерованими плаваючими блоками з підписами [11, с. 1-2], а в контексті обробки даних таблиці часто є окремими файлами з даними, які за допомогою комп'ютерних програм інтерпретуються як таблиці з певною структурою.

Представлені в статті [11] положення, хоча і описані для організації видобутку даних з таблиць, представлених в текстових документах медичного характеру, будуть корисними для подальшого проєктування об'єктів розроблюваної автоматизованої системи.

Перед обробкою таблиці визначаються дві її характеристики:

– тип;

– модель даних.

За типом таблиці можна визначити необхідні методи її обробки та інтерпретації, а визначення структури моделі даних таблиці дозволяє цю обробку автоматизувати та для деяких випадків спростити.

Автор статті [11] визначає наступні види таблиць:

– одновимірні (спискові);

– двовимірні (матричні);

– багатовимірні.

Одновимірні таблиці описуються одним заголовком. Такі таблиці можуть записуватись у декілька колонок для економії місця, але в контексті програмної обробки це не релевантно, так як такі таблиці описуються простим (впорядкованим чи ні) списком значень. Приклад одновимірної таблиці наведений в таблиці 1.1:

Таблиця 1.1 – Приклад одновимірної таблиці

Доступні кольори
Блакитний
Жовтий
Червоний
Синій
…

Двовимірні таблиці мають рядки та стовпці. Дані в них розміщуються у вигляді матриці, зазвичай з двома позначеннями – заголовками рядка та стовпця. Такі таблиці також можуть мати декілька рівнів заголовків рядків та стовпців. Приклад двовимірної таблиці наведений в таблиці 1.2:

Таблиця 1.2 – Приклад двовимірної таблиці

Номер	Ім'я	Посада	…
2	Андрій	директор	…
4	Микола	менеджер	…
…	…	…	…

Багатовимірні таблиці мають більше двох вимірів та поділяються на таблиці з супер-рядками та мульти-таблиці.

Таблиці з супер-рядками мають мають в собі рядки, що групують заголовки інших рядків. Рівнів такого вкладення може бути декілька, в результаті чого формується деревоподібна структура. Такі таблиці зазвичай представляються у текстовому вигляді, де рівні структури позначаються відступами. Приклад таблиці з супер-рядками наведений в таблиці 1.3:

Таблиця 1.3 – Приклад таблиці з супер-рядками

Ідентифікатор	Температура, °C	Швидкість, м/с	…
A3M102	20	2	…
	30	2,1	…
A4M102	35	3	…
	45	3,1	…
…	…	…	…

Мульти-таблиці складаються з декількох, зазвичай схожих таблиць, об’єднаних в одну. В деяких випадках, заголовки приєднаних таблиць наслідують структуру заголовку першої таблиці. Приклад такої таблиці наводити не будемо, так як така структура частіше зустрічається в друкованому вигляді для наглядності аніж у якості цифрової репрезентації даних.

1.3 Структурованість даних

Структурованими називають дані, структура яких є певним чином регульованою та незмінною для всього об’єму кожної окремої одиниці їх зберігання (таблиця, список, документ, файл, чанк, і т.д.). Такі дані легко розподілити по визначеним полям і проводити пошук як за допомогою розроблених людиною запитів, так і в режимі реального часу з допомогою програмного відповідного забезпечення [12].

Неструктурованими називають дані, які важко або неможливо зберігати з допомогою визначених наперед структур. Основними прикладами неструктурованих даних є:

– відео;

– аудіо;

– фото;

– текст;

– дані про погоду (в окремих випадках);

– геологічні дані (в окремих випадках) і т.п..

Ефективне виконання пошукових запитів, а отже і обробка, для таких даних вимагає існування метаданих, а зберігання потребує більше ресурсів, складніших структур та використання нетривіальних файлових систем (хоча в сучасному світі останній пункт можна опустити). Зберігання і обробка неструктурованих даних ще більше ускладнюється неможливістю знати перед отриманням метаданих розміри та іноді форму одиниці даних.

Також визначають напівструктуровані дані, що частіше за все мають вигляд неструктурованих даних, об’єднаних з відповідними метаданими (наприклад, електронна пошта: текстові дані є неструктурованими, але заголовки (назва, тема, основний текст) дозволяють систематизувати їх зберігання та обробку).

1.4 Програмні рішення обробки даних з можливістю автоматизації

1.4.1 Google workspace

Google Workspace це набір веб-застосунків для хмарних обчислень, організації роботи та колаборації, що розробляється компанією Google. До нього входять:

– поштовий клієнт – Gmail;

– комунікація – Contacts, Chat, Meet;

– календар – Calendar;

– хмарне сховище – Drive;

– редактори – Docs, Sheets, Slides, та інші.

Більшість функціоналу з роботи з документами та організації діяльності надається безкоштовно всім користувачам, що мають безкоштовний обліковий запис Google, проте Google Workspace пропонує корпоративні послуги, такі як власні адреси електронної пошти в окремому домені (напр. asd@company.com) та можливість придбати необмежене сховище Google Drive. Крім того доступні адміністративні інструменти для управління власним Workspace, різні

налаштування для його учасників та технічна підтримка телефоном чи електронною поштою 24/7.

В рамках дослідження нас цікавить можливість використання скриптової мови програмування Google Apps Script для написання скриптів, що можуть використовувати та модифікувати дані у багатьох додатках Google Workspace та яка представляється як аналог MS Office VBA та Office.js скриптів відповідно написаних для додатків з пакету MS Office 365 (див. розділ 1.4.2), а також редактор таблиць Google Sheets.

Apps Script це хмарна платформа, що базується на мові програмування JavaScript [13]. Користувачам надається редактор коду у браузері в якому доступні за замовчуванням бібліотеки для взаємодії з додатками Google Workspace. Окрім інших функцій, Apps Script дозволяє створювати власні функції та макроси для Google Sheets [14].

На даний момент інструмент широко застосовується для написання надбудов для застосунків Google Workspace, які кожен користувач може встановити та використовувати, а розробники оновлювати та підтримувати і код яких закритий для користувачів.

Apps Script має ряд обмежень, зокрема:

– скрипти користувачів мають обмежений час для виконання;

– доступ по мережі до інших сервісів та сервісів Google обмежено;

– в операціях з часом та датою часто стаються помилки, так як кінцеві точки роботи надбудов та скриптів можуть знаходитись в різних часових поясах.

Загалом Apps Script представляє собою JavaScript з надбудовами, тому сфери та способи його застосування (в рамках інфраструктури Google) є різноманітними і обробка даних є лише однією з них.

Робоче вікно Google Sheets наведене на рисунку 1.2.

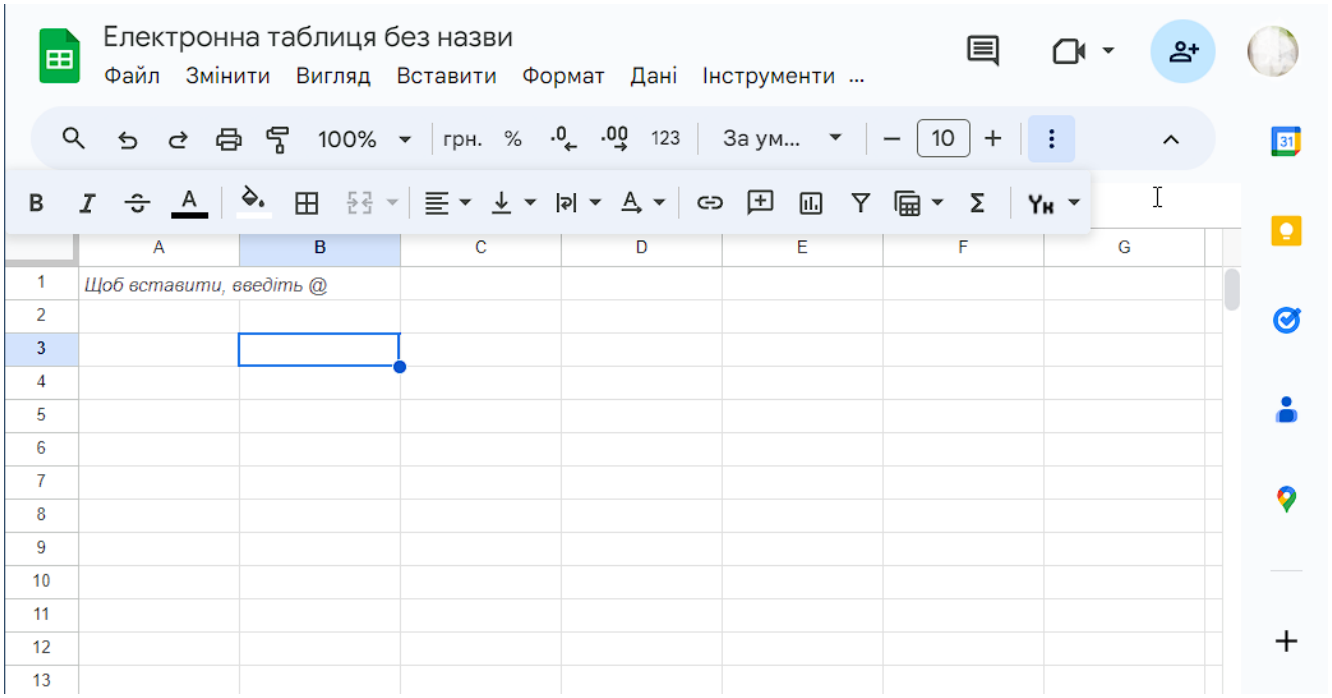


Рисунок 1.2 – Робоче вікно Google Sheets

Загалом додаток має звичний для Google плаский дизайн зі стандартними (для редактора таблиць) функціями на панелі інструментів та робоче поле з комірками з пронумерованими рядками та позначеними літерами стовпцями. Базовий функціонал редактора дещо спрощений у порівнянні з, наприклад, MS Excel, але достатній для виконання повсякденних завдань.

1.4.2 MS Office

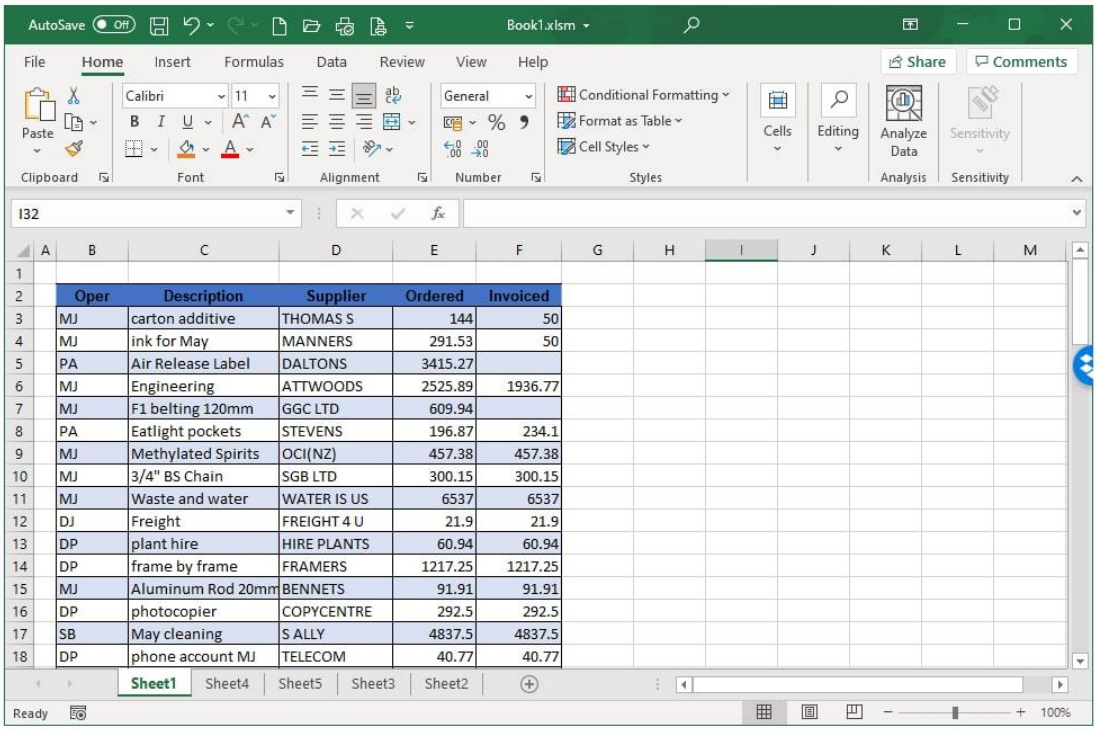
MS Office (MS Office 365, MS 365 та ще ряд назв) - широко відомий набір програм від Microsoft (на поточний момент частина пакету Microsoft 365, розповсюджуваного за моделлю підписки). Він включає в себе, серед інших продуктів, MS Excel, що є одним з найвідоміших редакторів таблиць.

Excel має весь типовий функціонал редактора таблиць та використовує сітку комірок, розміщених в пронумерованих рядках та позначених літерами стовпцях для організації роботи з даними. Також користувачам доступний набір визначених математичних та статистичних функцій, які можна застосовувати як напряму з числами так і зі значеннями в комірках або діапазонах комірок (де це

можливо). Крім того, в редакторі присутня можливість створювати графіки та діаграми різного виду з використанням введених даних [15].

Робоче вікно MS Excel наведено на рисунку 1.3. (див. нижче) Воно має вигляд, аналогічний до Google Sheets, з відмінністю у використанні в Excel стрічки меню, яка групує графічні панелі інструментів та показує одну з них за раз при виборі, та ширшим функціоналом в принципі.

В редакторі також існує можливість створення користувацьких макросів для виконання різноманітних операцій без необхідності написання коду. Схожий на це функціонал задумано реалізувати в розроблюваній системи, проте не для макросів, а для графічного налаштування обробки даних, обраних користувачем.



	Oper	Description	Supplier	Ordered	Invoiced
3	MJ	carton additive	THOMAS S	144	50
4	MJ	ink for May	MANNERS	291.53	50
5	PA	Air Release Label	DALTONS	3415.27	
6	MJ	Engineering	ATTWOODS	2525.89	1936.77
7	MJ	F1 belting 120mm	GGC LTD	609.94	
8	PA	Eatlight pockets	STEVENS	196.87	234.1
9	MJ	Methylated Spirits	OCI(NZ)	457.38	457.38
10	MJ	3/4" BS Chain	SGB LTD	300.15	300.15
11	MJ	Waste and water	WATER IS US	6537	6537
12	DJ	Freight	FREIGHT 4 U	21.9	21.9
13	DP	plant hire	HIRE PLANTS	60.94	60.94
14	DP	frame by frame	FRAMERS	1217.25	1217.25
15	MJ	Aluminum Rod 20mm	BENNETS	91.91	91.91
16	DP	photocopier	COPYCENTRE	292.5	292.5
17	SB	May cleaning	S ALLY	4837.5	4837.5
18	DP	phone account MJ	TELECOM	40.77	40.77

Рисунок 1.3 – Робоче вікно MS Excel

Окрім згаданого функціоналу у Excel реалізовано два способи автоматизації створення та перетворення таблиць, які на даний момент використовуються одночасно:

– Excel VBA (Visual Basic for Applications);

– Office.js (Office scripts) [16].

Excel VBA, старший з двох засобів, це мова програмування, заснована на Visual Basic, яку Microsoft інтегрувала у всі свої продукти Office. Ця мова має широкий функціонал та фактично на ній можна написати будь-який додаток, надбудову чи скрипт для (в нашому випадку) Excel.

VBA легко інтегрується в декілька додатків Office одразу, наприклад: необхідно відправити електронного листа в Outlook зі згенерованою презентацією в PowerPoint коли значення в певній комірці таблиці стає більшим за 90%. При цьому користувачам не потрібно встановлювати нічого додатково, як от при використанні зовнішніх скриптів, написаних на Python чи інших мовах програмування.

Також за роки існування для VBA створено багато документації, посібників та інтернет-ресурсів, що допомагають новим розробникам навчитись використанню цього інструменту.

Проте VBA має і свої проблеми, зокрема:

– незручне використання зовнішніх API;

– швидкість роботи;

– нестабільність, особливо для 32-бітних систем;

– застарілий синтаксис, зокрема для високорівневого програмування;

– відсутність великої підтримки Microsoft для своїх застосунків (на користь Office.js, про який далі), яка обмежується невидаленням її як такої, періодичними змінами, що дозволяють VBA не конфліктувати з новими версіями додатків Office та технічною підтримкою великих клієнтів, бізнес яких працює на VBA;

– обмежена можливість використання новішого чи певним чином зміненого з часом функціоналу додатків Office з коду;

– складність адаптації коду VBA з Windows на MacOS, неможливість використання у всіх інших формах додатків Office (веб, мобільні).

Натомість, Microsoft активно просуває використання Office.js (Office Scritps) – обгортки для JavaScript, розробленої на заміну VBA. Office.js працює на всіх версіях додатків Office і не потребує (теоретично) адаптації під різні платформи, так як код на JavaScript є універсальним.

Переваги Office.js перед VBA у наступному:

– оновлення та доступ до будь-яких JavaScript-бібліотек, що дозволяє реалізувати фактично що завгодно;

– кросплатформеність;

– швидкість використання зовнішніх ресурсів даних;

– JavaScript є широко відомим та багатьом розробникам не доведеться вчити його з нуля для використання.

Незважаючи на переваги Office.js та слабку підтримку VBA, перше скоріше за все не стане єдиним стандартом для всіх користувачів Office, так як багато компаній залежать від використання VBA для ведення свого бізнесу і не захочуть ризикувати фінансовими та часовими втратами, що є неминучими при інтеграції нової технології з нуля.

Крім того, багато спеціалістів з VBA не є за спеціальністю розробниками, і вивчили цю мову для спрощення своїх повсякденних задач. Такі люди скоріше за все не захочуть вчити принципово нову мову для того щоб виконувати вже раніще автоматизовані старим методом задачі [17].

1.4.3 Інші програмні пакети

Тут варто згадати безкоштовний набір програм для офісної роботи LibreOffice (та інші подібні open- чи closed-source пакети програм для роботи з документами), який надає фактично дзеркальний до редакторів Google та MS функціонал з редагування документів та маніпуляції даними (звичайно, без онлайн-функцій та організації робочого процесу для команд, для яких необхідна інфраструктура згаданих компаній).

До складу LibreOffice входять наступні програмні засоби:

– Writer – текстовий редактор з WYSIWYG функціоналом, може створювати заповнювані PDF-форми;

– Calc – табличний редактор, схожий на MS Excel, з окремими відмінностями;

– Impress – редактор презентацій, схожий на MS PowerPoint;

– Draw – редактор векторної та растрової графіки а також інструмент для створення діаграм;

– Math – додаток для створення (та обчислення) математичних формул, які можна вкласти у інші документи набору програм LibreOffice;

– Base – додаток для управління базами даних, схожий на MS Access, що дозволяє створювати як вкладені в документи бази, так і слугувати інтерфейсом для взаємодії з різними сумісними СУБД.

Програми з набору підтримують довгий список альтернативних форматів файлів, а також власний OpenDocument формат для сумісності з різними програмами (проте перетворення форматів не завжди однозначні і зворотні).

LibreOffice підтримує ряд мов для написання скриптів та розширень:

– LibreOffice Basic – для невеликих макросів у Writer, Calc та Base, аналог MS VBA;

– C++;

– Java;

– Python;

– MS CLI;

При чому інтерпретатори для Python та LibreOffice Basic вкладені у більшість інсталяторів LibreOffice, що дозволяє їх використовувати без додаткових надбудов.

API LibreOffice, під назвою “UNO” (Universal Network Objects) є детально задокументованим [18].

1.5 Огляд неінтегрованих в редактор засобів обробки даних

1.5.1 R

R це мова та середовище для статистичних обчислень та створення графічних матеріалів на основі статистичних даних. R – одна з сучасних гілок розвитку мови S (друга – S-PLUS), розробленої в Bell Laboratories [19].

R має досить вичерпний набір статистичних та графічних засобів для обробки та представлення даних, зокрема:

– лінійне та нелінійне моделювання;

– класичні статистичні перевірки;

– аналіз часових послідовностей;

– класифікація та кластеризація даних та інші.

Однією з найбільших переваг R є простота створення та форматування графічних матеріалів (графіків та діаграм різного виду), що фактично одразу готові до використання у публікаціях та звітах і також за потреби можуть включати в себе формули та математичні символи. Хоча налаштування за замовчуванням для цих представлень досить детально пропрацьовані, користувач вільний вносити власні зміни у будь-які деталі.

R доступна безкоштовно за ліцензією GNU General Public License від Free Software Foundation у вигляді вихідного коду. Вона збирається та запускається на різноманітних UNIX платформах (включно з FreeBSD та Linux), Windows та MacOS. [20]

Окрім базового функціоналу, R можна доповнити пакетами, доступними у мережі Інтернет, а також власноруч написаними функціями та кодом.

Також для R доступна IDE Rstudio, що також має відкритий код, доступний за посиланням [21]. Вигляд робочого вікна RStudio зображено на рисунку 1.4.

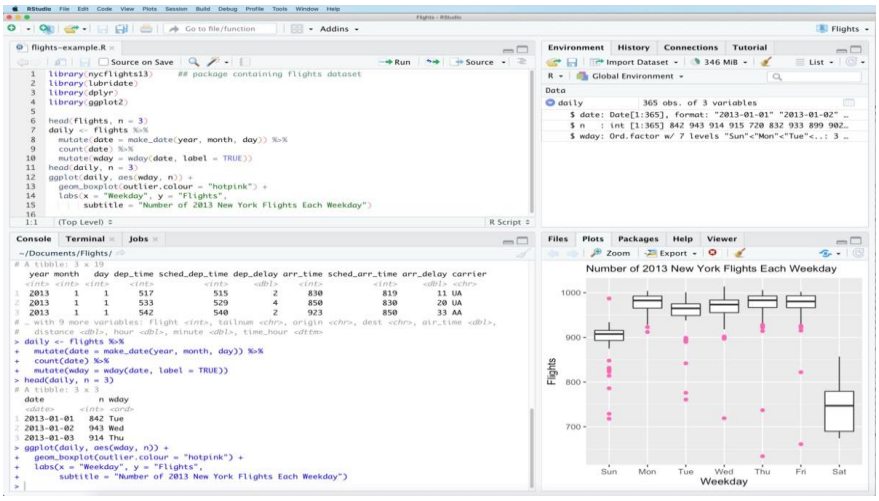


Рисунок 1.4 – Робоче вікно Rstudio (MacOS)

На рисунку 1.4:

– зліва зверху – поле для введення коду;

– справа зверху – вікно змінних, в якому відображаються дані, збережені у пам’ять (зберігаються між виконаннями скриптів);

– зліва знизу – консоль для виводу виконання команд;

– справа знизу – вікно виводу графічних матеріалів (графіків і т.д.);

Окрім згаданих можна відкрити і інші види, а також змінювати розмір вікон та переставляти їх у довільному порядку.

1.5.2 MATLAB

Програмний пакет MATLAB від MathWorks надає функціонал роботи з даними в матричному форматі з допомогою редактора та власної мови програмування. Для MATLAB постійно розробляються та покращуються т.з. тулбокси, кожен з яких спеціалізується на певній сфері обчислень чи виді аналізу і містить в собі необхідні для відповідних обчислень визначення функцій та структур. Всі тулбокси MATLAB детально задокументовані та протестовані[22].

Зокрема, в MATLAB доступні функції для аналізу та візуалізації даних. Підхід до створення графічних об’єктів у MATLAB більш програматичний і потребує розуміння роботи відповідних тулбоксів, але дозволяє створювати читабельні графіки з великою кількістю змінних та порівняльними характеристиками.

Також MATLAB має інтерфейси для використання в обчисленнях коду на наступних мовах:

– Python;

– C/C++;

– Fortran;

– Java та інші.

А також існує можливість проводити обчислення у хмарі MathWorks Cloud або AWS чи Azure.

На рисунку 1.5 зображене робоче вікно Matlab.

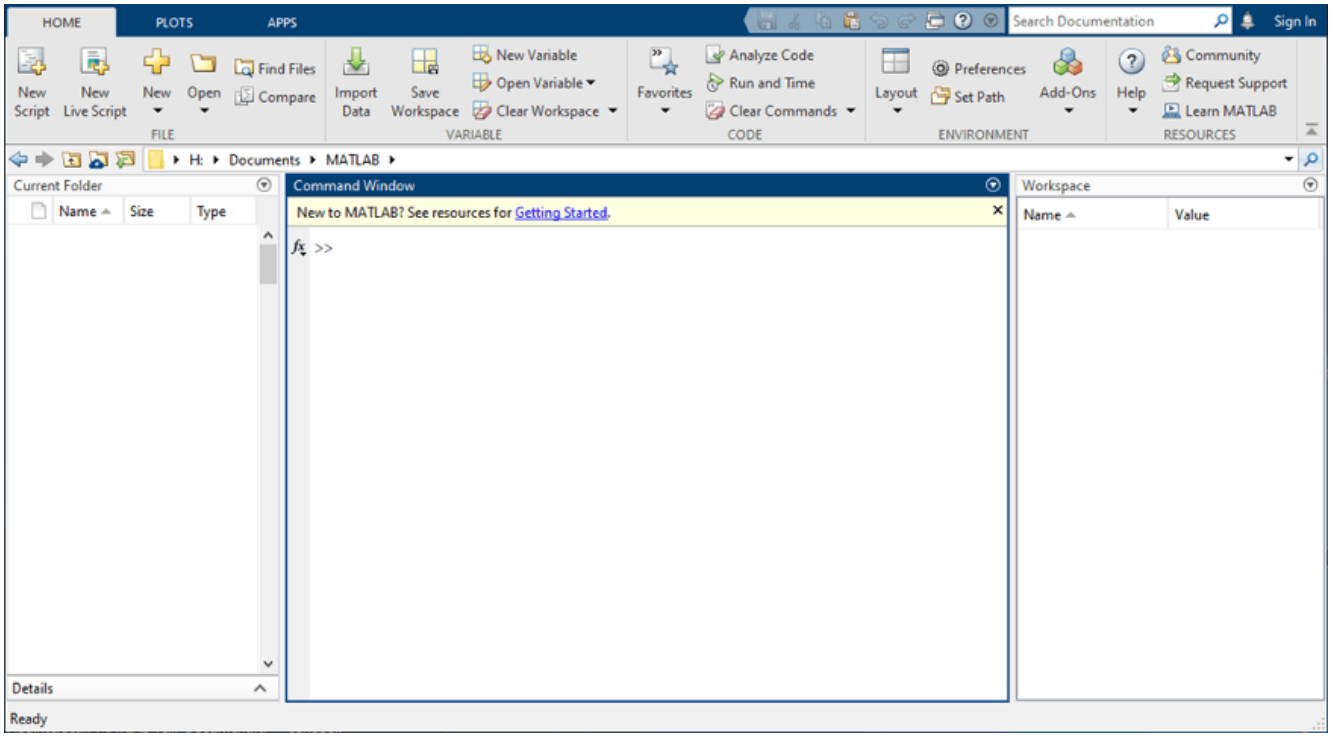


Рисунок 1.5 – Робоче вікно Matlab 2021

На рисунку 1.5:

– згори – стрічка меню з різними інструментами;

– посередині – поле консолі, в яку виводиться вивід функцій а також де можна виконувати команди;

– зліва – дерево проєкту;

– справа – перелік змінних, збережених в пам’яті;

Також в Matlab доступне робоче поле для написання скриптів для виконання декількох команд в послідовності замість виконання по одній в консолі, вікна з згенерованими графіками, що з’являються над основним вікном програми (залежно від налаштувань графіки) та інші різноманітні робочі поля та вікна, що надаються у тулбоксах, що використовує користувач.

За аналогом з RStudio всі частини інтерфейсу (окрім стрічки меню) можна переміщувати та змінювати їх розмір для адаптації під користувача та його поточну задачу у програмі.

1.6 Алгоритмічні засоби обробки даних

1.6.1 M, Power Query

M це мова запитів, на якій розроблений інструмент Power Query (частина Power Pivot від MS, доступного в Excel). Вона є мешап (англ. Mashup) мовою, тобто використовується для написання запитів, що змішують різні дані.

Power Query, написаний на M, складається з виразу let, де дані та вирази зв'язуються між собою та виразу in, що містить результат запиту. Рядки у виразі let, які також називають кроками, розділяються комами, з можливістю написання C-подібних коментарів (/ ...” Для однорядкового та /* ... */ для багаторядкового).

Використання M необхідне для певних видів запитів, але для багатьох простіших завдань користувачі можуть скористатись графічним інтерфейсом Power Query без написання коду [23].

1.6.2 MDX

MultiDimentional eXpressions це мова запитів для OLAP-систем, що використовують СУБД. Мова призначена для роботи з OLAP-кубами (далі – кубами) (багатовимірними (3 і більше вимірів) масивами даних). Також в MDX є функціонал для проведення обчислень, з синтаксисом схожим на табличні формули.

MDX має 6 основних типів даних:

– Scalar – число або рядок, визначається літералом або повертається функцією;

– Dimension/Hierarchy – вимір куба. MDX вважає виміри взаємно незалежними. Виміри містять члени, організовані за певною ієрархією(ями) що складається з рівнів. Визначаються унікальним іменем або повертаються функцією. Деякі імплементації не розрізняють виміри та ієрархії як окремі типи даних, інші навпаки (напр. Microsoft Analysis Services);

– Level – рівень в ієрархії вимірів, визначається унікальним іменем або повертається функцією;

– Member – член ієрархії вимірів. Визначається унікальним іменем, напр. [Time].[Fiscal].[Month].[*August 2023*], повним іменем (qualified name) -//-.[*Q3*].[*August 2023*], чи повертається функцією;

– Tuple – впорядкований набір з одного чи більше членів з різних вимірів. Визначаються переліком членів (напр. ([Time].[Fiscal].[Month].[August], [Customer].[By Geography].[All Customers].[USA], [Measures].[Sales]) для продажів покупцям з США у Серпні) або повертаються функцією;

– Set – впорядкований набір tuple-ів з однаковою вимірністю. Визначається переліком tuple-ів або повертається функцією;

– Інші типи даних у вигляді іменованих властивостей перелічених вище типів, викликаються за назвою або спеціальною функцією для доступу до властивостей.

Приклад запиту, написаного на MDX, зображено на рисунку 1.6.

```
SELECT
   { [Measures].[Store Sales] } ON COLUMNS,
   { [Date].[2002], [Date].[2003] } ON ROWS
FROM Sales
WHERE ( [Store].[USA].[CA] )
```

Рисунок 1.6 – Запит MDX для отримання об’ємів продажів за 2002 та 2003 роки для магазинів у Каліфорнії, США

У наведеному прикладі запит визначає інформацію, що потрапить в результуючий набір даних наступним чином:

– вираз SELECT визначає вісі запиту як член Store Sales виміру Measures і члени 2002 і 2003 виміру Date;

– вираз FROM визначає, що дані беруться з куба Sales;

– вираз WHERE визначає вісь перерізу як член USA.CA виміру Store.

Максимальна кількість вісей запиту MDX – 128.

Для запитів з двома і більше вісями, частина з них має бути визначені колонками (ті, дані з яких треба отримати), а частина рядками (ті, що накладають

обмеження на результат), при цьому порядок розміщення рядків та стовпців у запиті не важливий. Якщо в запиті тільки одна вісь, то вона має бути стовпцем.

Також використання квадратних дужок не обов’язкове якщо ідентифікатор об’єкта не є зарезервованим словом і не містить символів окрім літер, чисел та підкреслень [24].

1.6.3 DAX

Data Analysis eXpressions це мова формул та запитів для MS PoverPivot, Power BI Desktop та табличних моделей SQL Server Analysis Services (SSAS). DAX включає деякі функції, що використовуються у формулах Excel з додатковими функціями, що розроблені для роботи з реляційними даними та проведення динамічної агрегації.

В певній мірі, DAX є еволюцією MDX (див. Вище). Мова розроблена як легка для вивчення і в той же час така, що дає користувачу доступ до потужності та гнучкості PowerPivot та табличних моделей SSAS.

DAX не є мовою програмування.Її можна використовувати для створення власних обчислень для обчислюваних стовпців, вимірів, обчислюваних таблиць, груп обчислень, рядків власного формату та виразів для фільтрування у контексті рольової безпеки у табличних моделях (користувачі мають ролі і за ними отримують доступ до визначених даних та операцій).

DAX може обчислювати значення для 7 типів даних:

– Integer – ціле число;

– Real – дійсне число;

– Currency – валюта;

– Date (datetime) – дата і час;

– Boolean – булеве значення;

– String – рядок;

– Variant – змінний тип. Використовується для результатів виконання функцій, що повертають різні типи даних залежно від певних умов всередині функції;

BLOB-и обробляються табличною моделлю, але ними не можна маніпулювати напряму через вирази DAX [25].

1.6.4 Інші неспеціалізовані мови

Для обробки даних також можна використовувати невеликі програми, написані на різних мовах програмування. Вичерпний список наводити не будемо, так як фактично всі мови програмування так чи інакше мають інструменти для маніпулювання файлами та даними у різному вигляді.

Деякі з мов, наприклад Python, часто нативно підтримуються редакторами для простої інтеграції написаних на них програм у функціонал редактора. В такому випадку розробнику достатньо написати скрипт (іноді у відповідності до наданого розробниками застосунку API), який би виконував бажану ним дію та підключити його в налаштуваннях додатку.

Такий підхід дає свободу використовувати будь-які інструменти конкретної мови програмування на свій розсуд та для своїх потреб. З іншого ж боку, для застосування такого підходу треба мати знання та вміння з використання задуманої мови для поставленої задачі і часто доведеться писати частини скрипта з нуля. Також потенційно можуть виникати проблеми з форматами різних файлів при читанні з них даних.

Загалом цей спосіб найуніверсальніший, але і найбільш затратний по часу і вимогливий до вмінь користувача.

1.7 Бази даних

Бази даних та сумісні з ними СУБД розраховані на зберігання та організацію великих об’ємів даних різного виду (залежно від бази даних) і

відповідно мають засоби для проведення вибірки даних, якими можуть користуватись їх користувачі.

Проте ці засоби зазвичай доступні напряму лише адміністраторам бази та розробникам додатків, що її використовують, а кінцевим користувачам вже представляються отримані та оброблені дані. Це не робить використання баз даних для обробки невеликих об’ємів даних неможливим, але у такому випадку необхідно спочатку розробити спосіб конвертації даних з файлів у сутності в базі, а потім визначити набір запитів, необхідних для формування бажаного результату, а також спосіб перетворення цього результату на файл, який можна зберегти та можливо модифікувати.

Тому в рамках даного дослідження нас більше цікавлять методи, що використовуються у базах даних для зберігання, обробки та перетворень даних.

Почнемо огляд з реляційних баз даних на основі SQL.

При написанні SQL-запиту він проходить наступні кроки (описані в загальних термінах) перед виконанням в СУБД:

– компіляція;

– оптимізація;

– виконання.

Крок компіляції запиту складається з парсингу – розпізнавання окремих виразів та ідентифікаторів мови SQL, після чого відбувається біндинг запиту – генерація плану виконання запиту, тобто кроків, які необхідно виконати для виконання запиту.

Під час оптимізації запиту відбуваються спроби покращення плану виконання запиту та вибір найкращих з доступних алгоритмів (сортування, пошуку, об'єднання таблиць і т.д.) для його виконання. Після цього кроку план запиту вважається готовим до виконання.

Виконання запиту відбуваються СУБД, яка запускає скомпільований та оптимізований план запиту [26].

Сучасні СУБД при оптимізації запиту враховують не тільки сам запит, а і склад даних, збережених в базі, наявність індексів та інших допоміжних

сутностей. Крім того, в деяких системах існує можливість автоматичного створення статистичних відомостей про дані в базі, на основі яких також оптимізуються запити.

Висновки до розділу 1

Розглянуті засоби обробки даних дають уяву про усталені форми організації цього процесу в різних сферах бізнесу та досліджень. Проте, саме автоматична обробка даних буде лише частиною спектру задач, що за задумкою покладатимуться на розроблювану систему. Крім того, можливість такого використання системи буде скоріше побічним ефектом її функціоналу аніж основною ціллю її проектування та розробки в даній роботі.

Переважно розглядатимемо табличні, чи такі які можна легко привести до табличних, дані, так як структура таблиць строго визначена і дозволяє значно спростити і проєктування графічного інтерфейсу програми і відповідну реалізацію самої системи. Під час розробки системи необхідно буде врахувати всі можливі варіанти табличних даних та розробити способи спрощення їх представлення.

Зважаючи на природу неструктурованих даних, в рамках даного дослідження розглядатиметься обробка структурованих та напівструктурованих даних. Для оптимізації роботи задуманої системи та можливого спрощення розробки шляхом розділення на окремі модулі з однією функцією потенційно можна використати механізм обробки запитів перед виконанням у СУБД.

Загалом, у сфері обробки даних в основному переважають засоби, що дозволяють отримувати з великих наборів даних певні обчислені величини (суми, відсотки, прибутки і т.п.) або статистичні результати для певних досліджень чи підтримки прийняття рішень.

Зважаючи не переглянуті аналоги можна сказати що задумана система буде достатньо унікальним за своїм принципом продуктом, тому подальшу розробку можна вважати доцільною.

2 ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ

2.1 Формування функціональних вимог

Через природу системи, що полягає у монопольному використанні кожним фізичним користувачем, абстрактний Користувач у задуманої системи завжди один, тому увесь функціонал буде йому доступний і розглядатиметься в цьому розділі і надалі як такий.

Скорочений перелік функціональних вимог, що висуваються до системи наведено в таблиці 2.1.

Таблиця 2.1 – Перелік функціональних вимог до системи

Вимога	Опис	Примітка
Можливість обробки табличних даних	Система завжди може завантажувати та обробляти табличні дані за умови їх коректного виду	-
Можливість оцінки оброблюваності певним чином структурованих даних	Система може визначити наскільки і які частини даних можна обробити в певний структурований спосіб	Потенційно можна надати користувачу можливість визначати самостійно спосіб обробки
Збереження налаштувань обробки	Користувач може зберігати налаштування обробки певного формату файлів та в подальшому їх використовувати в системі	-
Портативність налаштувань обробки	Налаштування обробки зберігаються у вигляді файлів і можуть переноситись на інші машини для використання в іншій копії системи	В налаштуваннях системи можна додати визначення папок, в яких

Вимога	Опис	Примітка
		система шукатиме налаштування для швидкого доступу до збережних варіантів
Графічна організація процесу обробки даних	Користувач має мати змогу налаштувати всю або більшу частину обробки структурованих даних з допомогою графічного інтерфейсу.	-
Попередній перегляд результату	Користувач під час налаштування обробки має мати змогу попередньо переглядати вигляд результату або його частини для великих об’ємів даних.	-
Можливість налаштування обробки програмним способом	Користувач має мати змогу організувати дообробку даних, необхідну після налаштованої в графічному інтерфейсі певним алгоритмічним чином не покидаючи системи	Потенційно буде необхідна розробка API для певної мови програмування для її використання всередині
Коректна обробка неіснуючих даних	Користувач може налаштувати обробку неіснуючих значень відповідно до його потреб. За замовченням неіснуючі значення	-

Вимога	Опис	Примітка
	ігноруватимуться	
Коректна обробка та попередження користувача про некоректні значення	Користувач має бути попереджений про неконсистентні значення в даних для обробки (напр. рядок в стовпці з числами і т.п.) та налаштувати обробку таких значень. За замовченням некоректні значення ігноруватимуться в обчисленнях	-
Локальність обробки даних	Дані користувача обробляються виключно локально на його машині	-
Можливість автоматизації обробки	Користувач зможе налаштувати автоматичну обробку даних за вказаними збереженими налаштуваннями.	Для цього вказуватимуться розміщення даних (та можливо файлів налаштувань) та необхідний буде запуск системи та виклик відповідної функції

Варто зазначити що вказані вимоги під час подальшого проєктування та розробки системи можуть бути змінені або відкинуті у разі неможливості реалізації, а також можуть бути створені нові вимоги та задокументовані далі в записці.

В наступному підрозділі дані вимоги будуть розглянуті та описані у вигляді узагальнених прецедентів роботи користувача з системою.

2.2 Розгляд прецедентів роботи користувача з системою

Діаграма прецедентів, що містить весь описаний в даному підрозділі задуманий функціонал системи, знаходиться в додатку А.

2.2.1 Обробка даних

Безпосередньо обробка даних є основним призначенням розроблюваної системи. Тому вона включає в себе більшість інших прецедентів роботи користувача з системою. Наведемо в даному описі тільки базові дії з обробки даних, а решту буде описано в наступних пунктах.

Діаграму прецедентів для функціоналу обробки табличних даних зображено на рисунку 2.1.

Рисунок 2.1 – Діаграма прецедентів до функціоналу обробки даних

Детальний опис елементів діаграми прецедентів, наведеної на рис. 2.1 представлений в таблиці 2.2.

Таблиця 2.2 – Опис прецедентів обробки даних

Дія	Опис
Обробка табличних	В рамках прецеденту Користувач здійснює обробку даних,

Дія	Опис
даних	завантажених в систему. Передбачає завантаження файлу з даними в систему, налаштування необхідної обробки, попередній перегляд результату та запуск обробки
Завантаження даних	Користувач обирає в інтерфейсі системи файл(и) з даними, які необхідно обробити та має побачити у вікні даних частину завантажених даних
Налаштування обробки	Користувач, використовуючи графічний інтерфейс та/або надане програмне API налаштовує обробку завантажених даних
Налаштування через інтерфейс	Передбачає використання графічного інтерфейсу системи для налаштування обробки завантажених користувачем даних
Нал. обробки програмним способом	Передбачає використання створеного(их) користувачем скрипта для обробки завантажених даних разом чи замість налаштованої в інтерфейсі обробки (детальніше в п. 2.2.4)
Попередній перегляд результату	Користувач може бачити частковий результат обробки для перевірки правильності налаштування.
Запуск обробки	Користувач запускає обробку обраних даних налаштованим способом та отримує файл визначеного під час налаштування обробки формату

Основною частиною обробки даних з використанням системи є її налаштування користувачем у графічному інтерфейсі чи іншими описаними в даному розділі методами, решта дій несуть допоміжну роль у виконанні описаного прецеденту.

2.2.2 Збереження налаштувань обробки

Користувач має можливість після налаштування обробки файлу певного виду зберегти створені налаштування для подальшого застосування для обробки аналогічних файлів. Під час розробки системи необхідно буде продумати функціонал перевірки завантажених даних на факт можливості застосування до них певних збережених налаштувань користувача.

Діаграму прецедентів для функціоналу збереження налаштувань обробки зображено на рисунку 2.2.

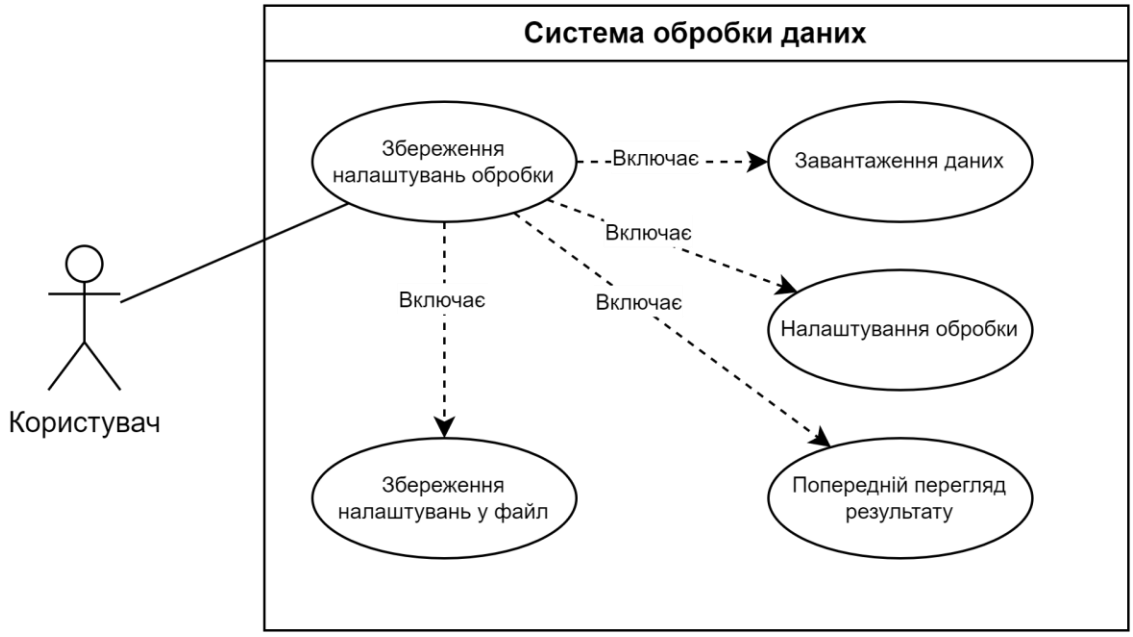


Рисунок 2.2 – Діаграма прецедентів до функціоналу збереження налаштувань обробки

Детальний опис елементів діаграми прецедентів, наведеної на рис. 2.2 представлений в таблиці 2.3.

Таблиця 2.3 – Опис прецедентів збереження налаштувань обробки

Дія	Опис
Збереження налаштувань обробки	В рамках прецеденту Користувач завантажує дані та налаштовує їх обробку, налаштування якої потім зберігає в окремому файлі для подальшого повторного

Дія	Опис
	використання в будь-якому екземплярі системи
Завантаження даних Налаштування обробки Попередній перегляд результату	Описані у таблиці 2.2.
Збереження налаштувань у файл	Користувач вказує назву та розміщення файлу, в який будуть збережені налаштування обробки, які він щойно створив

Ключовим етапом тут знову виступає первинне налаштування обробки даних, яке дає можливість зберегти та перевикористати згадані налаштування в майбутньому.

2.2.3 Перенесення налаштувань обробки

Користувач може переносити збережені раніше налаштування між машинами у вигляді файлів та використовувати їх як зазвичай.

Діаграму прецедентів для функціоналу перенесення налаштувань обробки зображено на рисунку 2.3.

Рисунок 2.3 – Діаграма прецедентів до функціоналу перенесення налаштувань обробки

Детальний опис елементів діаграми прецедентів, наведеної на рис. 2.3 представлений в таблиці 2.4.

Таблиця 2.4 – Опис прецедентів перенесення налаштувань обробки

Дія	Опис
Перенесення налаштувань обробки	В рамках прецеденту Користувач зберігає налаштування обробки, після чого переносить їх на іншу машину з встановленим екземпляром системи та використовує для обробки даних
Збереження налаштувань обробки	Описаний у таблиці 2.3.
Перенесення/копіювання файлу налаштувань	Користувач самостійно поза системою переносить файл налаштувань між машинами
Вказання розміщень для пошуку налаштувань	Користувач в інтерфейсі системи вказує розміщення на диску для пошуку файлів налаштувань обробки для системи
Вибір збережених	Користувач завантажує дані та обирає зі знайдених у

Дія	Опис
налаштувань обробки	налаштованих розміщеннях на диску файлів налаштувань необхідний для обробки завантажених даних

Ключовим етапом використання збережених налаштувань обробки виступає вказання їх розміщення на машині користувача, так як система не може самостійно визначити файл містить налаштування обробки чи дані без його завантаження.

2.2.4 Налаштування обробки програмним способом

Користувач може також інтегрувати в обробку даних власні скрипти, написані на мовах, що будуть обрані та обґрунтовані далі в роботі, застосовуючи їх до чи після обробки налаштованої безпосередньо в системі.

Діаграму прецедентів для функціоналу налаштування обробки програмним способом зображено на рисунку 2.4.

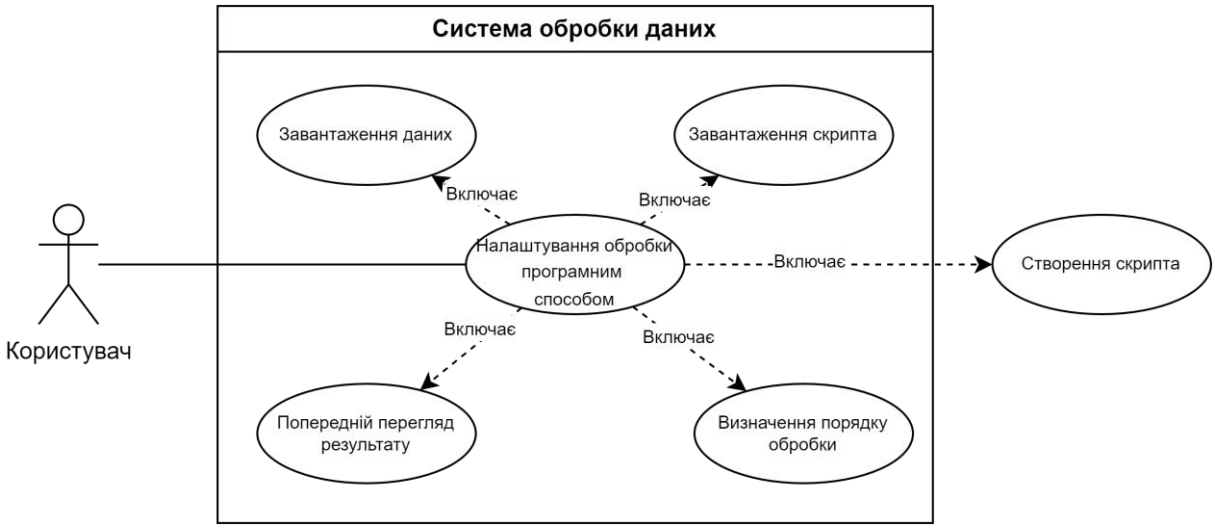


Рисунок 2.4 – Діаграма прецедентів до функціоналу налаштування обробки програмним способом

Детальний опис елементів діаграми прецедентів, наведеної на рис. 2.4 представлений в таблиці 2.5.

Таблиця 2.5 – Опис прецедентів налаштування обробки програмним способом

Дія	Опис
Налаштування обробки програмним способом	В рамках прецеденту Користувач застосовує створені скрипти для обробки завантажених даних разом чи окремо від налаштувань обробки в інтерфейсі
Завантаження даних Попередній перегляд результату	Описані у таблиці 2.2.
Створення скрипта	Користувач самостійно створює скрипт для обробки певного виду даних у відповідності до API системи та зберігає його у файл. Відбувається поза системою
Завантаження скрипта	Користувач вказує через інтерфейс системи розташування на диску, в яких слід шукати скрипти для обробки даних, після чого обирає зі знайдених системою скриптів необхідний
Визначення порядку обробки	Користувач може окрім завантаженого скрипта також налаштувати обробку в інтерфейсі системи та визначити порядок застосування обробки перед чи після скрипта(ів)

Ключовим етапом для налаштування обробки з використанням користувацьких скриптів є створення даних скриптів користувачем у відповідності до розробленого API та синтаксичних правил мови, яку він обрав для розробки.

2.2.5 Налаштування обробки некоректних даних

Користувач може окремо налаштувати обробку неіснуючих чи некоректних даних (напр: NULL-значення, відсутні значення, значення неправильного типу для стовпчика таблиці), які виявить система.

Діаграму прецедентів для функціоналу налаштування обробки некоректних даних зображено на рисунку 2.5.

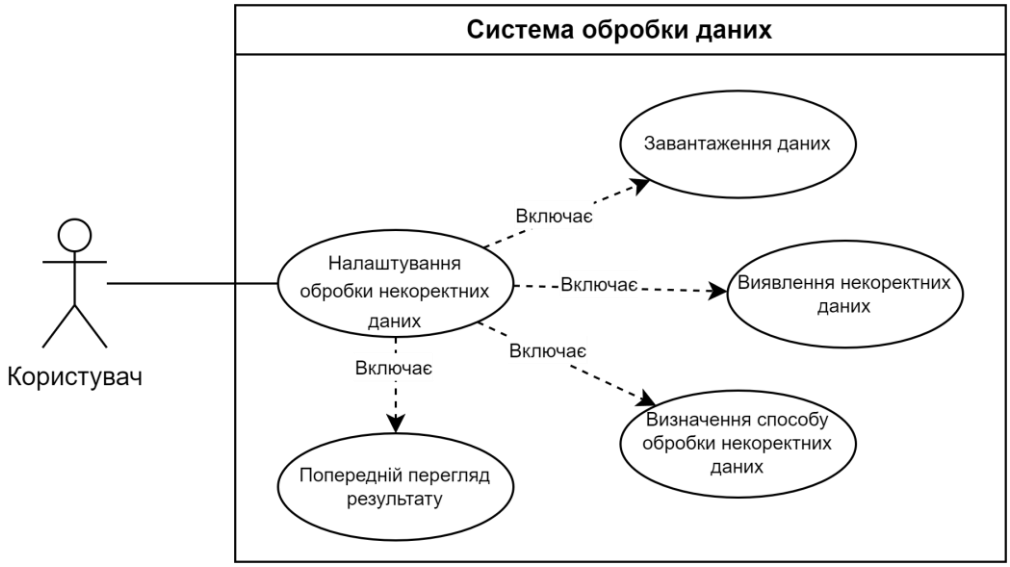


Рисунок 2.5 – Діаграма прецедентів до функціоналу налаштування обробки некоректних даних

Детальний опис елементів діаграми прецедентів, наведеної на рис. 2.5 представлений в таблиці 2.6.

Таблиця 2.6 – Опис прецедентів налаштування обробки некоректних даних

Дія	Опис
Налаштування обробки некоректних даних	В рамках прецеденту Користувач визначає які дані в певній частині чи цілому масиві завантажених в систему даних вважаються некоректними та з допомогою графічного інтерфейсу налаштовує правила їх обробки при виявленні

Дія	Опис
Завантаження даних Попередній перегляд результату	Описані у таблиці 2.2.
Виявлення некоректних даних	Користувач може за підказкою системи, при виявленні типових видів некоректних даних, чи самостійно вказати які дані вважати некоректними під час налаштування обробки
Визначення способу обробки некоректних даних	Користувач може обрати спосіб обробки некоректних даних при їх появі у завантаженому масиві. Наприклад ігнорувати, прийняти за 0 чи 1 залежно від операції і т.д.

Ключовим етапом налаштування обробки некоректних даних є визначення типів та/або значень даних, які вважатимуться некоректними при обробці.

2.2.6 Автоматизація обробки

Користувач може використовуючи збережені раніше налаштування обробки певних типів файлів налаштувати автоматизовану обробку декількох таких файлів, що знаходяться в обраній директорії на машині користувача за умови запущеної на даній машині системи та відкритого для неї доступу до обраної(их) директорії(й).

Діаграму прецедентів для функціоналу автоматизації обробки зображено на рисунку 2.6.

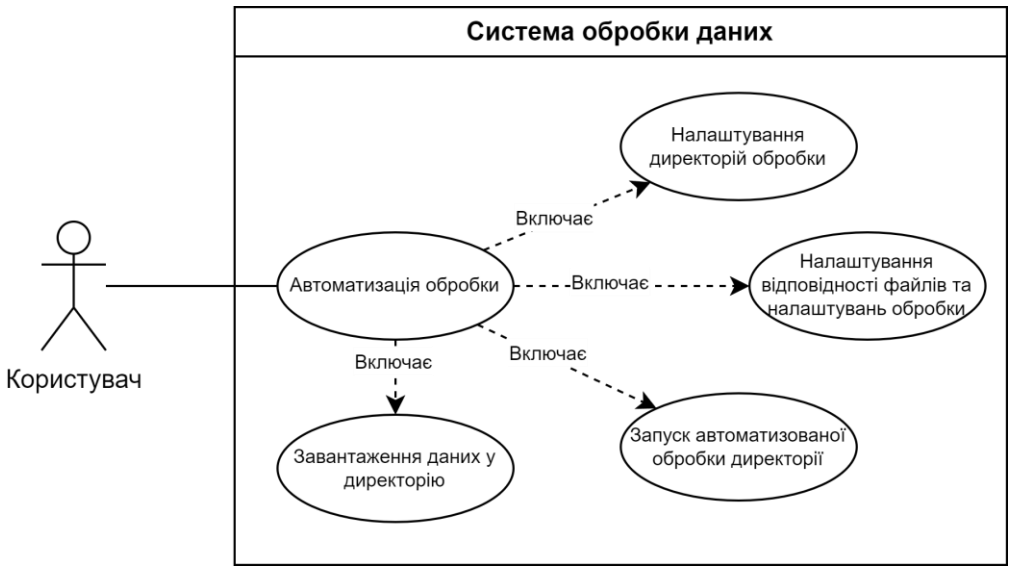


Рисунок 2.6 – Діаграма прецедентів до функціоналу автоматизації обробки

Детальний опис елементів діаграми прецедентів, наведеної на рис. 2.6 представлений в таблиці 2.7.

Таблиця 2.7 – Опис прецедентів автоматизації обробки

Дія	Опис
Автоматизація обробки	В рамках прецеденту Користувач може налаштувати автоматизовану обробку даних, що потраплятимуть у визначені директорії на диску машини, на якій працює система. Для обробки на машині має бути запущений екземпляр системи та налаштовані способи обробки даних
Налаштування директорії обробки	Користувач з допомогою графічного інтерфейсу системи вказує, в яких директоріях система має шукати дані для автоматизованої обробки та зберігати результати обробки
Налаштування відповідності файлів та налаштувань обробки	Користувач з допомогою графічного інтерфейсу визначає для різних файлів, які необхідно обробити відповідні налаштування обробки шляхом вказання

Дія	Опис
	відповідних збережених файлів налаштувань
Запуск автоматизованої обробки директорії	Користувач з допомогою графічного інтерфейсу вмикає моніторинг системою вказаних для обробки директорій
Завантаження даних у директорію	Користувач завантажує у вказані раніше директорії файли необхідного виду, вони автоматично обробляються системою і результати обробки зберігаються визначеним користувачем способом

Ключовим етапом автоматизації обробки даних з допомогою розроблюваної системи є визначення відповідності налаштувань обробки та даних або їх розміщень, до яких дані налаштування необхідно застосувати при запуску автоматизованої обробки.

2.3 Формування нефункціональних вимог

Нефункціональні вимоги, що висуваються до розроблюваної системи наведено в таблиці 2.8.

Таблиця 2.8 – Перелік нефункціональних вимог до системи

Вимога	Опис	Примітка
Дружність інтерфейсу	Інтерфейс системи має бути дружнім для користувача та слідувати сучасним контрактам дизайну та стандартам якості.	Визначається суб’єктивно
Інтуїтивність інтерфейсу	Інтерфейс системи має бути інтуїтивним та зрозумілим, за	*Інструкції та підказки

Вимога	Опис	Примітка
	неможливості для певного функціоналу мають бути присутні інструкції та підказки*	інтегровані в інтерфейс системи
Відносна швидкість роботи	Час обробки даних невеликого об’єму (50-100 рядків), налаштованої тільки в інтерфейсі без очевидно великих або нескінченних циклів має бути не більше 10 секунд	В інших випадках при складній обробці або помилках користувача має бути можливість зупинити обробку.
Точність обчислень	Обчислення з цілими числами мають бути абсолютно точними з попередженням про переповнення, числами з плаваючою комою мають мати точність до 10 знаків	-
Імутабельність оригінальних файлів	Фали з даними, завантажені користувачем мають бути незмінними з боку системи	За потреби внести зміни у вихідний файл користувач буде вимушений скористатись зовнішнім редактором
Швидкість налаштування обробки	Налаштувати обробку файлу певної структури має бути так само чи швидше за аналогічну ручну обробку файлу такої ж структури невеликого	Для складної обробки може не виконуватись, оцінюється

Вимога	Опис	Примітка
	розміру (50-100 рядків для таблиць)	суб’єктивно
Кросплатформеність	Система має бути доступною на Windows та Linux.	Рішення про реалізацію підтримки вказаних та/або інших платформ буде прийняте на етапі вибору інструментів розробки.
Керування помилками	У випадку виникнення помилок, що перешкоджають роботі системи користувач має мати можливість отримати інформацію про причину помилки та можливі способи її вирішення	Помилки мають мають логуватись у визначеному користувачем місці

Так само як і функціональні, описані нефункціональні вимоги до системи можуть бути змінені або відкинуті у разі неможливості реалізації, а також можуть бути створені нові вимоги та задокументовані в подальших розділах пояснювальної записки.

Необхідність та оцінку виконання кожної з вимог залишимо на розсуд дослідника та оцінимо після програмної реалізації системи.

Висновки до розділу 2

Описане призначення розробки та наявні обмеження об’єму реалізації у зв’язку з часовими рамками дослідження дозволяють обмежити можливі варіанти реалізації системи до нативного додатку без доступу до мережі інтернет, який містить в собі весь або частково функціонал, описаний функціональними вимогами до системи а також задовольняє всім або частині описаних у розділі нефункціональним вимогам. При цьому сформульовані вимоги допоможуть у оцінці розроблених прототипів системи системи на предмет завершеності реалізації задуманих та описаних функцій.

Описані функціональні вимоги до системи в основному зосередженні на забезпеченні реалізації інструментів для організації обробки даних в автоматизованому та ручному режимах а також у портативності налаштувань обробки для зменшення часу розробки даних налаштувань з допомогою графічного інтерфейсу системи у випадку декількох користувачів з суміжними запитами з обробки даних.

Виконання функціональних вимог перевірятиметься шляхом мануального тестування за відповідними для кожної або спільними для декількох з вимог тест-кейсами. Оцінка виконання відповідатиме проходженню чи не проходження відповідного тест-кейсу.

Описані нефункціональні вимоги в основному зосереджені на забезпеченні задовільного досвіду користувача (UX) та створенні дружнього інтерфейсу користувача (UI). Їх виконання призведе до спрощення використання системи кінцевими споживачами та, відповідно, збільшення відповідності готової розробки до її визначеного на початку розділу призначення.

Виконання нефункціональних вимог перевірятиметься шляхом мануального тестування на згаданих раніше тест-кейсах. Оцінка виконання буде надана суб’єктивно (для суб’єктивних показників, таких як дружність інтерфейсу), та об’єктивно за результатами роботи системи чи характеристиками застосованих технологій на розсуд тестувальника.

3 ПРОЄКТУВАННЯ СИСТЕМИ

3.1 Аналіз потенційних сценаріїв роботи системи

Перед проєктуванням майбутньої системи опишемо можливі сценарії її роботи, частково розглянуті у попередньому розділі з боку функціоналу, тепер з точки зору користувача та різних типів даних. Такий підхід дасть визначити основні фокуси під час проєктування та подальшої розробки системи та її критичні елементи, як видимі так і не видимі користувачу, на розробку та тестування яких необхідно буде витратити найбільше часу.

Такий підхід також дасть можливість вибірково скоротити час розробки елементів системи, що матимуть меншу важливість для користувача та основного функціоналу системи. В контексті обмеженого часу дослідження це стане важливим засобом оптимізації розробки та ключовим методом виконання якомога більшої кількості вимог до системи.

В наступних пунктах описані можливі сценарії використання розроблюваної системи Користувачем.

3.1.1 Обробка даних табличного виду

В даному сценарії користувач завантажує в програму файл з даними табличного виду. Такі дані можуть бути збережені у ряді різних форматів, наприклад:

– DAT – довільний формат даних, часто специфічний для програми, що його використовує/визначає;

– CSV – дані, розділені комами;

– TSV – дані, розділені табуляціями;

– XML – таблиця у вигляді XML-об’єктів;

– JSON – таблиця у вигляді JSON-об’єктів;

– XLS та XLSX – таблиця Excel сучасного виду та зразка Excel 2007 відповідно;

– ODS – таблиця формату OpenDocument (LibreOffice і т.п.), та інші.

Так для табличних даних нема як такого єдиного стандартного формату представлення у вигляді файлів і багато табличних редакторів використовують власні формати, підлаштовані під їх функціонал, розроблюваній системі необхідно буде перетворювати завантажений формат табличних даних у власне ж внутрішнє представлення таблиць.

Після цього користувач може з допомогою графічного інтерфейсу налаштувати обробку завантаженої таблиці. Очевидно, що необхідна буде розробка функцій агрегації декількох значень, таких як сума, добуток та ін., які можна буде застосувати до діапазону комірок завантаженої таблиці.

Користувач з допомогою функції попереднього перегляду може переконатись що налаштована обробка даних його задовольняє та запустити або зберегти її налаштування.

При запуску обробки системі необхідно буде провести відповідні налаштуванням перетворення таблиці у внутрішньому представленні, після чого конвертувати результат у обраний користувачем з доступних(го) формат файлу.

На цьому даний сценарій використання завершується.

Розглянемо питання, на які необхідно дати відповіді при створенні системи для реалізації описаного сценарію роботи (на ці та поставлені далі питання посилання виглядатимуть наступним чином: питання 3.1.1.1, де перші цифри позначають номер пункту, в якому знаходиться питання, а остання позначає номер питання у списку нижче):

1. Які формати табличних даних підтримуватимуться системою у якості вхідних даних та результатів обробки?

2. Який вид матиме внутрішнє представлення табличних даних?

3. Які функції налаштування обробки табличних даних мають бути доступними користувачу для задоволення потреб з обробки цих даних?

4. Яким чином буде реалізовуватись попередній перегляд результатів обробки?

Відповіді на перелічені питання будуть розглянуті у подальших підрозділах.

3.1.2 Обробка логів

В даному сценарії користувач завантажує в програму файл, що містить дані логів певної системи чи програми. Дані логів також зберігаються у ряді різних форматів, наприклад:

– TXT – простий текстовий файл з записами логів, розділеними довільним чином;

– JSON – набір JSON-об’єктів, що представляють собою записи логів, напівструктурований формат;

– CEF – текстовий формат логів, призначений для безпекових пристроїв та застосунків;

– CLF – текстовий формат логів, призначений для веб-серверів;

– ELF – текстовий формат логів, призначений для веб-застосунків (розширений варіант CLF), та інші [27].

Аналогічно до табличних даних, дані логів не мають єдиного стандартного формату представлення. Тому в даному випадку також необхідна буде розробка власного внутрішнього представлення логів а також засобів прямого та зворотного перетворення даних в та з даного представлення.

Після завантаження даних користувач може з допомогою графічного інтерфейсу налаштувати обробку завантажених логів. Через переважно текстову природу логів, імовірно необхідна буде реалізація функціоналу парсингу логів з вказанням роздільних символів для подальшого розділення кожного з записів на частини та налаштування обробки даних у вигляді, наближеному до обробки табличних даних, описаної в попередньому пункті.

Користувач з допомогою функції попереднього перегляду може переконатись що налаштована обробка даних його задовольняє та запустити або зберегти її налаштування.

При запуску обробки системі необхідно буде провести відповідні налаштуванням перетворення даних логів у внутрішньому представленні, після

чого конвертувати результат у обраний користувачем з доступних(го) формат файлу.

На цьому даний сценарій використання завершується.

Розглянемо питання, на які необхідно дати відповіді при створенні системи для реалізації описаного сценарію роботи:

1. Які формати логів підтримуватимуться системою у якості вхідних даних?

2. Який вид матиме внутрішнє представлення даних логів?

3. Які функції налаштування обробки даних логів мають бути доступними користувачу для задоволення потреб з обробки цих даних?

До даного сценарію також застосовується питання 3.1.1.4.

Варто зазначити, що в рамках дослідження розглядається саме обробка логів а не довільних текстових даних, так як їх умовна структурованість та існування ряду стандартів ведення логів дозволяє однозначно визначити методи, що будуть в подальшому застосовані для обробки таких даних.

3.1.3 Статистична обробка даних

В даному сценарії користувач завантажує в програму файл, що містить дані певного дозволеного системою формату і хоче провести над ними статистичний аналіз засобами обробки, що надаються системою.

Для досягнення цієї цілі користувачу необхідно надати ряд статистичних функцій для використання під час налаштування обробки даних.

До таких функцій можуть належати (всі функції застосовуються до декількох значень):

– середнє значення;

– медіана;

– мода;

– середньоквадратичне відхилення;

– мінімум та максимум;

– кількість значень, та інші.

Також можлива необхідність у використанні складніших функцій, таких як кореляція значень, формування розподілу величини, розділення на квартилі та інші, але в контексті розроблюваної системи кількість доступних функцій не є першочерговим пріоритетом.

Після налаштування обробки з обраними функціями користувач запускає обробку та отримує обчислений результат.

На цьому даний сценарій використання завершується.

Розглянемо питання, на які необхідно дати відповіді при створенні системи для реалізації описаного сценарію роботи:

1. Які статистичні функції будуть реалізовуватись в системі?

2. Яким способом буде реалізовано доступ до статистичних та взагалі функцій обробки даних у графічному інтерфейсі?

При наданні відповідей на дані питання в подальших пунктах керуватимемось в основному часовими рамками дослідження, так як статистична обробка не є базовим функціоналом системи.

3.1.4 Об’єднання та обробка даних з декількох джерел

В даному сценарії користувач завантажує в декілька файлів, що містять дані певного дозволеного системою формату. З допомогою графічного інтерфейсу, користувач може налаштувати обробку даних з декількох файлів, завантажених у систему.

В такому випадку внутрішнє представлення даних у системі повинне вміти розрізняти дані з різних джерел та обробляти їх відповідно.

Після налаштування, користувач запускає обробку та отримує результат.

На цьому даний сценарій використання завершується.

Розглянемо питання, на які необхідно дати відповіді при створенні системи для реалізації описаного сценарію роботи:

1. Яким чином буде організований інтерфейс для забезпечення роботи з декількома джерелами одночасно?

2. Яким чином розрізнятимуться у внутрішньому представленні даних системи дані з різних джерел?

При наданні відповідей на дані питання в подальших пунктах також керуватимемось в основному часовими рамками дослідження, так як описаний функціонал є побічним і не входить до функціональних вимог.

3.1.5 Збереження налаштувань обробки

В даному сценарії користувач завантажує в систему дані певного дозволеного системою формату та проводить налаштування їх обробки, після чого зберігає дані налаштування для подальшого використання.

На цьому даний сценарій використання завершується.

Для реалізації цього сценарію необхідно буде визначитись з способом серіалізації налаштувань а також організувати перевірку можливості застосування певних збережених налаштувань обробки до обраного файлу для забезпечення коректної роботи системи.

Розглянемо питання, на які необхідно дати відповіді при створенні системи для реалізації описаного сценарію роботи:

1. Який формат файлів використовуватиметься для збереження налаштувань обробки?

2. Яким чином відбуватиметься перевірка сумісності налаштувань обробки та завантажених даних?

Зазначимо, що внутрішнє представлення налаштувань обробки необхідно буде спроєктувати таким чином, щоб його було легко серіалізувати та десеріалізувати відповідно у файл налаштувань та з нього.

3.1.6 Автоматизація обробки даних

В даному сценарії користувач використовує раніше збережені налаштування обробки для автоматизованої обробки файлів з даними дозволених для використання в системі форматів.

Для цього користувачу необхідно буде вказати в системі розміщення, в яких знаходитимуться файли для обробки а також відповідність файлів та налаштувань обробки, після чого вказати розміщення для збереження результатів обробки та запустити моніторинг системою вказаних розміщень на диску.

На цьому даний сценарій використання завершується.

Розглянемо питання, на які необхідно дати відповіді при створенні системи для реалізації описаного сценарію роботи:

1. Яким способом буде реалізована перевірка директорій, вказаних користувачем на наявність файлів для обробки?

2. Яким чином встановлюватиметься відповідність між файлами з даними та налаштуваннями обробки?

Також помітимо, що для зменшення прямої взаємодії з файловою системою користувача принцип роботи даного функціоналу може бути змінений під час розробки в бік ручного завантаження файлів, для яких певним чином автоматично визначатимуться налаштування обробки.

3.1.7 Використання алгоритмічних реалізацій обробки даних

В даному сценарії користувач завантажує дані дозволеного для використання в системі формату а також власний скрипт для повної чи часткової їх обробки, написаний у відповідності до API розроблюваної системи. Після цього користувач може також провести налаштування обробки в графічному інтерфейсі, та, в разі виконання цієї дії, визначити порядок обробки даних (налаштування в інтерфейсі та скрипт). Після цього користувач запускає обробку та отримує результат.

На цьому даний сценарій використання завершується.

Розглянемо питання, на які необхідно дати відповіді при створенні системи для реалізації описаного сценарію роботи:

1. Яку або які мови програмування будуть використовуватись для написання користувацьких скриптів?

2. Якою буде структура API системи та до якого функціоналу матимуть доступ його користувачі?

Зазначимо, що потенційно даний функціонал буде відкинутий під час розробки через те, що система в основному орієнтується на нетехнічних користувачів, які більш знайомі з графічними інтерфейсами програм. Крім того, реалізація функціоналу системи з урахуванням необхідності використання її API а також інтеграція зовнішніх вихідних файлів з кодом значно ускладнить розробку системи на всіх етапах.

3.2 Вхідні та результуючі дані

Розглянуті в попередньому підрозділі сценарії використання системи дозволяють сформувати уявлення про типи даних, з якими зможе працювати проєктована система.

Визначення та закріплення типів даних, придатних до обробки майбутньою системою допоможе обмежити велику кількість варіантів реалізації функціоналу лише актуальними елементами. Крім того, це значно спростить реалізацію алгоритмічної частини системи, так як обробка різних типів даних потенційно потребує різних програмних реалізацій одного і того ж функціоналу, що варто звести до мінімуму враховуючи часові та ресурсні обмеження розробки.

В наступних пунктах описані типи вхідних та вихідних даних майбутньої системи. Також будуть зазначені деталі реалізації їх обробки, які необхідно буде врахувати в процесі розробки системи.

3.2.1 Вхідні дані

Основним типом даних, що оброблятиметься в системі будуть табличні дані, тому спочатку визначимо, які формати табличних даних прийматимуться на вхід системи.

Одним з найпростіших форматів табличних даних (і взагалі структурованих даних) є CSV. Даний формат описується запропонованим стандартом RFC 4180 [28], проте як такий не є строго стандартизованим, наприклад замість ком можуть використовуватись пробіли, табуляції, крапки з комою. При чому в таких випадках застосунки все одно визначають формат файлу як .csv.

Крім того, рядок заголовку нічим не відрізняється від подальших записів (csv – plain-text формат), тому про його наявність слід запитувати користувача при завантаженні даних.

Також варто зазначити, що для представлення у форматі CSV табличних даних у яких більше 2 вимірів необхідно ввести певний рівень надмірності даних, але конвертація даних в CSV формат правильного виду в рамках дослідження покладається на користувача (наприклад, з допомогою таких інструментів як OpenRefine [29]).

Зазначимо також основні правила формату CSV:

– кожен запис займає один рядок, але роздільником записів може виступати як перенесення рядка (ASCII/LF=0x0A) так і повернення каретки разом з перенесенням рядка (ASCII/CRLF=0x0D 0x0A). Крім того, поля запису можуть мати перенесення рядків всередині, тому фізично запис може займати декілька рядків;

– поля запису розділені комами (або табуляціями, або крапками з комами і т.д);

– початкові та кінцеві пробіли біля розділювачів полів інгноруються;

– поля з символами, що є розділювачами полів, повинні екрануватись подвійними лапками;

– поля з подвійними лапками повинні екрануватись подвійними лапками, а самі подвійні лапки дублюватись (компанія ”Перемога” перетворюється на поле ”компанія ””Перемога”””);

– поля, що містять перенесення рядків повинні екрануватись подвійними лапками;

– поля, що містять початкові та кінцеві пробіли, які необхідно зберегти, повинні екрануватись подвійними лапками;

– поля в принципі дозволено завжди екранувати подвійними лапками;

– перший запис CSV файлу може бути заголовком, що містить імена колонок подальших записів, але це не обов’язково і ніяк не визначається програмно [28, 30].

Бачимо, що навіть для відносно простого та обмеженого формату, CSV має ряд особливостей, які необхідно буде врахувати в реалізації системи.

Так як реалізація взаємодії зі складнішими та пропрієтарними форматами табличних даних займе багато часу, а також враховуючи те, що більшість сучасних табличних редакторів (Excel, Goggle Sheets, LibreOffice Calc) мають опцію збереження файлу у форматі CSV та те, що основне призначення системи все таки у вторинній обробці даних, приймемо рішення в поточній ітерації реалізації системи використовувати лише CSV як формат вхідних табличних даних.

Таким чином, відповідь на питання 3.1.1.1: табличні дані у форматі CSV.

Також розглянемо можливість обробки в системі файлів логів.

Логи роботи певних систем чи програм представляють собою одновимірну послідовність записів стандартизованого виду [31], що відображають стан системи чи певні події всередині чи поза нею. Принципово, до логів можна відноситись як до одновимірної таблиці.

Серед безлічі форматів логів існують структуровані, напівструктуровані та неструктуровані. Так як неструктуровані логи програмно обробити важко, розглядатимемо для обробки в системі лише перші два види логів [32].

Також помітимо, що більшість структурованих форматів логів використовують певні символи-розділювачі для представлення різних частин запису. Такий підхід схожий на організацію формату CSV з довільними розділювальними символами, а так як логи зберігаються в основному в текстовому вигляді, можна реалізацію їх обробки та представлення сумістити з обробкою та представленням даних у форматі CSV, додаючи лише опцію користувачу вказати роздільний символ та символ, що розділює записи.

Тоді відповідь на питання 3.1.2.1: струтуровані текстові формати, що використовують роздільні символи для виділення полів запису.

3.2.2 Результати обробки

Так як у попередньому пункті ми звели формати вхідних даних системи до варіанцій CSV, оберемо цей самий формат і для результатів обробки, а саме записи з даними, представленими простим текстом, розділені символами CRLF та з комою в якості роздільнго символу полів запису.

На виході матимемо однозначно визначений налаштуваннями обробки результат у вигляді CSV файлу.

3.2.3 Збережені налаштування обробки

У якості формату для збереження налаштувань обробки оберемо plain-text файл з власним розширенням .pidps (Process It Data Processing Settings). З огляду на те, що користувач налаштовує обробку даних покроково, зберігати налаштування обробки ми також можемо у вигляді покрокових дій над даними.

Приблизний вміст файлу налаштувань обробки даних для розроблюваної системи зображено на рисунку 3.1.

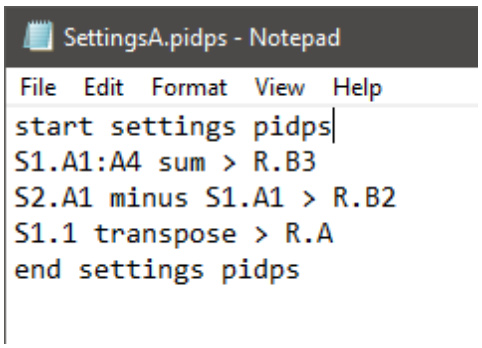


Рисунок 3.1 – Приблизний вміст файлу налаштувань обробки

Помітимо при цьому, що в процесі розробки системи вид команд та позначення розміщень даних в даному файлі можуть змінитись через особливості реалізації, але основна ідея та структура залишиться незмінною, а саме (відповідь на питання 3.1.5.1):

1. Початок з певного зарезервованого виразу, що позначає початок файлу налаштувань (за його відсутності чи невірності система не сприйматиме файл налаштувань як коректний);

2. Команди для обробки даних, по одній на рядок, кожна з яких містить операнд(и), дію над ними та розміщення результату цієї дії;

3. Кінець зарезервованим виразом, дзеркальним до початкового.

Така структура файлу налаштувань допоможе однозначно зберігати користувацькі налаштування обробки даних у відносно простому та портативному форматі, що матиме навіть деяку міру читабельності.

3.3 Внутрішні представлення даних

Так як вирішено використовувати для обробки лише дані CSV-подібного формату, значно спрощується реалізація внутрішнього представлення даних.

Розглянемо детально структуру даних, з якою працюватиме система.

Розглянемо чотири варіанти реалізації двовимірних таблиць:

1. Об’єкт, що містить масив комірок з координатами;

2. Об’єкт, що містить масив рядків, кожний з яких містить масив комірок з координатами (по суті масив записів CSV);

3. Об’єкт, що містить масив стовпців, кожний з яких містить масив комірок з координатами;

4. Об’єкт, що містить масиви рядків та стовпців, які в свою чергу мають посилання на відповідні комірки, що до них входять.

Перший варіант найпростіший в реалізації але значно ускладнить розробку та сповільнить роботу функцій обробки даних, яким необхідно буде взаємодіяти одразу з цілими рядками чи стовпцями.

Варіанти 2 та 3 позбуваються необхідності у зберіганні координат комірок, так як їх позиція в масиві відповідно рядка чи стовпця слугуватиме таким ідентифікатором, але знову ж обидві реалізації ускладнюють роботу зі стовпцями та рядками відповідно.

Нарешті, четвертий варіант хоча і міститиме багато надлишкових даних, допоможе під час розробки системи значно спростити реалізацію функцій обробки даних як по рядкам, так і по стовпцям та окремим коміркам.

Оберемо для реалізації внутрішнього представлення даних варіант 4 (відповідь на питання 3.1.1.2 та 3.1.2.2). Крім того, для розрізнення джерел даних при завантаженні декількох файлів одночасно внутрішнє представлення додатково матиме поле ідентифікатор джерела даних (відповідь на питання 3.1.4.2).

Так як обрані формати для вхідних та вихідних даних ідентичні, внутрішні представлення їх також збігатимуться.

Структуру відношень між внутрішніми елементами системи представлено на концептуальній діаграмі класів (див. Додаток Б).

3.4 Вибір та обґрунтування алгоритмів обробки даних

В таблиці 3.1 наведені функції обробки даних, що плануються до реалізації (відповідь на питання 3.1.1.3, 3.1.2.3 та 3.1.3.1).

Таблиця 3.1 – Перелік функцій для обробки даних

Функція	Опис	Примітка
Елементарні арифметичні операції	Додавання, віднімання, множення та ділення	Застосовні до комірок та діапазонів
Переміщення рядка	Переміщення рядка в іншу позицію	
Переміщення стовпця	Переміщення стовпця в іншу позицію	
Сума	Сума числових значень певного діапазону числових даних	
Середнє значення	Середнє значення певного діапазону числових даних	
Медіана	Медіана певного діапазону числових даних	У випадку парної кількості значень повертається перша з медіан
Мінімум	Мінімальне значення певного діапазону числових даних	
Максимум	Максимальне значення певного діапазону числових даних	
Кількість значень	Кількість непустих значень у певному діапазоні	
Транспонування	Перетворення рядка на стовпець чи навпаки або транспонування певної	

Функція	Опис	Примітка
	прямокутної області даних	

Так як система розрахована на обробку даних, які б інакше за відносно прийнятний час могла б обробити людина, необхідність у реалізації будь-яких алгоритмів окрім прямих для вказаних функцій відпадає. Прямі алгоритми для описаних функцій елементарні, тому описувати їх тут не будемо. Оптимізація процесу обробки даних в системі може бути фокусом подальших досліджень.

Також зазначимо, що хоча описаний набір функцій далеко не вичерпний, з його допомогою можна провести велику кількість перетворень над даними, тому в контексті даного дослідження вважатимемо його достатнім.

3.5 Вибір та обґрунтування технологій розробки системи

Для розробки системи була обрана мова програмування C# з фреймворком .Net 8.0 (LTS) підтримуваним Microsoft, а також графічну підсистему WPF.

C# є високорівневою мовою програмування, яка реалізує декілька парадигм. Вона є статично та сильно типізованою, підтримує імперативне та декларативне програмування, функціональне, об’єктно-орієнтоване та компоненто-орієнтоване програмування а також механізм шаблонізації об’єктів (Generics) [33].

Вибір C# та WPF разом був зумовлений їх повною сумісністю та простотою взаємодії між кодом на C# та графічним інтерфейсом, описаним на XAML для WPF.

WPF має недолік, який полягає в тому, що він доступний лише для операційної системи Windows і відповідно при його використанні неможлива реалізація кросплатформеності застосунку. У зв’язку з часовими обмеженнями дослідження, наявним досвідом роботи з WPF та побічним характером вимоги до кросплатформеності застосунку по відношенню до об’єкта дослідження дану вимогу прийнято рішення відкинути.

Незважаючи на вказаний недолік WPF, код на C#, що буде реалізовувати обробку та збереження даних буде цілком портативним та з невеликими доробками його можна буде використати з будь-яким іншим засобом графічного представлення, інтегрованим в .Net інфраструктуру (наприклад MAUI) для реалізації системи на інших платформах. Проте, як вже зазначалось, дана доробка знаходиться поза межами поточного дослідження.

3.6 Вибір програмного забезпечення для розробки системи

Для розробки системи була обрана Microsoft Visual Studio 2022 Community Edition [34].

Visual Studio – це IDE, розроблене компанією Microsoft. Дане середовище розробки було обрано для використання з наступних причин:

– підтримка C#, WPF та .Net «з коробки» (є рідною IDE від Microsoft для цих технологій);

– швидке створення шаблонних проєктів інфраструктури .Net, зокрема проєкту з використанням WPF на C#;

– наявність інтегрованих інструментів збірки та дебагу проєктів на C# з WPF;

– вбудований контроль версій через Github та Git;

– функціонал інтелектуального завершення коду залежно від контексту, що допомагає у оптимізації часу розробки;

– синтаксичний аналіз та перевірка коду під час написання, що дозволяє значно швидше виявляти деякі помилки без запуску програми;

– гнучкі налаштування редактора коду та вікна програми для вибору бажаного стилю IDE.

Всі ці та багато інших особливостей Visual Studio доступні користувачам одразу після встановлення (деякий інструментарій необхідно обрати для встановлення під час інсталяції, але також можна дозавантажити пізніше) навіть

користувачам безкоштовного Community Edition, на яке накладаються наступні технічні обмеження:

– робота в команді до 5 розробників;

– доступні тільки unit-тести (можна використовувати зовнішні інтсрументи);

– відсутня можливість використовувати Visual Studio в якості сервера Azure DevOps для контролю версій (окремий такий наявний сервер використовувати можна);

– урізаний функціонал зі створення віртуальних середовищ виконання програм.

А також наступні ліцензійні обмеження:

– заборонено використовувати для виконання замовлень з розробки ПЗ

– заборонено використовувати підприємствам з більше ніж 1 мільйоном доларів США у річному прибутку (включаючи підпорядковані компанії) або маєте 250 або більше комп’ютерів/користувачів;

– дозволено використовувати в якості окремого розробника якщо ви не виконуєте зовнішні замовлення з розробки ПЗ;

– дозволено використовувати для проєктів з відкритим вихідним кодом;

– дозволено використовувати навчальним закладам у освітніх цілях та в аудиторіях;

– дозволено використовувати підприємствам, що мають одночасно 5 або менше розробників (що працюють у Visual Studio), за умови що підприємство є власником розроблюваного ПЗ і продає його самостійно (не виконує замовлення більших підприємств які матимуть право власності на розроблене ПЗ) [35].

Очевидно, ліцензійні обмеження на нас не накладаються, а технічні критично не впливають на розробку та тестування системи, тому вибір Microsoft Visual Studio 2022 Community Edition вважаємо обґрунтованим.

Для організації контролю версій системи був обраний онлайн сервіс GitHub[36] та утиліта Git[37].

GitHub дозволяє зручно та швидко створювати та управляти репозиторіями вихідного коду а також легко інтегрується в Visual Studio як remote репозиторій. Крім того, дослідник має досвід роботи як з Git так і з GitHub, тому вибір системи контролю версій вважаємо обґрунтованим.

3.7 Вибір архітектури системи

Задумана реалізація системи представляє собою комп'ютерну програму без доступу в інтернет, яка працює з локальними файлами та файловою системою користувача.

Розглянемо переваги розробки локальної комп'ютерної програми для даного дослідження:

– нема необхідності у доступі до мережі інтернет, що в умовах сучасного світу може стати перевагою над більш залежними від мережі програмами;

– відсутні затримки, пов'язані з доступом до інтернету, хоча швидкість роботи системи в такому випадку залежить від наявних ресурсів машини користувача;

– більше контролю над даними для користувача, так як вони не відсилаються на зовнішні сервери для обробки;

– проста дистрибуція програми через один із магазинів програм або навіть as-is у вигляді завантаження з сайту;

– відсутня вартість підтримки серверів та мережевої інфраструктури, необхідних для роботи типових веб-додатків;

– простіше проєктування та розробка так як нема потреби враховувати особливості серверів, обмін даними з ними і т.д. [38].

Так як розроблювана система призначена для локального використання кожним користувачем окремо для задоволення особистих потреб з обробки даних вважаємо що форма локальної комп'ютерної програми без доступу в інтернет підходить для реалізації системи.

Враховуючи локальність та відсутність потреби у масштабуванні системи, прийнято рішення про використання монолітної архітектури програми. Багато з них не є релевантними для відносно невеликих локальних програм а також для випадку, коли над розробкою працює 1 людина.

Крім того, локальні комп’ютерні програми в принципі складно реалізувати в будь-якому іншому вигляді через необхідність наявності всіх елементів програми для її коректної роботи та неможливості «підвантажити» щось, чого не вистачає, під час роботи.

Переваги ж монолітної архітектури системи у випадку нашої розробки наступні:

– швидкість розробки, все знаходиться в одному проєкті та репозиторії;

– простота тестування, нема потреби тестувати всі частини окремо;

– вартість, хоча і не застосовна до даного дослідження метрика, значно зменшується через попередні два пункти, а також локалізацію всіх елементів системи.

Зважаючи на викладене вище, вважаємо вибір монолітної архітектури для розроблюваної системи обґрунтованим та доцільним.

3.8 Проєктування процесів системи

Для опису процесів, що відбуваються в системі, а також основних принципів її роботи та взаємодії з користувачем скористаємось інструментом UML діаграм, які допоможуть в повній мірі описати роботу розроблюваної системи з різних боків.

Перелік створених діаграм:

– діаграма потоків даних 0-го рівня для функції ручного налаштування обробки даних (див. додаток В);

– блок-схема алгоритму роботи системи при ручному налаштуванні обробки даних (див. додаток Г);

– блок-схема алгоритму роботи системи при автоматизованій обробці даних (див. додаток Д);

– діаграма активності системи при ручному налаштуванні обробки даних (див. додаток Е).

Розроблені діаграми описують майбутній функціонал системи з різних боків, що допоможе в його подальшій розробці.

Відповідь на питання 3.1.5.2 – система розглядатиме наявність використаних у файлі налаштувань діапазонів даних у вхідному файлі. Відповідно, у разі їх несумісності користувач буде про це попереджений та налаштування обробки не завантажені при прямій обробці або не прив’язані до файлу при налаштуванні автоматизованої обробки даних.

3.9 Проєктування інтерфейсів користувача

3.9.1 Опис користувачів

Всі можливі користувачі системи мають однакову роль – виконання перетворення даних в системі. Тому далі користувач – будь хто з доступом до системи.

Користувач може завантажувати в систему дані, в т.ч. з декількох джерел, налаштовувати обробку завантажених даних та зберігати в себе у файловій системі отриманий результат.

Також користувач може зберігати налаштування та алгоритм обробки для використання його в подальшому на таких самих файлах чи файлах подібної структури (якщо збережений алгоритм можливо до них застосувати).

Цільова аудиторія системи знайома з офісними програмами різних виробників, таких як Microsoft та Google, також можливо з рішеннями з відкритим кодом, такими як LibreOffice та інші.

3.9.2 Вибір стилю інтерфейсів користувача

Враховуючи специфіку цільової аудиторії а також особисті вподобання автора, оберемо стиль інтерфейсів користувача, схожий на реалізований в нових версіях MS Office.

Зокрема, використаємо наступні особливості дизайну продуктів Microsoft:

– плоскі елементи з фокусом на мінімалізм;

– кольорова гама, що складається зі світлого фону та контрастуючих тексту, іконок та стрічок інструментів;

– стрічка інструментів з закладками (за їх необхідності в реалізації).

Інтерфейс програми буде чітко розділений на робочу зону з полем вхідних даних та полем попереднього перегляду обробки (відповідь на питання 3.1.1.4) та зону меню та інструментів згори вікна, що міститиме всі текстові меню системи а також стрічку інструментів з піктограмами для кожної з імплементованих функцій обробки даних (відповідь на питання 3.1.3.2).

3.9.3 Опис основних видів системи

В розроблюваній системі передбачаються наступні види, доступні користувачу:

– головне вікно;

– вікно налаштувань;

– вікно налаштувань автоматизованої обробки.

Головне вікно міститиме елементи інтерфейсу для роботи з даними та доступ до меню системи.

Вікно налаштувань міститиме налаштування системи, такі як роздільний символ CSV файлів та інші.

Вікно налаштувань автоматизованої обробки міститиме елементи інтерфейсу для налаштування обробки файлів у автоматизованому режимі.

В таблиці 3.2 відображені елементи, які будуть мітитись у перелічених видах системи.

Таблиця 3.2 – Приналежність елементів інтерфейсу до видів системи

Вид системи	Елементи інтерфейсу
Головне вікно	– робоче поле з вхідними даними; – поле попереднього перегляду результатів обробки; – стрічка інструментів налаштування обробки; – рядок меню.
Вікно налаштувань	– стовпчик категорій налаштувань (опціонально); – область розміщення налаштувань системи.
Вікно налаштування автоматизованої обробки	– робоче поле з завантаженими файлами – робоче поле з обраними файлами налаштувань обробки; – стрічка інструментів налаштування автоматизованої обробки; – рядок меню.

Загалом описані види системи наслідуватимуть близький до продуктів серії MS Office, як вже зазначалось в пункті 3.7.2 про стилістику, спосіб організації елементів інтерфейсу.

3.9.4 Макети інтерфейсів системи

Розглянемо розроблені макети інтерфейсів системи у відповідності до опису видів системи, а також частково опису функціоналу.

Макет головного вікна системи зображено на рисунку 3.2.

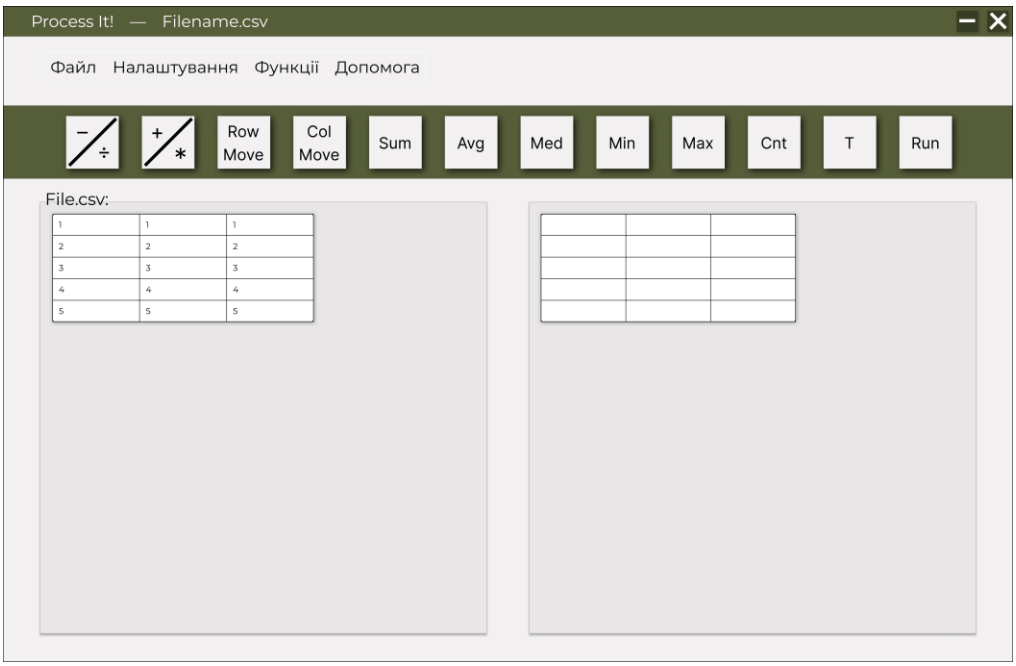


Рисунок 3.2 – Макет головного вікна програми

На даному макеті вікна бачимо задуманий вигляд рядка меню, стрічки інструментів та робочих зон системи. Крім того, наведемо на рисунку 3.7.4.2 варіант головного вікна системи при роботі з декількома вхідними файлами одночасно (відповідь на питання 3.3):

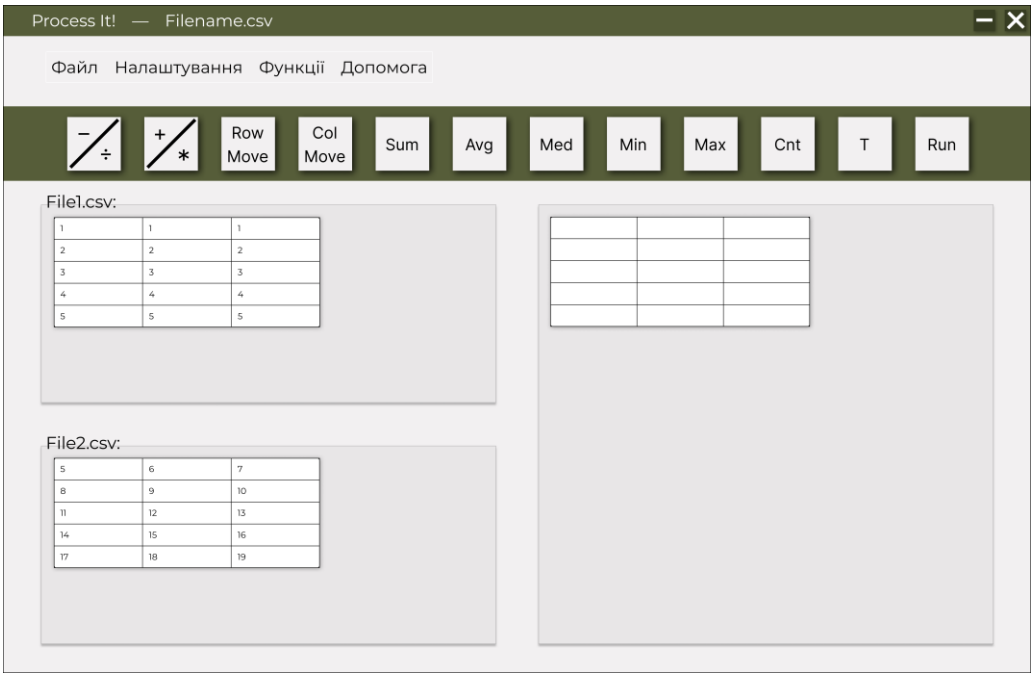


Рисунок 3.3 – Макет головного вікна програми при завантаженні декількох вхідних файлів

Також наведемо макети для вікна налаштувань (рисунок 3.4) та вікна налаштування автоматизованої обробки (рисунок 3.5):

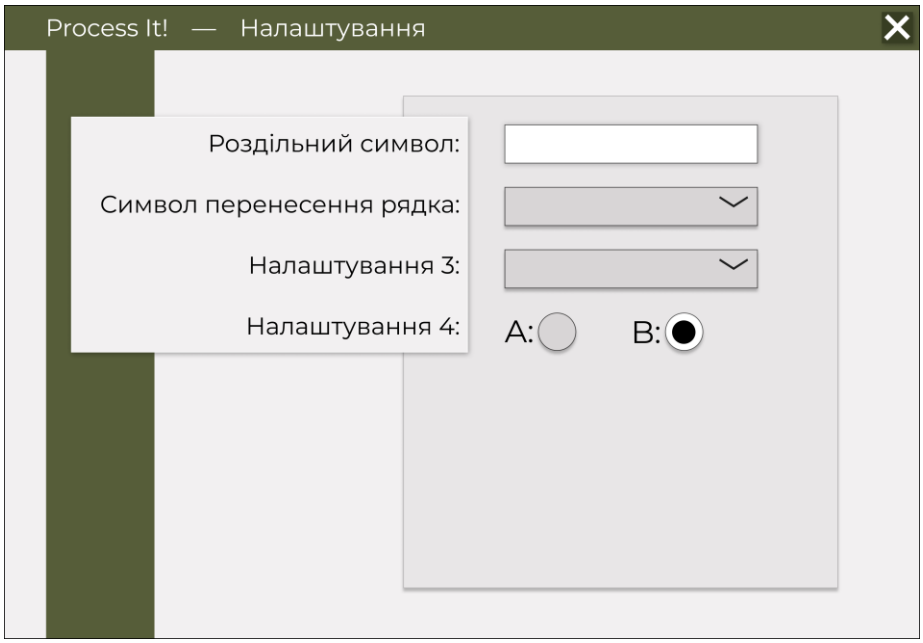


Рисунок 3.4 – Макет вікна налаштувань

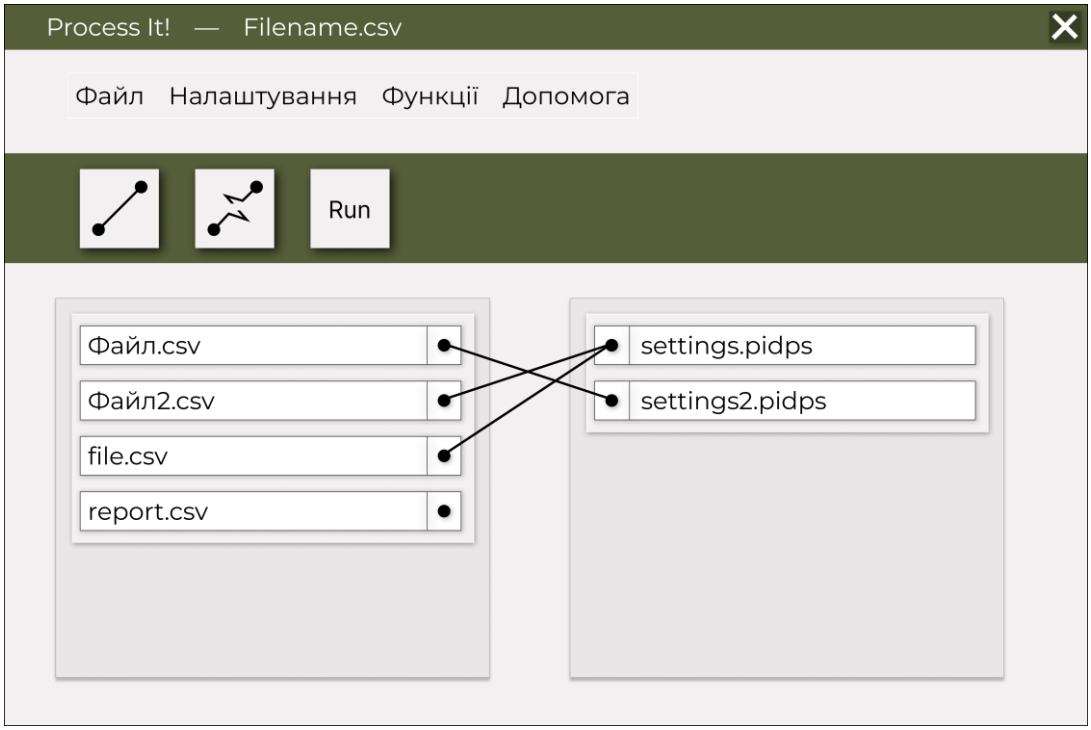


Рисунок 3.5 – Макет вікна налаштування автоматизованої обробки

Помітимо, що в процесі розробки інтерфейсів системи був розроблений відмінний від описаного вище спосіб організації автоматизованої обробки даних, який не потребує прямої взаємодії розроблюваної програми з файловою системою користувача без його участі та надає графічні засоби налаштування такої обробки, що є кращим варіантом враховуючи філософію системи.

Так як про директорії більше мова не йде, питання 3.1.6.1 відпадає.

Відповідь на питання 3.1.6.2 – відповідність встановлюватиметься з допомогою інструментів графічного інтерфейсу, наведених на стрічці інструментів на рис. 3.7.4.3, де 1 інструмент – призначення зв’язку дані-налаштування, 2 інструмент – розірвання призначеного зв’язку дані-налаштування.

Висновки до розділу 3

Проведені в розділі дослідження дозволили визначитись з основними характеристиками майбутньої системи, зокрема:

– обрані технології та програмні засоби розробки: C#, .Net, WPF, Visual Studio;

– визначений цільовий функціонал системи: обробка CSV-файлів різного формату, реалізація функцій з таблиці 3.1, налаштування обробки декількох файлів одночасно, налаштування автоматизованої обробки файлів;

– спроєктовані основні процеси системи (див. Підрозділ 3.6);

– спроєктовані основні інтерфейси системи (див. Підрозділ 3.7);

– надані відповіді на питання з підрозділу 3.1, що допомогли визначитись з шляхом реалізації системи, окрім питань 3.1.7.1 та 2 (прийняте рішення відкинути функціонал використання користувацьких скриптів, розроблених відповідно до API системи).

Проведене проєктування дозволяє перейти до етапу розробки системи з визначеними цілями та очікуваннями від готової системи.

4 СТРУКТУРНА СХЕМА СИСТЕМИ

4.1 Складові частини системи

На першому етапі розробки системи визначимо її частини, які будемо далі в певному вигляді реалізовувати.

До складових частин системи віднесемо наступні:

– інтерфейс користувача;

– модуль завантаження даних з файлів;

– модуль інтерпретації алгоритму обробки;

– модуль конвертації файлів налаштувань обробки;

– модуль автоматизованої обробки файлів;

– модуль допомоги користувачу.

Помітимо, що хоча в даному контексті частини системи ми назвали «модулями», вони не є окремими незалежними елементами, а виступають значущими внутрішніми частинами розроблюваної системи в її монолітній архітектурі.

В таблиці 4.1 наведений короткий опис кожної з зазначених частин системи.

Таблиця 4.1 – Опис складових частин системи

Частина системи	Опис
Інтерфейс користувача	Даний модуль забезпечує взаємодію користувача з системою та передачу даних про завантажувані в систему файли іншим модулям, залежно від обраного користувачем функціоналу
Модуль завантаження даних з файлів	Даний модуль забезпечує перетворення завантажених користувачем в систему файлів у внутрішнє представлення даних системи для подальшої їх обробки
Модуль інтерпретації	Даний модуль отримує з інтерфейсу послідовність налаштувань обробки системи, що визначає користувач,

Частина системи	Опис
алгоритму обробки	та перетворює її на алгоритм обробки даних, який використає система для проведення цієї обробки над внутрішнім представленням даних
Модуль конвертації файлів налаштувань обробки	Даний модуль забезпечує перетворення з налаштованої користувачем обробки даних у файл налаштувань обробки при збереженні налаштувань та навпаки при використанні збережених налаштувань для обробки даних користувачем
Модуль автоматизованої обробки файлів	Даний модуль використовує надані йому у вигляді внутрішнього представлення дані та інтерпретовані налаштування обробки для проведення автоматизованої обробки даних, налаштованої користувачем
Модуль допомоги користувачу	Даний модуль забезпечує користувача підказками до використання функціоналу системи і як такий з іншими частинами системи напряму не взаємодіє

Описані складові частини системи будуть розроблені у якості дочірніх проєктів основного рішення системи в IDE Visual Studio та використані як її складові частини у коді. Приклад роботи частини описаних модулів можна побачити на діаграмі послідовності для функціоналу автоматизованої обробки даних (див. додаток Ж).

4.2 Структурна схема системи

Структурна схема системи, що показує взаємодію між описаними у попередньому підрозділі складовими частинами системи наведена у додатку И.

Висновки до розділу 4

Визначені основні складові частини та розроблена структурна схема системи дозволять у процесі розробки концентрувати увагу на основних елементах та спростити процес прийняття рішень про побудову ієрархії об’єктів програми. Крім того, зазначення основних частин системи дозволить нам, за потреби, відкинути побічні, які не увійшли до структурної схеми але були задумані як частини системи раніше.

Також помітимо, що функціонал визначених модулів системи майже напряму відповідає деяким із зазначених у розділі 2 функціональним вимогам. На виконання функціональних, а також нефункціональних вимог буде звертатись увага при безпосередній розробці системи для забезпечення відповідності спроєктованої системи реальній.

Крім того, виконання чи не виконання поставлених вимог до системи також буде детально розглянуте у розділі 6 (тестування системи), для оцінки відповідності результату розробки поставленим цілям дослідження.

5 РОЗРОБЛЕННЯ СИСТЕМИ

5.1 Обрана методика розробки

Для вибору методології розробки розглянемо наступні факти:

– вимоги до розроблюваної системи є встановленими та гарантовано незмінними (принаймні ззовні);

– нема необхідності у розподілі обов’язків між членами команди (розробник один);

– масштаб розробки відносно невеликий;

– часові рамки дослідження обмежені досить строго.

На основі зазначених фактів можемо зробити висновок, що для даного проєкту доцільно буде використати методологію водоспаду (англ. Waterfall), яка є лінійною, простою, однозначно визначеною на кожному кроці та чудово працює в середовищі без різких змін вимог до продукту [39].

Частково ми вже пройшли декілька етапів даної методології, а саме:

– встановлення вимог до системи (функціональних та нефункціональних);

– проєктування системи.

Тепер перед нами наступні два етапи методології:

– імплементація системи;

– тестування системи.

Помітимо, що так як розробка не є комерційною та не потребує використання веб серверів, типовий кінцевий етап розгортання та підтримки системи в рамках даного дослідження буде пропущений.

Посилання на вихідний код розробленої системи можна знайти в Додатку К.

5.2 Зміни, внесені до спроєктованої архітектури системи

Під час розробки системи кардинальних змін у описаних на попередніх етапах дослідження елементах системи, способах їх взаємодії та її загальну архітектуру внесено не було.

5.3 Зміни, внесені до інтерфейсів системи

Під час розробки системи були внесені незначні зміни у розроблені в пункті 3.7.4 інтерфейси системи, викликані обмеженнями використаних засобів графічного оформлення програми (WPF).

Принципових змін до інтерфейсів системи внесено не було, тому наводити тут незначущі зміни вважаємо надлишковим.

Висновки до розділу 5

Використана для розробки системи методологія Waterfall гарно підійшла для даного дослідження. Крім того, якщо врахувати попередні його етапи, то можна дійти висновку що насправді вона застосовувалась з самого початку, ще з моменту формування вимог до системи. Можна зробити висновок, що дана методологія підходить для всього циклу проведення подібних досліджень.

Також зазначимо, що під час розробки не виникло потреби відхилятись у значущій мірі від спроєктованих архітектури, функціоналу, процесів та інтерфейсів системи, що значно пришвидшило та спростило процес розробки.

В результаті маємо систему готову до проведення тестування.

6 ТЕСТУВАННЯ СИСТЕМИ

6.1 Засоби та методики тестування

Для тестування системи був обраний підхід мануального тестування, так як він є простим в реалізації та для систем з графічним інтерфейсом невеликого масштабу досить ефективним у виявленні проблемних місць, які модульне чи юніт-тестування можуть не виявити через свою програмну природу.

В рамках тестування був перевірений функціонал, що відповідає кожній з функціональних вимог до системи, сформованих у розділі 2 (окрім тих, що на етапі проєктування були відкинуті як побічні чи позамасштабні для поточного дослідження, так як розробнику і так відомо про їх невиконання), а також суб’єктивно оцінене виконання нефункціональних вимог до системи.

6.2 Тест-кейси

Так як користувач у системі завжди один, усі тест-кейси виконуватимуться від його особи.

Усі вхідні об’єкти для тестування системи представляють собою CSV файли з числовими значеннями.

У таблиці 6.1 наведена відповідність функціональних вимог, описів тест-кейсів та результатів тестування.

Таблиця 6.1 – Опис тест-тест кейсів у відповідності до функціональних вимог до системи

Вимога	Тест-кейс	Результат
Можливість обробки табличних даних	Користувач завантажує в систему табличні дані у форматі CSV, налаштовує обробку довільним чином та отримує результат	+
Можливість оцінки	-	вимога не

Вимога	Тест-кейс	Результат
оброблюваності певним чином структурованих даних		реалізовувалась
Збереження налаштувань обробки	Користувач завантажує в систему табличні дані у форматі CSV, налаштовує обробку довільним чином та зберігає дані налаштування у файл формату .pidps	+
Портативність налаштувань обробки	Користувач переносить файл формату .pidps між машинами та просто переміщає у файловій системі та знову використовує для обробки сумісних даних у форматі CSV	+
Графічна організація процесу обробки даних	Налаштування обробки даних повністю реалізоване через графічний інтерфейс, користувач використовує його для налаштування обробки	+
Попередній перегляд результату	Підтверджується рядом попередніх тест-кейсів, в яких було проведене налаштування обробки даних: попередній перегляд результату обробки доступний одразу	+
Можливість	-	вимога не

Вимога	Тест-кейс	Результат
налаштування обробки програмним способом		реалізовувалась
Коректна обробка неіснуючих даних	Користувач завантажує файл CSV з пустими комірками та налаштує його обробку так, щоб вони в ній брали участь, після чого отримує результат обробки	+- реалізований безумовний захист внутрішніх обчислень від некоректних значень, може в результаті надавати неочікувані результати (відсутні значення приймаються за 0 або 1)
Коректна обробка та попередження користувача про некоректні значення	-	вимога не реалізовувалась
Локальність обробки даних	Користувач завантажує та обробляє дані без доступу в інтернет	+
Можливість автоматизації	Користувач завантажує один чи декілька файлів CSV з даними а	+

Вимога	Тест-кейс	Результат
обробки	також один чи декілька файлів зі збереженими налаштуваннями обробки. Після чого встановлює в графічному інтерфейсі відповідність між даними та налаштуваннями та запускає автоматизовану обробку	

Бачимо, що в ході мануального тестування було підтверджене виконання більшості функціональних вимог.

6.3 Оцінка виконання функціональних вимог

Підтверджені як невиконані в ході тестування лише вимоги, які спеціально відкидались на попередніх етапах, тому результат тестування системи можемо вважати задовільним.

6.4 Оцінка виконання нефункціональних вимог

Оцінку виконання поставлених нефункціональних вимог наведемо в таблиці 6.2.

Таблиця 6.2 – Оцінка виконання нефункціональних вимогам

Вимога	Оцінка	Результат
Дружність	Інтерфейс не перевантажений та слідує	+

Вимога	Оцінка	Результат
інтерфейсу	сучасним контрактам дизайну комп’ютерних програм	
Інтуїтивність інтерфейсу	Інтерфейс інтуїтивний, кожна функція чітко позначена та з питаннями допомагає функція допомоги	+
Відносна швидкість роботи	Час обробки даних невеликого об’єму (50-100 рядків) становить < 5 секунд	+
Точність обчислень	Точність обчислень з цілими та дійсними числами забезпечується типами даних мови C#	+
Імутабельність оригінальних файлів	Файли з даними, завантажені користувачем ніяк не змінюються системою під час обробки	+
Швидкість налаштування обробки	Налаштування обробки структури даних файлу невеликого розміру (50-100 рядків) значно швидше за проведення ручної обробки тих самих даних	+
Кросплатформеність	Розроблений варіант системи не є кросплатформеним	-
Керування помилками	У випадку виникнення помилок, що перешкоджають роботі системи, вона самостійно їх локалізує, попереджуючи про помилку користувача	+

Бачимо, що з нефункціональних не виконана тільки вимога з кросплатформеності програми, яку було відкинуто на етапі вибору технологій та ПЗ для розробки системи.

Висновки до розділу 6

Результати тестування системи показали, що всі задумані до виконання на етапі проєктування системи функціональні та нефункціональні вимоги були майже в повній мірі виконані (окрім свідомо відкинутих та визначених як другорядні), що свідчить про успішну реалізацію системи.

Зазначимо, що наявні пропуски у виконанні поставлених функціональних вимог легко виправляються проведенням ще однієї ітерації методології водоспаду, в якій вже було б заплановано реалізацію пропущених вимог. Враховуючи результати, вважаємо цю доробку такою, що виходить за межі даного дослідження.

З іншого ж боку, для виконання відкинутої нефункціональної вимоги про кросплатформеність доведеться змінити технології, що використовувались для розробки графічного інтерфейсу та повністю його переробити.

7 РОЗРОБЛЕННЯ СТАРТАП-ПРОЄКТУ

7.1 Опис ідеї проєкту

В цьому підрозділі будуть розглянуті:

– зміст ідеї стартап-проекту;

– напрямки використання стартап-проекту;

– переваги для кінцевого користувача;

– відмінність від аналогів.

Перші три з 4 пунктів опишемо в таблиці 7.1.

Таблиця 7.1 – Опис ідеї та переваг стартап-проєкту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Система для обробки структурованих та напівструктурованих даних, спрямована на дообробку та перетворення малих та середніх об’ємів даних з допомогою графічного інтерфейсу	Первинна обробка невеликих об’ємів даних	Нема необхідності в складній чи дорогій системі, швидке отримання результату
	Дообробка отриманих з різних джерел даних	Швидка обробка даних для отримання інформації та підтримки прийняття рішень
	Перетворення структури даних	Економія часу для переведення одного виду звітності в іншу аналогічну
	Автоматизація обробки невеликих стандартизованих файлів	Прискорення виконання невеликих мануальних завдань

Основним і єдиним застосуванням задуманої системи є обробка даних у різному вигляді, тому всі вигоди для користувачів так чи інакше пов’язані з отриманням результатів такої обробки для різних форм даних.

Для аналізу техніко-економічних переваг ідеї перед конкурентами повернемось до розділу 1 та оберемо конкурентні товари, з якими порівнюватимемо майбутню розробку.

Загальні мови програмування відкинемо, так як вони пропонують інструмент для створення рішення а не саме рішення. Також не братимемо до уваги бази даних, так як різниця у масштабі та вартості занадто велика.

Таким чином, проведемо порівняння задуманого проєкту з Google Sheets з скриптами, з MS Excel з скриптами та з R та RStudio (таблиця 7.2).

Таблиця 7.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проєкту

№	Техніко-економічні характеристики ідеї	Власний проєкт	Товари/концепції конкурентів			W	N	S
			G-Sheets	MS Excel	R + Studio			
1	Спрямованість на надання користувачам засобів обробки даних	Так	Так	Так	Так		+	
2	Можливість прямого редагування даних	Ні	Так	Так	Ні	+		
3	Можливість автоматизації обробки даних без додаткових знань	Так	Ні	Ні	Ні			+
4	Відносно низька	Так	Так	Ні	Так		+	

№	Техніко-економічні характеристики ідеї	Власний проєкт	Товари/концепції конкурентів			W	N	S
			G-Sheets	MS Excel	R + Studio			
	вартість продукту							
5	Наявність клієнтської бази	Ні	Так	Так	Так	+		
6	Standalone додаток без додаткових встановлень та залежностей	Так	Ні	Ні	Так			+
7	Автоматизована обробка файлів без їх відкриття	Так	Ні	Ні	Так			+

Хоча і існує деякий перетин функціоналу з розглянутими конкурентними продуктами, підхід до його реалізації в задуманому проєкті принципово відрізняється. Крім того задуманий і унікальний функціонал, тому можна вважати ідею конкурентоспроможною.

7.2 Технологічний аудит ідеї проєкту

Проведемо тепер аналіз технологічної здійсненності ідеї проєкту, а саме вирішимо за яким технологіями реалізовуватиметься частина ідеї проєкту, чи існують ці технології та чи вони доступні авторам проєкту (таблиця 7.3).

Таблиця 7.3 – Технологічна здійсненність ідеї проєкту

№	Ідея проєкту	Технології і реалізації	Наявність технологій	Доступність технологій
1	Формат представлення структурованих даних	CSV, TSV, XLS, XLSX, ODS та ін.	Наявні, дороблювати не потрібно	Доступні
2	Обробка структурованих даних з допомогою графічного інтерфейсу	WPF, MAUI, інші.	Наявні, дороблювати не потрібно	Доступні
3	Мова програмування та фреймворк для розробки комп’ютерних програм	C#, .Net 8.0	Наявні, дороблювати не потрібно	Доступні
4	Мова розмітки для розробки графічного інтерфейсу	XAML	Наявні, дороблювати не потрібно	Доступні
5	Формат даних для збереження налаштувань обробки	.pidps	Не наявні, необхідна розробка	Доступні за умови розробки
6	Допоміжні бібліотеки для обробки даних	Бібліотеки з відкритим вихідним кодом або надані Microsoft	Наявні, дороблювати не потрібно	Доступні
Обрані технології реалізації ідеї проєкту: мова програмування C#; мова розмітки XAML; Фреймворки .Net, WPF; формат даних CSV; формат налаштувань обробки пропрієтарний .pidps; бібліотека VisualBasic.FileIO та ін.				

За результатами аналізу технологічної здійсненності проєкту можемо побачити, що фактично всі елементи, необхідні для його реалізації вже наявні та є доступними для використання, окрім власного формату збережених налаштувань обробки даних, який є текстовим та в принципі не дуже складним в реалізації.

Тому ідею проєкту вважатимемо технологічно здійсненною.

7.3 Аналіз ринкових можливостей запуску проєкту

Проаналізуємо ринкові можливості проєкту, почавши з розгляду попиту на продукт (таблиця 7.4).

Таблиця 7.4 – Попередня характеристика потенційного ринку стартап-проєкту

№	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	< 4
2	Загальний обсяг продаж, доларів	60 мільярдів
3	Динаміка ринку	Зростає
4	Наявність обмежень для входу	Мінімальні, зокрема відомість бренду
5	Специфічні вимоги до стандартизації та специфікації	Відсутні
6	Середня норма рентабельності в галузі, %	30-40%

Результати аналізу потенційного ринку стартап проєкту показали ситуацію на ринку, позитивну для впровадження та подальшого розвитку ідеї стартап-проєкту.

Тепер проаналізуємо потенційних клієнтів проєкту. Їх характеристика наведена у таблиці 7.5.

Таблиця 7.5 – Характеристика потенційних клієнтів стартап-проєкту

№	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Дообробка даних організацій	Бізнеси, що мають незавершені CRM та інші системи та які потребуть швидкої реалізації дрібних операцій з обробки даних	Клієнт керується потребою отримання кінцевого результату обробки даних, без якого важко вести бізнес	Точність, швидкість роботи, безпека даних, простота у вивченні та застосуванні
2	Первинна обробка невеликих об’ємів даних		Клієнт керується потребою в обробці даних невеликих масштабів, яку дорого задовольняти іншими способами	
3	Обробка даних для особистого користування	Приватні особи	У клієнта існує особиста потреба чи інтерес до обробки даних в невеликому масштабі	Точність роботи, безпека даних, простота та зручність, ненав’язлива цінова політика

Далі проаналізуємо фактори загроз та можливостей (таблиці 7.6 та 7.7, відповідно).

Таблиця 7.6 – Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Поява конкурента	Створення іншою компанією аналогічного продукту для обробки даних	Перевірка цілісності авторських прав, розробка додаткового функціоналу, що дозволив би успішно конкурувати на ринку
2	Незацікавленість клієнтів	Клієнти не бачать сенсу купувати продукт та не роблять цього	Рекламна кампанія, проведення безкоштовних майстер-класів з використання системи а також надання безкоштовних пробних ліцензій для ознайомлення

Таблиця 7.7 – Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Вільний ринок	Відсутність на ринку продуктів зі схожим функціоналом	Ініціація впровадження використання в домашніх та закордонних компаніях для створення суттєвої клієнтської бази
2	Стало відомо про провал розробки CRM однієї з компаній	Шанс запропонувати проміжний варіант для підтримки роботи підприємства клієнта, можливість отримати пожиттєвого клієнта	Ініціація переговорів з керуючими особами компанії щодо впровадження використання системи на постійній основі, укладання контрактів на підтримку та розробку нового функціоналу

Таким чином, проєкт має наступні загрози: появу конкурентного продукту та ігнорування з боку клієнтів. Проте наявні і наступні можливості: шанси використання вільного ринку для впровадження інноваційної ідеї в багатьох підприємствах та можливість запропонувати допомогу компаніям, чиї системи обробки даних провалились в розробці в обмін на лояльність.

Тепер перейдемо до аналізу пропозиції. Почнемо із аналізу конкуренції на ринку (таблиця 7.8).

Таблиця 7.8 – Ступеневий аналіз конкуренції на ринку

№	Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1	Тип конкуренції: монополія	Відсутні продукти з принципово ідентичним функціоналом	Спостереження за появою конкурентів, нарощування клієнтської бази
2	Рівень конкурентної боротьби: міжнародний	Основні конкуренти з дотичним функціоналом це міжнародні компанії	Розширення клієнтської бази в національних рамках для подальшого виходу з допомогою гарної репутації на міжнародний ринок
3	Галузева ознака: внутрішньогалузева	Конкуренція тільки в галузі обробки даних	Розробка та впровадження нового функціоналу для забезпечення гнучкості системи
4	Конкуренція за видами товарів: товарно-видова	Конкуренція в рамках одного виду товару	Розробка нових функцій для отримання переваги над конкурентами

№	Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
5	За характером конкурентних переваг: неціновa	Конкуренти з ідентичним функціоналом відсутні	Нема потреби в регулюванні ціни на продукт у відповідь на конкурентів
6	За інтенсивністю: не марочна	Користувачів цікавить функціонал системи а не її виробник	Розширення функціоналу для захоплення більшої кількості клієнтів

Тепер проведемо детальніший аналіз умов конкуренції в галузі за моделлю 5 сил М. Портера (таблиця 7.9).

Таблиця 7.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Відсутні	Google, Microsoft	Відсутні	Функціонал, зміни ціни	Відсутні
Висновки	Прямі конкуренти в галузі відсутні	Можуть увійти у ринок, імовірність невелика	У постачальниках немає потреби	Клієнти хочуть мінімізувати витрати	Товари замінники відсутні

За результатом аналізу конкуренції в галузі, можемо сказати, що робота компанії на ринку є можливою. Для конкурентоспроможності, стартапу необхідно

буде запровадити систему встановлення цін, яка б відображала можливості потенційних клієнтів а також постійно розвивати функціонал для утримання унікальної ніші.

Зважаючи на викладене вище в розділі, проведемо обґрунтування конкурентоспроможності проєкту (таблиця 7.10).

Таблиця 7.10 – Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор значущим для порівняння конкурентних проєктів)
1	Ціна продукту	Одним з ключових факторів при виборі продуктів схожого виду є їх ціна, так як навіть великі компанії не бажають переплачувати за продукти, що використовують. Нижча ціна продукту стане необхідним поштовхом до спроби його використання
2	Актуальність	Обробка даних зараз всюди, так само як і незавершені автоматизовані системи, тому попит на товар, що пропонується існуватиме
3	Простота впровадження	Система інтуїтивна, легка до вивчення та не ресурсомістка, що означає зменшення витрат на її підтримку в довготривалому використанні
4	Достатній та розщирюваний функціонал	Функціонал системи цілком задовольняє основні потреби з обробки структурованих даних, але компанія не закрита до пропозицій та може розробляти наприклад ексклюзивний функціонал для своїх клієнтів, а також випускати оновлення для всіх

За визначеними факторами конкурентоспроможності проведемо аналіз сильних та слабких сторін стартап-проєкту (таблиця 7.11)

Таблиця 7.11 – Порівняльний аналіз сильних та слабких «Process It!»

№	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні						
			-3	-2	-1	0	1	2	3
1	Ціна продукту	15	+						
2	Актуальність	16			+				
3	Простота впровадження	18	+						
4	Достатній та розщирюваний функціонал	13				+			

Нарешті для завершення ринкового аналізу проведемо SWOT-аналіз проєкту (таблиця 7.12).

Таблиця 7.12 – SWOT-аналіз стартап-проєкту

Сильні сторони: ціна продукту, актуальність функціоналу, простота впровадження	Слабкі сторони: необхідність у розширенні функціоналу
Можливості: відсутність прямих аналогів продукту, мала конкуренція	Загрози: можливість появи конкурентів з ширшим функціоналом

На основі проведеного SWOT-аналізу розробимо альтернативи ринкової поведінки для виведення стартап-проєкту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проєкти конкурентів.

Далі визначені альтернативи проаналізуємо з точки зору строків та імовірності отримання ресурсів (таблиця 7.13).

Таблиця 7.13 – Альтернативи ринкового впровадження стартап-проєкту

№	Альтернатива ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Впровадження безкоштовного пробного періоду користування продуктом на 2 тижні	Висока	3-4 тижні
2	Перемовини з національними бізнесами щодо зацікавленості у використанні системи	Середня	1-2 місяці
3	Таргетована реклама в соціальних мережах	Середня	5-6 місяців
4	Демонстрація продукту на міжнародних тематичних виставках	Висока	1-2 роки

Як найефективніший за обома параметрами варіант, можна обрати першу альтернативну ринкову поведінку, а саме впровадження безкоштовного пробного періоду використання системи на 2 тижні.

7.4 Розроблення ринкової стратегії проєкту

Почнемо розроблення ринкової стратегії проєкту з визначення охоплення ринку, тобто опису цільових груп потенційних споживачів (таблиця 7.14).

Таблиця 7.14 – Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Великі компанії	+	Середній	Висока	Середня
2	Невеликі компанії	+	Високий	Низька	Висока
3	Окремі працівники	+	Середній	Низька	Висока
4	Окремі ентузіасти	+	Низький	Висока	Середня
Які цільові групи обрано: великі та невеликі компанії, окремі працівники					

За результатами аналізу цільових груп було вирішено обрати стратегію диференційованого маркетингу для успішної роботи з кількома сегментами ринку. Для розвитку в обраних сегментах ринку сформуємо базову стратегію розвитку (таблиця 7.15).

Таблиця 7.15 – Визначення базової стратегії розвитку

Обрана альтернатива розвитку проєкту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку

Обрана альтернатива розвитку проєкту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Впровадження безкоштовного пробного періоду користування продуктом на 2 тижні	Диференці йований маркетинг	Унікальність та повсюдне використання у зв'язку з високою доступністю продукту, зменшена чутливість до ціни та збільшена клієнтська база	Стратегія диференціації

Тепер оберемо та опишемо стратегію конкурентної поведінки (таблиця 7.16).

Таблиця 7.16 – Визначення базової стратегії конкурентної поведінки

Чи є проєкт «першопрохідцем» на ринку	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
Так	Шукати нових а також забирати існуючих споживачів у конкурентів	Так, підтримувані формати даних та опції обробки	Стратегія зайняття конкурентної ніші

За результатами проведеного аналізу в таблицях 7.5, 15 та 16 розробимо стратегію позиціонування стартап-проєкту на ринку (таблиця 7.17).

Таблиця 7.16 – Визначення базової стратегії конкурентної поведінки

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспромо жні позиції власного стартап-проєкту	Вибір асоціацій, які мають сформувати комплексну позицію власного проєкту

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проєкту	Вибір асоціацій, які мають сформувати комплексну позицію власного проєкту
Зручність використання, швидка робота, невисока ціна, надійність	Стратегія диференціації	Функціонал швидкої обробки структурованих даних, ціна, простота у впровадженні	1 – простота у використанні 2 – обширний функціонал 3 – низька ціна

В результаті маємо узгоджену систему рішень щодо ринкової поведінки стартап-компанії у різних ситуаціях, яка визначатиме напрями роботи компанії на ринку.

7.5 Розроблення маркетингової програми проєкту

Для початку підсумуємо результати аналізу конкурентоспроможності товару з попереднього розділу (таблиця 7.18).

Таблиця 7.18 – Альтернативи ринкового впровадження стартап-проєкту

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами
1	Дообробка даних для підприємств	Швидкий та простий спосіб реалізувати кінцеву обробку даних для бізнесу	Локальність, низька ціна, низький поріг входу для використання, підтримка та розробка нових функцій для клієнтів
2	Первинна	Легка в освоєнні	Гарантована приватність даних,

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами
	обробка даних для приватних осіб	система, що не відправляє дані по мережі інтернет	інтуїтивний інтерфейс, портативність та малий розмір програми. Можливість одночасної обробки декількох файлів

Тепер розробимо трирівневу маркетингову модель товару (таблиця 7.19).

Таблиця 7.19 – Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="3">Головна функція системи: спростити та автоматизувати обробку структурованих даних малих масштабів</td></tr>
<tr><td rowspan="5">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх/Тл/Е/Ор</td></tr>
<tr><td>1. Інтуїтивний інтерфейс
2. Безпека даних
3. Портативність
4. Дешевизна</td><td>М
Нм
М
М</td><td>Е
Тх
Тл
Економічний</td></tr>
<tr><td colspan="3">Якість: тестування системи (ручне та автоматичне)</td></tr>
<tr><td colspan="3">Пакування: архів з інсталятором</td></tr>
<tr><td colspan="3">Марка: Process It!</td></tr>
<tr><td rowspan="2">III. Товар із підкріпленням</td><td colspan="3">До продажу: тестовий період 3 тижні, секція допомоги</td></tr>
<tr><td colspan="3">Після продажу:
– повний доступ до використання системи;
– оновлення функціоналу;
– допомога у використанні;</td></tr>
</table>

Рівні товару	Сутність та складові
	– розробка додаткових функцій.
За рахунок чого потенційний товар буде захищено від копіювання: авторське право, патентування, DRM- та інший захист від піратства	

Тепер перейдемо до визначення цінових меж потенційного товару (таблиця 7.20).

Таблиця 7.20 – Визначення меж встановлення ціни

Рівень цін на товари замінники	Рівень цін на товари аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
Товари замінники на ринку відсутні	~70 доларів на рік, ~7 доларів на місяць	Залежить від масштабу організації	20-50 доларів на рік (2-5 доларів на місяць)

Далі визначимо оптимальну систему збуту товару (таблиця 7.21).

Таблиця 7.21 – Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Купівля ліцензії онлайн з підпискою використання системи	Реклама товару, залучення нових клієнтів, дистрибуція пробних версій системи	Переговори напряму з компаніями, через посередників. Чим глибше тим більше потенційних клієнтів	Власна онлайн система збуту, побудована на зв’язках та рекламі

Нарешті, розробимо для стартап-проєкту концепцію маркетингових комунікацій, що спирається на попередньо визначені системи та стратегії (таблиця 7.22).

Таблиця 7.22 – Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікації, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
Зацікавленість у спрощенні ручної роботи за комп'ютером	Соціальні мережі, месенджери, електронна пошта, телефон	Реклама в соціальних мережах, месенджерах, регулярне висвітлення інформації про оновлення системи на сайті	Зацікавити потенційних покупців спробувати пробну версію системи	Розповідь про функціонал продукту, згадка про пробну версію. Висвітлення інтуїтивності та простоти інтерфейсу

Висновки до розділу 7

Аналіз стартап-проєкту, заснованого на ідеї магістерської дисертації показав, що такий проєкт має досить високу рентабельність та великі шанси на комерціалізацію, оскільки на даний момент на ринку відсутні продукти, функціонал яких би віддзеркалював функціонал системи, задуманої в стартап-проєкті.

Хоча прямих конкурентів у проєкту нема, наявні конкуренти що пропонують дотичний функціонал. Проте вони націлені на інші цільові аудиторії, тому не загрожують успішності проєкту на ринку.

Крім того, визначені альтернативи ринкового впровадження проєкту дозволяють скласти план дій навіть на випадок якщо конкуренти зі схожим функціоналом все таки з’являться.

Загалом, умови для впровадження проєкту є цілком сприятливими, а імплементація розробленого стартап-проєкту є доцільною з комерційної точки зору.

ВИСНОВКИ

1. Розгляд проблеми обробки структурованих даних малого масштабу, що полягає у необхідності простого та доступного способу приведення даних до бажаного виду при відсутності чи недостатності автоматизованих програмних засобів (дані, отримані після обробки з їх допомогою не представляють собою кінцеву ціль обробки), показав, що принцип роботи майже всіх доступних засобів досягнення мети «дообробки» даних заснований на ручній обробці даних, які часто надаються у вигляді, не готовому до сприймання людиною. Саме тому було вирішено в рамках даного дослідження спроєктувати та розробити робочу версію автоматизованої системи, яка б дозволила спростити та пришвидшити обробку невеликих об’ємів структурованих даних.

2. Основної причиною наукової та практичної проблеми, що розглядається в даному дослідженні виступають застарілі, провальні, непідтримувані, занадто складні для модифікації або з інших причин нерухомі (в плані реалізації нового функціоналу та адаптації до нових вимог старого) інформаційні системи. Вони часто продукують дані, які частково або зовсім не відповідають кінцевим вимогам користувачів таких систем, що значно збільшує час обробки даних. Це, в свою чергу, значно сповільнює швидкість прийняття рішень, що залежать від цих даних на підприємствах. Такі затримки можуть з певною імовірністю виливатись у грошові втрати та помилки у веденні бізнесу, що безперечно є негативним фактором, який, з допомогою результатів даного дослідження ми намагались мінімізувати.

3. Сформовані та виконані під час розробки функціональні та нефункціональні вимоги для задуманої системи на думку дослідника виступають мінімальним набором функціоналу та властивостей програмного продукту, який необхідно мати розробкам, автори яких хочуть в подальшому реалізовувати аналогічний функціонал з обробки даних.

4. Основною ціллю дослідження була реалізація системи, яка продукує схожі або ідентичні до розглянутих аналогів результати, але при цьому пропонує принципово новий метод їх досягнення. Розроблена система реалізує підхід,

схожий за своїм принципом на декларативне програмування, тобто, для того щоб отримати результат необхідно саме його, а не необхідні кроки його досягнення задати системі. Такий підхід значно спрощує поріг входу до автоматизації обробки даних.

5. Розроблений варіант спроєктованої системи показав що, незважаючи на невиконання декількох з поставлених до нього вимог та обмеженість функціоналу у зв'язку з часовими рамками дослідження, задуманий підхід до організації обробки даних значно спрощує даний процес для користувача. Крім того, в разі подальшого розвитку реалізації системи для розширення її функціоналу до рівня конкурентів, вона може привабливішою для потенційних клієнтів, які вже задовольняють свої потреби з обробки даних з допомогою інших програмних засобів, за згадані засоби через простоту використання.

5. Розробка стартап-проєкту для потенційного виведення спроєктованої системи на ринок показала адекватно високу рентабельність запуску такого проєкту у зв'язку з унікальністю функціоналу (а скоріше, як вже зазначалось вище, підходу до його надання користувачам) та, що випливає з попереднього, відсутності прямих конкурентів. Крім того, варто зазначити, що при перенесенні основної ідеї системи на інші платформи дистрибуція, наприклад SaaS (система як сервіс), капіталізація розробленого стартап-проєкту стала б ще успішнішою за умовний прогнозований рівень.

6. Загалом вважаємо цілі дослідження досягнутими а його саме завершеним. Проте, запропонуємо декілька напрямків в яких його можна було б розвинути в майбутньому:

– реалізація системи для ширшого спектру видів даних;

– оптимізація швидкодії ядра системи для створення можливості обробки більших об'ємів даних;

– інтеграція моделей машинного навчання для генерації потенційних варіантів обробки даних без налаштування користувачем.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Susanto A. How business use information systems?. International journal of scientific & technology research. 2019. Vol. 8, no. 1. URL: https://www.ijstr.org/final-print/jan2019/How-Business-Use-Information-Systems.pdf.

2. Hwang Y. Data - the modern world's most precious resource. IoT For All. URL: https://www.iotforall.com/data-worlds-most-precious-resource.

3. Kumar A. Finishing the unfinished crm system. LinkedIn. URL: https://www.linkedin.com/pulse/finishing-unfinished-crm-system-ashish-kumar.

4. Marks G. 48% Of Sales Leaders Say Their CRM System Doesn't Meet Their Needs. The Good News Is That This Is Fixable. LinkedIn.com. URL: https://www.linkedin.com/pulse/48-sales-leaders-say-crm-system-doesnt-meet-needs-good-gene-marks-cpa-1c.

5. Herrick J. CRM and the big impact of small data - benchmarkone. BenchmarkONE. URL: https://www.benchmarkone.com/blog/crm-big-impact-small-data/.

6. Kumari R. Top 10 companies that uses big data. Analytics Steps - A leading source of Technical & Financial content. URL: https://www.analyticssteps.com/blogs/companies-uses-big-data.

7. Gordon S. 5 ways small data can be more valuable than big data. CRM Magazine August 22, 2014. URL: https://www.destinationcrm.com/Articles/Web-Exclusives/Viewpoints/5-Ways-Small-Data-Can-Be-More-Valuable-than-Big-Data-98927.aspx.

8. Wonderflow. What is small data (in just 4 minutes). Wonderflow.ai. URL: https://www.wonderflow.ai/blog/what-is-small-data.

9. MyDataModels | Website. URL: https://www.mydatamodels.com/wp-content/uploads/2020/02/Small-Data-1536x983.jpg.

10. Guardian staff reporter. Forget big data, small data is the real revolution. the Guardian. URL: https://www.theguardian.com/news/datablog/2013/apr/25/forget-big-data-small-data-revolution.

11. Disentangling the structure of tables in scientific literature / N. Milosevic et al. Natural language processing and information systems. Cham, 2016. P. 162–174. URL: https://doi.org/10.1007/978-3-319-41754-7_14.

12. What is unstructured data?. MongoDB. URL: https://www.mongodb.com/unstructured-data.

13. Apps Script | Google for Developers. Google for Developers. URL: https://developers.google.com/apps-script.

14. Google Apps Script overview | Google for Developers. Google for Developers. URL: https://developers.google.com/apps-script/overview (date of access: 06.01.2024).

15. Harvey G. AARP Excel 2010 for Dummies. Wiley & Sons, Incorporated, John, 2011. 64 p.

16. Gharani L. Excel VBA vs Office scripts. Xelplus. URL: https://www.xelplus.com/excel-vba-vs-office-scripts/.

17. Newman C. Will VBA Die in 2022? Is It Worth Learning? [Inside Scoop]. The Spreadsheet Guru. URL: https://www.thespreadsheetguru.com/are-vba-macros-dead/.

18. LibreOffice 7.4 SDK - Overview. api.libreoffice.org. URL: https://api.libreoffice.org/.

19. S (programming language). www.immagic.com. URL: https://www.immagic.com/eLibrary/ARCHIVES/GENERAL/WIKIPEDI/W120512S.pdf.

20. R: What is R?. R: The R Project for Statistical Computing. URL: https://www.r-project.org/about.html.

21. GitHub - rstudio/rstudio: RStudio is an integrated development environment (IDE) for R. GitHub. URL: https://github.com/rstudio/rstudio/.

22. MATLAB. MathWorks - Makers of MATLAB and Simulink - MATLAFast Track to MDXB & Simulink. URL: https://www.mathworks.com/products/matlab.html.

23. Power Query documentation - Power Query. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/power-query/.

24. Whitehorn M., Zare R., Pasumansky M. Fast Track to MDX. London : Springer London, 2006. URL: https://doi.org/10.1007/1-84628-182-2.

25. Data Analysis Expressions (DAX) Reference - DAX. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/dax/.

26. Pham D. How a SQL database engine works. Medium. URL: https://medium.com/@grepdennis/how-a-sql-database-engine-works-c67364e5cdfd.

27. Sharif A. 6 Common Log File Formats - CrowdStrike. crowdstrike.com. URL: https://www.crowdstrike.com/cybersecurity-101/observability/log-file-formats/.

28. Shafranovich Y. Common Format and MIME Type for Comma-Separated Values (CSV) Files. RFC Editor, 2005. URL: https://doi.org/10.17487/rfc4180.

29. OpenRefine. OpenRefine. URL: https://openrefine.org/.

30. CSV Comma Separated Value File Format - How To. Creativyst Software - Explored, Designed, Delivered.(sm). URL: https://www.creativyst.com/Doc/Articles/CSV/CSV01.shtml.

31. Log data structure. Designing highly-scalable applications. URL: https://scaling.dev/storage/log.

32. Structured logging - definition & overiew | Sumo Logic. Sumo Logic. URL: https://www.sumologic.com/glossary/structured-logging/.

33. C# docs - get started, tutorials, reference. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/dotnet/csharp/.

34. Visual Studio: IDE and Code Editor for Software Developers and Teams. Visual Studio. URL: https://visualstudio.microsoft.com/.

35. Pricing and Purchasing Options | Visual Studio. Visual Studio. URL: https://visualstudio.microsoft.com/vs/pricing/.

36. GitHub: Let’s build from here. GitHub. URL: https://github.com/.

37. Git. Git. URL: https://git-scm.com/.

38. What is Standalone Application? - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/what-is-standalone-application/.

39. Nikolaieva A. 8 Best Software Development Methodologies | Uptech. Mobile and web app development company | Uptech. URL: https://www.uptech.team/blog/software-development-methodologies.

40. Кафедра Інформаційних Систем та Технологій. URL: https://ist.kpi.ua/.